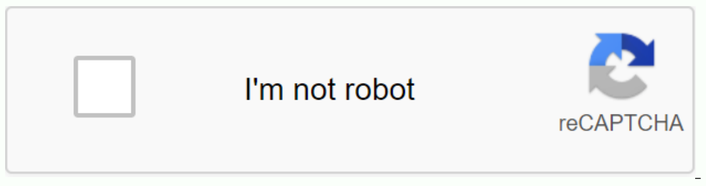
Webview mime type android



Continue

A view that displays web pages. In most cases, we recommend using a standard web browser, such as Chrome, to deliver content to you. To learn more about web browsers, read the guide to calling your browser with intent. WebViews allow web content to be displayed as part of an activity layout, but some features of fully expanded browsers are missing. WebView is useful when you need increased UI control and advanced configuration options that allow you to embed web pages in a specially designed environment for your application. To learn more about webview and alternatives to displaying web content, read the Web-based content documentation. WebView.FindListener Interface to listen for results. Class WebView.HitTestResult WebView.PictureListener This interface has been deprecated in api level 12. This interface is now deprecated. The WebView.VisualStateCallback class provides a callback interface delivered to WebView.postVisualStateCallback(long, WebView.VisualStateCallback) to receive visual status notifications. The WebView.WebViewTransport class transports the object to return the WebView across thread boundaries. From the android.view.View class android:accessibilityUd whether this view is a header for accessibility purposes. android:accessibilityLiveRegion Indicates to accessibility services whether the user should receive notifications when this view is changed. android:accessibilityPaneTitle This title should represent accessibility as a pane title. android:accessibilityTraversalPo sets the view ID after which it is visited in traversal accessibility. android:accessibilityTraversalBefore Sets the view ID before which this is visited in traversal accessibility. android:alpha alpha alpha view property, as a value between 0 (completely transparent) and 1 (completely opaque). android:autofillHints Describes the contents of the view so that the AutoComplete service can fill in the relevant data. android:autofilledHighlight Drawable to draw in view to mark it as autofilled Can be a reference to another resource, in form @[+][package:]type/name or theme attribute in form ? [package:]type/name. android:background Drawn for use as background. android:backgroundTint Tint for background use. android:backgroundTintMode Blending mode used to apply background tone. android:clickable Specifies whether this view responds to click events. android:contentDescription Defines text that briefly describes the contents of a view. android:contextClickable Specifies whether this view responds to context click events. android:defaultFocusHighlightEnabled Should this view the default focus highlight when concentrated, but there is no R.attr.state_focused defined in the background. android:drawingCacheQuality Determines quality drawing caches. android:duplicateParentState When this attribute is set to true, the view retrieves its resulting state (focused, pressed, etc.) from its immediate parent rather than from itself. android:base elevation with depth view. android:fadeScrollbars Specifies whether scroll bars fade when not in use. android:fadingEdgeLength Defines the length of fading edges. android:filterTouchesWhenObscured Specifies whether to filter touches when the view window is obscured by another visible window. android:fitsSystemWindows Boolean internal attribute to customize the view layout based on system windows, such as the status bar. android:focusable Specifies whether the view can focus. android:focusableInTouchMode Boolean, which controls whether the view can focus in touch mode. android:focusedByDefault Is this view the default focus view. android:forceHasOverlappingRendering Does this view have elements that can overlap when drawn. android:fore plan Defines drawable for drawing over content. android:foregroundGravity Defines gravity to apply to the foreground drawable. android:foregroundTint Tint to apply to foreground. android:foregroundTintMode Blending mode used to apply foreground hue. android:hapticFeedbackEnabled Boolean, which controls whether the view should have touch feedback enabled for events such as long presses. android:id Provide an ID name for this view to retrieve it later from View.findViewById() or Activity.findViewById(). android:importantForAccessibility Describes whether this view is important for accessibility. android:importantForAutofill Android hints that the view node associated with this view should be included in the view structure used for autocomplete purposes. android:importantForContentCapture tells Android whether the view node associated with this view should be used for content capture purposes. android:isScrollContainer Set this if the view serves as a scroll container, which means that you can shrink its general window so that there is room for the input method. android:keepScreenOn Specifies whether the view window should keep the screen on the screen when it is visible. android:keyboardNavigationCluster Is this view the root of the keyboard navigation cluster. android:layerType Specifies the type of layer that is in this view. android:layoutDirection Defines the direction of the layout drawing. android:longClickable Specifies whether this view responds to long-click events. android:minHeight Defines the minimum view height. android:minWidth Defines the minimum view width. android:nextClusterForward Defines the next keyboard navigation cluster. android:nextFocusDown Defines the next view to focus on when the next focus is View.FOCUS_DOWN if the reference refers to the that does not exist or is part of a hierarchy that is invisible, RuntimeException will cause when a reference is available. android:nextFocusForward Defines the next view to give focus when the next focus is View.FOCUS_FORWARD If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, RuntimeException will cause when the reference is available. android:nextFocusLeft Defines the next view to focus on when the next focus is View.FOCUS_LEFT. android:nextFocusRight Defines the next view to give focus when the next focus is View.FOCUS_RIGHT If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, RuntimeException will cause when the reference is available. android:nextFocusUp Defines the next view to give focus when the next focus is View.FOCUS_UP If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, RuntimeException will cause when the reference is available. android:onClick The name of the method in the context of this view to call when the view is clicked. android:outlineAmbientShadowColor Sets the ambient shadow color that is drawn when the view has a positive Z or elevation value. android:outlineSpotShadowColor Sets the color of the spot shadow that is drawn when the view has a positive Z or elevation value. android:padding Sets the padding, in pixels, of all four edges. android:paddingBottom Sets padding, in pixels, to the bottom edge; see R.attr.padding. android:paddingEnd Sets padding, in pixels, of the end edge; see R.attr.padding. android:paddingHorizontal Sets padding, in pixels, left and right edges; see R.attr.padding. android:paddingLeft Sets the left edge padding; see R.attr.padding. android:paddingRight Sets the padding of the right edge in pixels; see R.attr.padding. android:paddingStart Sets the padding, in pixels, of the start edge; see R.attr.padding. android:paddingTop Sets the padding, in pixels, of the top edge; see R.attr.padding. android:paddingVertical Sets padding, in pixels, top and bottom edge; see R.attr.padding. android:requiresFadingEdge defines which edges should be faded as you scroll. android:rotating view rotation in degrees. android:rotationX rotate the view around the x axis, in degrees. android:rotationY Rotate the view around the y-axis, in degrees. android:saveEnabled If false, no state will be saved for this view when it is locked. android:scaleX view scale toward x. android:scaleY view scale towards y. android:screenReaderFocusable Should this view be treated as a focus unit by screen reader accessibility tools. android:scrollIndicators Determines which indicators should be displayed can be scrolled. android:scrollX Initial horizontal scroll offset in pixels. android:scrollY The initial vertical scroll offset in pixels. android:scrollbarAlwaysDrawHorizontalTrack Specifies whether a horizontal scroll path should always be drawn. android:scrollbarAlwaysDrawVerticalTrack Specifies whether the vertical scroll bar should always be drawn. android:scrollbarDefaultDelayBeforeFade Defines the delay in milliseconds that the scroll bar waits before fading. android:scrollbarFadeDuration Defines the delay in milliseconds that the scroll bar takes to fade. android:scrollbarSize Sets the width of vertical scroll bars and the height of horizontal scroll bars. android:scrollbarStyle Controls the style and position of the scroll bar. android:scrollbarThumbHorizontal Defines the horizontal thumb of the scroll bar for drawing. android:scrollbarThumbVertical Defines the vertical thumb of the drawing scroll bar. android:scrollbarTrackHorizontal Defines a horizontal drawable scroll bar. android:scrollbarTrackVertical Defines a vertical scroll bar for drawing. android:scrollbars Specifies which scroll bars should be displayed when scrolling or not. android:soundEffectsEnabled Boolean, which controls whether the view should have sound effects enabled for events such as clicking and touching. android:stateListAnimator Sets the state-based animator for the view. android:tag Enter a tag for this view that contains a string that you want to download later from View.getTag() or searched for using View.findViewWithTag(). android:textAlignment Defines text alignment. android:textDirection Defines the direction of the text. android:theme Specifies the theme override for the view. android:tooltipText defines the text that appears in a small pop-up window when you hover over or press for a long time. android:transformPivotX x the location of the pivot point around which the view will rotate and scale. android:transformPivotY y the location of the pivot point around which the view will rotate and scale. android:transitionName names view so that it can be identified for transition. android:translationX translation in x view. android:translationY translation in y view. android:translationZ translation in from view. android:visibility Controls the initial visibility of the view. From the class android.view.ViewGroup int CLIP_TO_PADDING_MASK we trim to padding when FLAG_CLIP_TO_PADDING and FLAG_PADDING_NOT_NULL are set at the same time. int FOCUS_AFTER_DESCENDANTS This view will focus only if none of its children want it. int FOCUS_BEFORE_DESCENDANTS this view will focus before each of its children. int FOCUS_BLOCK_DESCENDANTS this view will block any of its children from getting focus, even if they are focusable. Int This constant is layoutMode. int LAYOUT_MODE_OPTICAL_BOUNDS This constant is int PERSISTENT_ALL_CACHES this constant has been deprecated at API level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, such as alpha animations, View.setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call view.draw(android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. int PERSISTENT_ANIMATION_CACHE This constant has been deprecated at API level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, such as alpha animations, View.setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call view.draw(android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. int PERSISTENT_NO_CACHE this constant has been deprecated at API level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers for example, for alpha animations, View.setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call view.draw(android.graphics.Canvas) in the view. However, these program-rendered uses are discouraged have compatibility with hardware-only rendering features, such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. int PERSISTENT_SCROLLING_CACHE This constant has been deprecated at API level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, such as alpha animations, View.setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call view.draw(android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. The android.view.View int class ACCESSIBILITY_LIVE_REGION_ASSERTIVE live region mode, specifying that accessibility services should interrupt ongoing speech to immediately announce changes to that view. int ACCESSIBILITY_LIVE_REGION_NONE live region mode that specifies that accessibility services should not automatically post changes to this view. int ACCESSIBILITY_LIVE_REGION_POLITE live region mode that specifies that accessibility services should post changes to this view. int AUTOFILL_FLAG_INCLUDE_NOT_IMPORTANT_VIEWS flag asking you to add views that are marked as not valid for autocomplete (see setImportantForAutofill(int)) to the view structure. A AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DATE a hint indicating that this view can be automatically populated with the expiration date of the credit card. The AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DAY a hint indicating that this view can be automatically filled in on the expiration date of the credit card. The AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_MONTH a hint indicating that this view can be automatically populated with the month the credit card expires. The AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_YEAR a hint indicating that this view can be automatically populated with the credit card expiration year. A AUTOFILL_HINT_CREDIT_CARD_NUMBER a hint indicating that this view can be credit card number. A AUTOFILL_HINT_CREDIT_CARD_SECURITY_CODE a hint indicating that this view can be automatically populated with a credit card security code. The AUTOFILL_HINT_EMAIL_ADDRESS a hint indicating that this view can be automatically populated with an email address. The AUTOFILL_HINT_NAME a hint indicating that this view can be with the user's real name. The AUTOFILL_HINT_PASSWORD a hint indicating that this view can be automatically filled with a password. The AUTOFILL_HINT_PHONE a hint indicating that this view can be automatically filled with a phone number. The AUTOFILL_HINT_POSTAL_ADDRESS a hint indicating that this view can be automatically populated with a postal address. A AUTOFILL_HINT_POSTAL_CODE a hint indicating that this view can be automatically populated with a zip code. The AUTOFILL_HINT_USERNAME a hint indicating that this view can be automatically populated with a user name. int AUTOFILL_TYPE_DATE autocomplete type for a field containing a date that is represented by a long number representing the number of milliseconds from the standard base time known as epoch, namely January 1, 1970, 00:00:00 GMT (see Date.getTime(). int AUTOFILL_TYPE_LIST AutoComplete a checklist box type that is populated by an int representing an element index inside the list (starting with 0). int AUTOFILL_TYPE_NONE autocomplete type for views that cannot be automatically populated. int AUTOFILL_TYPE_TEXT AUTOFILL_TYPE_TOGGLE Autofill type for a togglable field that is populated by a boolean. int DRAG_FLAG_GLOBAL A flag indicating that dragging can cross window boundaries. int DRAG_FLAG_GLOBAL_PERSISTABLE_URI_PERMISSION When this flag is used with DRAG_FLAG_GLOBAL_URI_READ and/or DRAG_FLAG_GLOBAL_URI_WRITE , granting URI permissions can be persisted during device restarts until explicitly revoked with context.revokeUriPermission(Uri, int) Context.revokeUriPermission}. int DRAG_FLAG_GLOBAL_PREFIX_URI_PERMISSION When this flag is used with DRAG_FLAG_GLOBAL_URI_READ and/or DRAG_FLAG_GLOBAL_URI_WRITE, the URI permission applies to any URI that matches the prefix of the original granted URI. int DRAG_FLAG_GLOBAL_URI_READ this flag is used with DRAG_FLAG_GLOBAL, the drag recipient will be able to request read access to the URI content contained in the ClipData object. int DRAG_FLAG_GLOBAL_URI_WRITE this flag is used with DRAG_FLAG_GLOBAL, the drag recipient will be able to request write access to the contents of the URIs contained in the ClipData object. int DRAG_FLAG_OPAQUE flag indicating that the drag shadow will be opaque. int DRAWING_CACHE_QUALITY_AUTO this constant has been deprecated at API level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware layer, the intermediate cache is largely unnecessary and can easily cause loss of due to the cost of creating and updating the layer. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you Canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. int DRAWING_CACHE_QUALITY_HIGH this constant has been deprecated at API level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and

can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. int DRAWING_CACHE_QUALITY_LOW this constant has been deprecated at API level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. int FIND_VIEWS_WITH_CONTENT_DESCRIPTION Find views that contain description of the content. int FIND_VIEWS_WITH_TEXT find renderer views of specific text. int FOCUSABLE This view wants keystrokes. int FOCUSABLES_ALL View flag indicating whether addFocusables(java.util.ArrayList, int, int) should add all focusable views regardless of whether they are focusable in touch mode. int FOCUSABLES_TOUCH_MODE Display a flag indicating addFocusables(java.util.ArrayList, int, int) should only add focusable views in touch mode. int FOCUSABLE_AUTO This view automatically determines focus. int FOCUS_BACKWARD Use with focusSearch(int). int FOCUS_DOWN Use with focusSearch(int). int FOCUS_FORWARD Use with focusSearch(int). int FOCUS_LEFT Use with focusSearch(int). int FOCUS_RIGHT Use with focusSearch(int). int FOCUS_UP Use with focusSearch(int). int GONE This view is invisible and does not take up space for layout. int HAPTIC_FEEDBACK_ENABLED Display a flag indicating whether this view should have touch feedback enabled for events such as long presses. int IMPORTANT_FOR_ACCESSIBILITY_AUTO automatically determine whether the view is important for accessibility. int IMPORTANT_FOR_ACCESSIBILITY_NO view is not valid for availability. int IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS view is not valid for availability, nor does it have any child views. int IMPORTANT_FOR_ACCESSIBILITY_YES view is important for availability. int IMPORTANT_FOR_AUTOFILL_AUTO Automatically determines whether the view is important for autocomplete. int IMPORTANT_FOR_AUTOFILL_NO view is not valid for autocomplete, but its children (if any) will be moved. int IMPORTANT_FOR_AUTOFILL_NO_EXCLUDE_DESCENDANTS view is not valid for autocomplete, and its children (if any) will not be moved. int IMPORTANT_FOR_AUTOFILL_YES view is important for autocomplete, and its children (if any) will be moved. int IMPORTANT_FOR_AUTOFILL_YES_EXCLUDE_DESCENDANTS view is important for autocomplete, but its children (if any) will not be moved. int IMPORTANT_FOR_CONTENT_CAPTURE_AUTO automatically determine whether a view is important for capturing content. int IMPORTANT_FOR_CONTENT_CAPTURE_NO view is not valid for capturing content, but its children (if any) will be moved. int IMPORTANT_FOR_CONTENT_CAPTURE_NO_EXCLUDE_DESCENDANTS view is not valid for capturing content, and its children (if any) will not be moved. int IMPORTANT_FOR_CONTENT_CAPTURE_YES view is important for capturing content, and its children (if any) will be moved. int IMPORTANT_FOR_CONTENT_CAPTURE_YES_EXCLUDE_DESCENDANTS view is important for capturing content, but its children (if any) will not be moved. int INVISIBLE This view is invisible but still occupies space for layout purposes. int KEEP_SCREEN_ON View flag indicating that the screen should remain on while the window containing this view is visible to the user. int LAYER_TYPE_HARDWARE Indicates that the view has a hardware layer. int LAYER_TYPE_NONE Indicates that the view does not have a layer. int LAYER_TYPE_SOFTWARE indicates that the view has a software layer. int LAYOUT_DIRECTION_INHERIT The horizontal layout direction of this view is inherited from its int LAYOUT_DIRECTION_LOCALE the horizontal direction of the layout of this derived from the default language script for the locale. int LAYOUT_DIRECTION_LTR The horizontal layout direction of this view is left-to-right. int LAYOUT_DIRECTION_RTL The horizontal layout direction of this view is right-to-left. int MEASURED_HEIGHT_STATE_SHIFT Bit shift MEASURED_STATE_MASK to get to height bits for functions that combine both width and height in one int, such as getMeasuredState() and childState, resolveSizeAndState(int, int, int). int MEASURED_SIZE_MASK GetMeasuredWidthAndState() and getMeasuredWidthAndState() bits that provide the actual measured size. int MEASURED_STATE_MASK GetMeasuredWidthAndState() and getMeasuredWidthAndState() bits that provide additional state bits. int MEASURED_STATE_TOO_SMALL Bit getMeasuredWidthAndState() and getMeasuredWidthAndState(), which indicates that the measured size is smaller than the space you want the view to have. int NOT_FOCUSABLE this view does not want keystrokes. int NO_ID Used to mark a view that does not have an IDENTIFIER. int OVER_SCROLL_ALWAYS Always allow a user to scroll through this view, provided it is a scrollable view. int OVER_SCROLL_IF_CONTENT_SCROLLS Allow a user to scroll through this view only if the content is large enough to meaningfully scroll, provided that it is a scrollable view. int OVER_SCROLL_NEVER Never allow a user to scroll through this view. int SCREEN_STATE_OFF indicates that the screen has changed state and is now off. int SCREEN_STATE_ON indicates that the screen has changed state and is now on. int SCROLLBARS_INSIDE_INSET scroll bar style to display scroll bars inside the padded area, increasing the padding of the view. int SCROLLBARS_INSIDE_OVERLAY scroll bar style to display scroll bars inside the content area without increasing the padding. int SCROLLBARS_OUTSIDE_INSET scroll bar style to display scroll bars at the edge of the view, increasing the padding of the view. int SCROLLBARS_OUTSIDE_OVERLAY scroll bar style to display scroll bars at the edge of the view without increasing the padding. int SCROLLBAR_POSITION_DEFAULT place the scroll bar in the default position specified by the system. int SCROLLBAR_POSITION_LEFT place the scroll bar along the left edge. int SCROLLBAR_POSITION_RIGHT Place the scroll bar along the right edge. int SCROLL_AXIS_HORIZONTAL Indicates scrolling along the horizontal axis. int SCROLL_AXIS_NONE Indicates no scroll axis. int SCROLL_AXIS_VERTICAL Indicates scrolling along the vertical axis. int SCROLL_INDICATOR_BOTTOM to the bottom edge of the view. int SCROLL_INDICATOR_END Direction for ending the view edge. int SCROLL_INDICATOR_LEFT scroll direction of the left edge of the view. Int Scroll direction for the right edge of the view. Int Int Scroll the direction for the start edge of the view. int SCROLL_INDICATOR_TOP scroll direction for the top edge of the view. int SOUND_EFFECTS_ENABLED a view flag indicating whether this view should have sound effects enabled for events such as clicking and tapping. int STATUS_BAR_HIDDEN this constant has been deprecated at API level 15. Use a SYSTEM_UI_FLAG_LOW_PROFILE. int STATUS_BAR_VISIBLE this constant has been deprecated at API level 15. Use a SYSTEM_UI_FLAG_VISIBLE. int SYSTEM_UI_FLAG_FULLSCREEN this constant has been deprecated at api level 30. Instead, use the WindowInsetsController#hide(int) function with type#statusBars(). int SYSTEM_UI_FLAG_HIDE_NAVIGATION This constant has been deprecated at API level 30. Instead, use windowinsetscontroller#hide(int) with Type#navigationBars(). int SYSTEM_UI_FLAG_IMMERSIVE this constant has been deprecated at api level 30. Instead, use WindowInsetsController#BEHAVIOR_SHOW_BARS_BY_SWIPE. int SYSTEM_UI_FLAG_IMMERSIVE_STICKY this constant has been deprecated at api level 30. Instead, use WindowInsetsController#BEHAVIOR_SHOW_TRANSIENT_BARS_BY_SWIPE. int SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN this constant has been deprecated at api level 30. For Offset windows, use layoutparams#setFitInsetsTypes(int) with type#statusBars() ()}. In case of non-complying windows filled by the screen, call #SetDecorFitsSystemWindows (boolean) error with falsehood. int SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION this constant has been deprecated at api level 30. For Offset windows, use layoutparams#setFitInsetsTypes(int) with type#navigationBars(). In case of non-complying windows filled by the screen, call #SetDecorFitsSystemWindows (boolean) error with falsehood. int SYSTEM_UI_FLAG_LAYOUT_STABLE This constant has been deprecated at api level 30. Instead, use windowinsets#getInsetsIgnoringVisibility(int) to retrieve snippets that do not change when the system bars change visibility state. int SYSTEM_UI_FLAG_LIGHT_NAVIGATION_BAR This constant has been deprecated at API level 30. Instead, use WindowInsetsController#APPEARANCE_LIGHT_NAVIGATION_BARS. int SYSTEM_UI_FLAG_LIGHT_STATUS_BAR this constant has been deprecated at api level 30. Instead, use WindowInsetsController#APPEARANCE_LIGHT_STATUS_BARS. int SYSTEM_UI_FLAG_LOW_PROFILE this constant has been deprecated at api level 30. Low profile mode is obsolete. Instead, hide the system bars if the application must be silent. Use windowInsetsController#hide(int) with Type#systemBars(). int SYSTEM_UI_FLAG_VISIBLE This constant has been deprecated at the INTERFACE level 30. SystemUiVisibility flags are obsolete. Instead, use the WindowInsetsController. int SYSTEM_UI_LAYOUT_FLAGS this constant has been deprecated at api level 30. System UI layout flags are obsolete. int TEXT_ALIGNMENT_CENTER center a paragraph, such as a ALIGN_CENTER. int TEXT_ALIGNMENT_GRAVITY default for the main view. int TEXT_ALIGNMENT_INHERIT Default text alignment. int TEXT_ALIGNMENT_TEXT_END Align to for example, ALIGN_OPPOSITE. int TEXT_ALIGNMENT_TEXT_START the beginning of a paragraph, such as ALIGN_NORMAL. int TEXT_ALIGNMENT_VIEW_END align to the end of the view, which is ALIGN_RIGHT if recognized in layoutDirection view is LTR and ALIGN_LEFT or otherwise. int TEXT_ALIGNMENT_VIEW_START Align to the beginning of the view, which ALIGN_LEFT if recognized in layoutDirection view is LTR ALIGN_RIGHT otherwise. int TEXT_DIRECTION_ANY_RTL Text Direction uses any-RTL. int TEXT_DIRECTION_FIRST_STRONG Text Direction uses the first strong algorithm. int TEXT_DIRECTION_FIRST_STRONG_LTR Text Direction uses the first strong algorithm. int TEXT_DIRECTION_FIRST_STRONG_RTL Text Direction uses the first strong algorithm. int TEXT_DIRECTION_INHERIT Text Direction is inherited by ViewGroup int TEXT_DIRECTION_LOCALE text direction comes from the system locale. int TEXT_DIRECTION_LTR text direction is forced to LTR. int TEXT_DIRECTION_RTL text direction is forced to RTL. The VIEW_LOG_TAG the logging tag used by this class from android.util.Log. int VISIBLE This view is visible. From the public android.view.View class, the static properties of the final<View, Float> ALPHA wrapper properties around the alpha function supported by the View#setAlpha(float) and View#getAlpha() methods. protected static final int[] EMPTY_STATE_SET Indicates that the view does not have a ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET ENABLED_FOCUSED_SELECTED_STATE_SET. selected and its window has focus. Protected static final int[] ENABLED_FOCUSED_STATE_SET Indicates that the view is enabled and has focus. protected static final int[] ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET Indicates that the view is enabled, concentrated, and its window has ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET ENABLED_SELECTED_STATE_SET. , and its window has focus. protected static trailing int[] ENABLED_STATE_SET Indicates that the view is enabled. Protected static trailing int[] ENABLED_WINDOW_FOCUSED_STATE_SET Indicates that the view is enabled and that its window has focus. Protected static trailing int[] FOCUSED_SELECTED_STATE_SET Indicates that the view is concentrated and selected. Protected Static Trailing Int[] FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET Indicates that the view is concentrated, selected, and its window has focus. protected static trailing int[] FOCUSED_STATE_SET Indicates that the view is concentrated. Protected Static Trailing Int[] FOCUSED_WINDOW_FOCUSED_STATE_SET Indicates that the view has focus and that its window has focus. protected static trailing int[] PRESSED_ENABLED_FOCUSED_SELECTED_STATE_SET </View, Float>the view is pressed, turned on, concentrated, and selected. Protected Static Trailing Int[] PRESSED_ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET Indicates that the view is pressed, turned on, concentrated, selected, and its window has focus. Protected static trailing int[] PRESSED_ENABLED_FOCUSED_STATE_SET Indicates that the view is pressed, enabled, and concentrated. Protected Static Trailing Int[] PRESSED_ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET Indicates that the view is pressed, turned on, concentrated, and its window has focus. Protected static trailing int[] PRESSED_ENABLED_SELECTED_STATE_SET Indicates that the view is pressed, enabled, and selected. Protected Static Trailing Int[] PRESSED_ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET Indicates that the view is pressed, turned on, selected, and its window has focus. Protected static trailing int[] PRESSED_ENABLED_STATE_SET Indicates that the view is pressed and enabled. Protected static trailing int[] PRESSED_ENABLED_WINDOW_FOCUSED_STATE_SET Indicates that the view is pressed, turned on, and its window has focus. Protected static trailing int[] PRESSED_FOCUSED_SELECTED_STATE_SET Indicates that the view is pressed, concentrated, and selected. Protected static trailing int[] PRESSED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET Indicates that the view is pressed, concentrated, selected, and its window has sharpness. Protected static trailing int[] PRESSED_FOCUSED_STATE_SET Indicates that the view is pressed and concentrated. Protected static trailing int[] PRESSED_FOCUSED_WINDOW_FOCUSED_STATE_SET Indicates that the view is pressed, concentrated, and its window has sharpness. Protected static trailing int[] PRESSED_SELECTED_STATE_SET Indicates that the view is pressed and selected. Protected static trailing int[] PRESSED_SELECTED_WINDOW_FOCUSED_STATE_SET Indicates that the view is pressed, selected, and its window has focus. Protected static trailing int[] PRESSED_STATE_SET Indicates that the view is pressed. Protected static trailing int[] PRESSED_WINDOW_FOCUSED_STATE_SET Indicates that the view is pressed and its window has focus. public static final Property<View, Float> ROTATION Wrapper properties around rotation functions supported by the view#setRotation(float) and View#getRotation() methods. public static final Property<View, Float> ROTATION_X A Property wrapper around the rotationX functionality handled by the View#setRotationX(float) and View#getRotationX() methods. public static final Property<View, Float> ROTATION_Y A Property wrapper around the rotationY functionality handled by the View#setRotationY(float) and View#getRotationY() methods. public static final Property<View, Float> SCALE_X A Property wrapper around the scaleX functionality handled by the View#setScaleX(float) and View#getScaleX() methods. public static final Property<View, Float> SCALE_Y A Property wrapper around the functionality handled by the View#setScaleY(float) and View#getScaleY() methods. protected static final int[SELECTED_STATE_SET]</View, Float> </View, Float> </View, Float> </View, Float> </View, Float> Float> zostanie wybrany widok. chronione statyczne końcowe int[] SELECTED_WINDOW_FOCUSED_STATE_SET Wskazuje, że widok jest zaznaczony i że jego okno ma fokus. public static final Property<View, Float> TRANSLATION_X A Property wrapper around the translationX functionality handled by the View#setTranslationX(float) and View#getTranslationX() methods. public static final Property<View, Float> TRANSLATION_Y A Property wrapper around the translationY functionality handled by the View#setTranslationY(float) and View#getTranslationY() methods. public static final Property<View, Float> TRANSLATION_Z A Property wrapper around the translationZ functionality handled by the View#setTranslationZ(float) and View#getTranslationZ() methods. protected static final int[] WINDOW_FOCUSED_STATE_SET Indicates the view's window has focus. public static final Property<View, Float> X A Property wrapper around the x functionality handled by the View#setX(float) and View#getX() methods. public static final Property<View, Float> Y A wrapper around the y functionality handled by the View#setY(float) and View#getY() methods. public static final Property<View, Float> Z A Property wrapper around the z functionality handled by the View#setZ(float) and View#getZ() methods. public static final Property wrapper around the z functionality handled by the View#setZ(float) and View#getZ() methods. public static final Property wrapper around the z functionality handled by the View#setZ(float) and View#getY() methods. public static final Property wrapper around the z functionality handled by the View#setZ(float) and View#getZ() methods. public static final Property wrapper around the z functionality handled by the View#setZ(float) and View#getZ() methods. public static final Property wrapper around the z functionality handled by the View#setZ(float) and View WebView(Kontekst kontekstu) Konstruuje nowy webview z obiektu kontekstu działania. WebView(kontekst kontekstowy, AtrybutSet attrs) Konstruuje nowy webview z parametrami układu. WebView(Kontekst kontekstu, AttributeSet attrs, int defStyleAttr) Konstruuje nowy WebView z parametrami układu i stylem domyślnym. WebView(Kontekst kontekstu, AttributeSet attrs, int defStyleAttr, int defStyleRes) Konstruuje nowy webview z parametrami układu i stylem domyślnym. WebView(kontekst kontekstu, AttributeSet attrs, int defStyleAttr, boolean privateBrowsing) Ten konstruktor jest przestarzały. Przeglądanie prywatne nie jest już obsługiwane bezpośrednio za pośrednictwem przeglądarki WebView i zostanie usunięte w przyszłej wersji. Preferuj używanie WebSettings, WebViewDatabase, and WebStorage for detailed control of your privacy data. void addJavascriptInterface(Object, String name) Injects the supplied Java object into this webview. void autofill(SparseArray Values)<AutofillValue> Automatically fills the contents of virtual child damage in this view. boolean canGoBack() Gets whether this WebView has a back history item. the boolean value canGoBackOrForward(int steps) Gets whether the page can return or forward the specified number of steps. the boolean value canGoForward() gets whether this WebView has a forward history element. This method is prone to inaccuracies due to race conditions between web rendering and UI threads; prefer WebViewClient#onScaleChanged. boolean canZoomOut() This method has been deprecated in api level 17. This method is prone to inaccuracy due to race conditions between rendering</AutofillValue> </View, Float> </View, Float> </View, Float> </View, Float> </View, Float> </View, Float> Float> threads; prefer WebViewClient#onScaleChanged. Picture capturePicture() This method has been deprecated in api level 19. Use onDraw(Canvas) to get a snapshot of the webview bitmap, or saveWebArchive(String) to save the content to a file. void clearCache(boolean includeDiskFiles) Clears the resource cache. static void clearClientCertPreferences(Runnable onCleared) Clears client certificate preferences stored in response to client certificate requests. void clearFormData() Removes the AutoComplete pop-up from the currently concentrated form field, if present. void clearHistory() Tells this webview to clear its internal back/forward list. void clearMatches() Clears the distinctive surrounding text matches created by findAllAsync(String). void clearSslPreferences() Clears the SSL preference table stored in response to SSL certificate errors. void clearView() This method has been deprecated in api level 18. Use WebView.loadUrl(about:blank) to reliably reset the view state and free up page resources (including all javascript running). void computeScroll() Called by the parent to request that the child update its values for mScrollX and mScrollY if necessary. WebBackForwardList copyBackForwardList() Gets WebBackForwardList for this WebView. PrintDocumentAdapter createPrintDocumentAdapter(String documentName) Creates a PrintDocumentAdapter that provides the content of this webview for printing. PrintDocumentAdapter createPrintDocumentAdapter() This method has been deprecated in api level 21. Use createPrintDocumentAdapter(java.lang.String), which requires the user to provide the name of the print document. WebMessagePort[] createWebMessageChannel() Creates a message channel to communicate with JS and returns message ports that represent the endpoints of that message channel. void destroy() Destroys the internal state of this webview. static void disableWebView() Indicates that the current process has no intention of using WebView and that an exception should be thrown if a webview is created or other methods are used in the android.webkit package. boolean dispatchKeyEvent(KeyEvent event) Send the key event to the next view on the focus path. void documentHasImages(Message response) Responds to the document to see if it contains references to the image. static void enableSlowWholeDocumentDraw() For L-version applications, WebView has a new default behavior that reduces memory consumption and improves performance by intelligently selecting the part of the HTML document that needs to be drawn. void evaluateJavascript(String script, ValueCallback<String> resultCallback) Asynchronously evaluates JavaScript in the context of the currently displayed page. static String findAddress(String addr) Ta has been deprecated at API level 28. This method is replaced by TextClassifier#generateLinks(android.view.textclassifier.TextLinks.Request). Avoid </String> </String> this method, even when targeting API levels where no alternative is available. int findAll(String find) This method has been deprecated at api level 16. findAllAsync(String) is preferred. void findAllAsync(String find) Searches for all instances found on the page and highlights them asynchronously. View findFocus() Find a view in a hierarchy rooted in that view that currently has focus. void findNext(boolean forward) Highlights and scrolls to the next match found by findAllAsync(String), wrapping around page boundaries as necessary. void flingScroll(int vx, int vy) void freeMemory() This method has been deprecated at API level 19. Caches are automatically discarded when they are no longer needed, and in response to system memory pressure. CharSequence getAccessibilityClassName() Returns the class name of this object to be used for accessibility purposes. AccessibilityNodeProvider getAccessibilityNodeProvider() Gets the provider to manage the virtual view hierarchy rooted in that view and reported to AccessibilityServices that explore the contents of the window. SslCertificate getCertificate() Gets an SSL certificate for the top-level or null main page if it does not have a certificate (the site is not secure). int getContentHeight() Gets the height of the HTML content. static PackageInfo getCurrentWebViewPackage() If the WebView has already been loaded into the current process, this method will return the package that was used to load it. The getFavicon() bitmap gets a favicon for the current page. Handler getHandler() WebView.HitTestResult getHitTestResult() Gets the HitTestResult result from the current cursor node. String[] getHttpAuthUsernamePassword(String host, String realm) This method has been deprecated at API level 26. Use WebViewDatabase#getHttpAuthUsernamePassword instead of String getOriginalUrl() Gets the original URL for the current page. int getProgress() Gets progress for the current page. boolean getRendererPriorityWaivedWhenNotVisible() Returns whether this webview requests RENDERER_PRIORITY_WAIVED when not visible. int getRendererRequestedPriority() Get the desired renderer priority for this webview. Uri getSafeBrowsingPrivacyPolicyUrl() Returns a URL that indicates a privacy policy for safe browsing reporting. float getScale() This method has been deprecated in api level 17. This method is prone to inaccuracies due to race conditions between web rendering and UI threads; prefer WebViewClient#onScaleChanged. WebSettings getSettings() Gets the WebSettings object used to control the settings for this WebView. TextClassifier getTextClassifier() Returns the textclassifier used by this webview. getTitle() Gets for the current page. GetUrl() Gets the URL for the current page. WebChromeClient getWebChromeClient() Gets chrome support. static ClassLoader getWebViewClassLoader() Returns ClassLoader used to load internal WebView WebView WebViewClient getWebViewClient() Gets WebViewClient. Looper getWebViewLooper() Returns looper corresponding to the thread on which webview calls must be made. WebViewRenderProcess getWebViewRenderProcess() Gets the access to the WebView rendering process associated with this WebView. WebViewRenderProcessClient getWebViewRenderProcessClient() Gets the rendering client object associated with this WebView. void goBack() Goes back to the history of this WebView. void goBackOrForward(int steps) Moves to the history element, which is the number of steps from the current element. void goForward() Goes forward in the history of this WebView. void invokeZoomPicker() Calls the graphical zoom selector widget for this WebView. boolean isPrivateBrowsingEnabled() Gets whether private browsing is enabled in this WebView. boolean isVisibleToUserForAutofill(int virtualId) Calculates whether this virtual autocomplete view is visible to the user. void loadData(String mimeType, String encoding) Loads the given data into this webview using the data schema URL. void loadDataWithBaseURL(String baseUrl, String data, String mimeType, String encoding, String historyUrl) Loads the given data into this WebView, using baseUrl as the primary content URL. void loadUrl(String url) Loads the specified URL. void loadUrl(String url, Map<String, String> additionalHttpHeaders) Loads a given URL with specific additional HTTP headers. boolean onCheckIsTextEditor() Verify that the view you are calling is a text editor, in which case it is worth automatically displaying a soft input window for it. void onChildViewAdded This method is deprecated. WebView no longer needs to implement ViewGroup.OnHierarchyChangeListener. This method does nothing now. void onChildViewRemoved(See p, View child) This method is obsolete. WebView no longer needs to implement ViewGroup.OnHierarchyChangeListener. This method does nothing now. InputConnection onCreateInputConnection(EditorInfo outAttrs) Creates a new inputconnection for InputMethod to interact with WebView. boolean onDragEvent(DragEvent event) Handles drags events sent by the system when startDragAndDrop() is called. void onFinishTemporaryDetach() Called after onStartTemporaryDetach() after the container is finished by changing the view. boolean onGenericMotionEvent Implement this method to handle general motion events. void onGlobalFocusChanged(See oldFocus, See newFocus) This method is obsolete. WebView should not have implemented ViewTreeObserver.OnGlobalFocusChangeListener. This method does nothing now. boolean onHoverEvent Implement this method for handling activation events. boolean onKeyDown(int keyCode, KeyEvent event) Default implementation KeyEvent): Press the view when KeyEvent#KEYCODE_DPAD_CENTER or </String, String> String> will be released if the view is enabled and clickable. boolean onKeyMultiple (int keyCode, int repeatCount, KeyEvent event) Default implementation of KeyEvent.Callback#onKeyMultiple(int, int, KeyEvent): always returns false (does not handle events). boolean onKeyUp(int keyCode, KeyEvent event) Default implementation KeyEvent.Callback#onKeyUp(int, KeyEvent): Click the view when you release KeyEvent#KEYCODE_DPAD_CENTER, KeyEvent#KEYCODE_ENTER, or KeyEvent#KEYCODE_SPACE. void onPause() Attempts to pause processing that can be safely paused, such as animations and geolocation. void onProvideAutofillVirtualStructure(ViewStructure structure, int flags) Populates the view structure containing virtual children to make a fullfil autocomplete request. ViewStructure traditionally represents a view, while for web pages it represents HTML nodes. void onProvideContentCaptureStructure(ViewStructure structure, int flags) fills the view structure for capturing content. void onProvideVirtualStructure(ViewStructure structure) Called when a help structure is retrieved from a view as part of activity.onProvideAssistData to generate an additional virtual structure in that view. void onResume() resumes webview after previous call onPause(). void onStartTemporaryDetach() This is called when the container has to temporarily disconnect the child from ViewGroup#detachViewFromParent(View). boolean onTouchEvent Implement this method to handle touch screen motion events. boolean onTrackballEvent Implement this method to handle trackball motion events. void onWindowFocusChanged(boolean hasWindowFocus) Called when a window containing this view gains or loses focus. Logical layerHorizontalScrollbar() This method has been deprecated in api level 23. This method is now obsolete. Logical LayerVerticalScrollbar() This method has been deprecated in api level 23. This method is now obsolete. boolean pageDown(boolean bottom) Scrolls the contents of this WebView by half the page size. boolean pageUp(boolean top) Scrolls the contents of this WebView by half the size of the view. void pauseTimers() Pauses all layout, parsing, and javascript timers for all webviews. boolean performLongClick() Calls onlongclicklistener this view if it is defined. void postUrl(String url, byte[] postData) Loads the URL from postData using the POST method to this WebView. void postVisualStateCallback(long requestId, WebView.VisualStateCallback callback) VisualStateCallback posts, which will be called when the current WebView state is ready to be drawn. void postWebMessage(WebMessage message, Uri targetOrigin) Publish the message to the main frame. void reload() Reloads the current URL. Void name) Removes a previously injected Java object from this webview. logical requestChildRectangleOnScreen(Logical ViewChildRectangleOnScreen(View Rect rect, boolean immediate) Triggered when a child of this group wants a specific rectangle to be placed on the screen. boolean requestFocus(int direction, Rect previouslyFocusedRect) Call this to try to focus on a specific view or to one of its children and give it hints about the direction and specific rectangle that the focus comes from. Searches for a view to focus on following the setting specified by getDescendantFocusability(). void requestFocusNodeHref(Message hrefMsg) Requests the URL of the anchor element or image element at the last tap point. void requestImageRef(Message msg) Requests the URL of the image last touched by the user. WebBackForwardList restoreState(Bundle inState) Restores the state of this webview from a given package. void resumeTimers() Resumes all layout, analysis, and JavaScript timers for all webviews. void savePassword(String host, String username, String password) This method has been deprecated at API level 18. Saving passwords in WebView will not be supported in future releases. WebBackForwardList saveState(Bundle outState) Saves the state of this webview used in Activity.onSaveInstanceState(Bundle). void saveWebArchive(String filename) Saves the current view as an internet archive. void saveWebArchive(String basename, boolean autoname, ValueCallback<String> callback) Saves the current view as an internet archive. void setBackgroundColor(int color) Sets the background color for this view. void setCertificate (SslCertificate certificate) This method is deprecated in API 17. Calling this function has no useful effect and will be ignored in future releases. static void setDataDirectorySuffix(String suffix) Define the directory used to store WebView data for the current process. void setDownloadListener(DownloadListener listener) Registers the interface to be used when the content cannot be supported by the rendering engine and should be downloaded. void setFindListener(WebView.FindListener listener) Registers the listener to receive notifications as the find-on-page operation progresses. void setHorizontalScrollbarOverlay (logical overlay) This method has been deprecated in api level 23. This method has no effect. void setHttpAuthUsernamePassword(String host, string sphere, string user name, string password) This method has been deprecated at API level 26. Use webviewdatabase#setHttpAuthUsernamePassword instead of void setInitialScale(int scaleInPercent) sets the initial scale for this webview. void setLayerType(int layerType, Paint paint) Specifies the type of layer that backs up this view. void setLayoutParams(ViewGroup.LayoutParams params) Set the layout parameters associated with this view. void setMapTrackballToArrowKeys This method has been deprecated at API level 17. Only the default case, true, will be supported in a future release. void setNetworkAvailable(boolean networkUp) Informs WebView of the network status. invalid </String> </String> scroll mode for this view. void setPictureListener(WebView.PictureListener listener) This method has been deprecated in api level 15. This method is now obsolete. void setRendererPriorityPolicy(int rendererRequestedPriority, boolean waivedWhenNotVisible) Set rendering priority policies for this WebView. Static void setSafeBrowsingWhitelist(List<String> hosts, ValueCallback<Boolean> callback) Sets the list of hosts (domain names/IP addresses) that are exempt from SafeBrowsing control. void setScrollBarStyle(int style) Specify the style of the scroll bars. void setTextClassifier(TextClassifier textClassifier) Sets the textclassifier for this WebView. void setVerticalScrollbarOver(logical overlay) This method has been deprecated at API level 23. This method has no effect. void setWebChromeClient Sets the chrome handler. Static void setWebContentsDebuggingEnabled(boolean enabled) Allows you to debug Web content (HTML/CSS/JavaScript) loaded into any WebViews of this application. void setWebViewClient(WebViewClient client) Sets a WebViewClient that will receive various notifications and requests. void setWebViewRenderProcessClient(executor executor, WebViewRenderProcessClient webViewRenderProcessClient) Sets the render client object associated with this WebView. void setWebViewRenderProcessClient(WebViewRenderProcessClient webViewRenderProcessClient) Sets the rendering client object associated with this webview. boolean shouldDelayChildPressedState() Returns true if the press state should be delayed for children or children of this viewgroup. boolean showFindDialog(String text, boolean showIme) This method has been deprecated in api level 18. This method does not work reliably on all versions of Android; implementing a custom find dialog box using WebView.findAllAsync() provides a more reliable solution. static void startSafeBrowsing(context context, ValueCallback<Boolean> callback) Initiates safe browsing. void stopLoading() Stops the current load. void zoomBy (float zoomFactor) Performs a zoom operation in this WebView. zoom() zooms in on this WebView. boolean zoomOut() Performs zoom out in this webview. int computeHorizontalScrollRange() Calculate a horizontal range that represents a horizontal scroll bar. int computeVerticalScrollExtent() Calculate the vertical vertical range of the thumb scroll bar in the vertical range. vertical move of the vertical thumb scroll bar within the horizontal range. int computeVerticalScrollRange() Calculate the vertical range that represents the vertical scroll bar. void dispatchDraw(Canvas canvas) Triggered by widoków podrzędnych. void onAttachedToWindow() Jest to tak się nazywa</Boolean> </Boolean> </String> </String> the view is attached to the window. void onConfigurationChanged(Configuration newConfig) Called when the current configuration of resources used by the application has changed. void onDraw(Canvas canvas) Implement this to make a drawing. void onFocusChanged(boolean focused, int direction, Rect previouslyFocusedRect) Called by the view system when the focus state of that view changes. void onMeasure (int widthMeasureSpec, int heightMeasureSpec) Measure the view and its contents to determine the measured width and measured height. void onOverScrolled (int scrollX, int scrollY, boolean clampedX, boolean clampedY) Triggered by overScrollBy (int, int, int, int, int, int, int, int, boolean) in response to scrolling results. void onScrollChanged (int l, int t, int oldl, int oldt) This is invoked in response to an internal scroll in this view (i.e. the view scrolled its own content). void onSizeChanged (int w, int h, int ow, int oh) This is called during the layout when the size of this view has changed. void onVisibilityChanged(View changedView, int visibility) Called when view visibility or view parent changes. void onWindowVisibilityChanged(int visibility) Called when the containing window has changed its visibility (between GONE, INVISIBLE, and VISIBLE). From the class android.view.ViewGroup void addChildrenForAccessibility(ArrayList<View> outChildren) Adds bundles of this view relevant for availability to a given list as output. void addExtraDataToAccessibilityNodeInfo(AccessibilityNodeInfo info, String extraDataKey, Bundle arguments) Adds additional data to AccessibilityNodeInfo based on an explicit request for additional data. void addFocusables(Array Views<View> int focusableMode) Adds all focusable views that are children of that view (possibly in this view if it is self-centered) to the views. void addKeyboardNavigationClusters(Views<View> Collection, Int Direction) Adds any keyboard navigation cluster roots that are children of this view (possibly in this view if it is the cluster root itself) to the views. boolean addStatesFromChildren() Returns whether it is the drawing states of the viewgroup group that also include drawing states for her children. void addTouchables(ArrayList<View> views) Add all touch views that are children of this view (possibly in this view, if it is touchable) to the views. void addView, ViewGroup.LayoutParams params) Adds a child view with specific layout parameters. void addView Adds a child view. void addView(View Child, int index, ViewGroup.LayoutParams params) Adds a child view with specific layout parameters. void addView Adds a child view. void addView int, height int) Adds a sub view with default viewgroup layout parameters and specified width</View> </View> </View> </View> </View> Height. boolean addViewInLayout(View Child, int index, ViewGroup.LayoutParams params, boolean preventRequestLayout) Adds a view during layout. boolean addViewInLayout(View Child, int index, ViewGroup.LayoutParams params) Adds a view during layout. void attachLayoutAnimationParameters(View child, ViewGroup.LayoutParams params, int index, int count) Subclasses should override this method to set layout animation parameters for the delivered child. void attachViewToParent(View Child, int index, ViewGroup.LayoutParams params) Appends the view to this view group. void bringChildToFront(See child) Change the order of the child so that it is on top of all other girls. boolean canAnimate() Indicates whether a group of views has the ability to animate their minor damage after the first layout. boolean checkLayoutParams(ViewGroup.LayoutParams p) void childDrawableStateChanged(View child) If addStatesFromChildren() is true, it refreshes the drawable state of this group (to include states with its underlying injuries). void childHasTransientStateChanged(See child, boolean childHasTransientState) Called when the child view has changed, whether it is tracking a transient state. void cleanupLayoutState(View child) Prevents a specific child element to be deployed during the next layout run. void clearChildFocus(View child) Called when a child of that parent opts out of void focus clearDisappearingChildren() Removes any pending animations for views that have been deleted. void clearFocus() Called when this view wants to give up focus. void debug(int depth) void detachAllViewsFromParent() Detaches all views from the parent. void detachViewFromParent(int index) Detaches the view from its parent. void detachViewFromParent Detachs the view from its parent. void detachViewsFromParent(int start, int count) Detaches the view range from their parents. WindowInsets dispatchApplyWindowInsets(WindowInsets insets) Request that window snippets be applied to that view or view in a sub tree. boolean dispatchCapturedPointerEvent(MotionEvent event) Pass the captured pointer event down to the focused view. void dispatchConfigurationChanged(Configuration newConfig) Send a notification that the resource configuration has changed in the view hierarchy. void dispatchDisplayHint(int hint) Send a hint as to whether this view is displayed. boolean dispatchDragEvent(DragEvent event) Detects whether this view is enabled and has a drag event listener. void dispatchDraw (Canvas canvas) Called by drawing to draw child views. void dispatchDrawableHotspotChanged(float x, float y) Dispatches drawable hotspot changes to child views that meet one or more of the following criteria: void container) Perform view.saveHierarchyState(android.util.SparseArray) freeze()} only for this view and not for its children. boolean </Parcelable> </Parcelable> event) Send the general traffic event to the currently concentrated view. boolean dispatchGenericPointerEvent(MotionEvent event) Send a general motion event to the view below the first pointer. boolean dispatchHoverEvent(MotionEvent event) Send activation event. boolean dispatchKeyEvent(KeyEvent event) Send the key event to the next view on the focus path. boolean dispatchKeyEventPreIme(KeyEvent event) Dispatch a key event before processing it with any input method associated with the view hierarchy. boolean dispatchKeyShortcutEvent(KeyEvent event) Raises a key hash event. void dispatchPointerCaptureChanged(boolean hasCapture) void dispatchProvideAutofillStructure(ViewStructure structure, int flags) Creates view structures for automatic population down the hierarchy when an assist structure is created as part of an autofill request. This implementation adds a view group to all child views, in addition to calling the default view implementation. void dispatchProvideStructure(ViewStructure structure) Dispatches the creation of a view structure down the hierarchy. void dispatchRestoreInstanceState(SparseArray<Parcelable> container) Called by restoreHierarchyState(android.util.SparseArray) to retrieve status for this view and its manual. void dispatchSaveInstanceState(SparseArray<Parcelable> container) Called by saveHierarchyState(android.util.SparseArray) to store status for this view and its children. void dispatchSetActivated(boolean activated) Dispatch setActivated to all of this View's children. void dispatchSetPressed(boolean pressed) Dispatch setPressed to all of this View's children. void dispatchSetSelected(boolean selected) Dispatch setSelected to all of this View's children. void dispatchSystemUiVisibilityChanged(int visible) This method is obsolete. Use WindowInsets#isVisible(int) to learn more about the display capabilities of the system bar by setting onapplyWindowInsetsListener in this view. void dispatchThawSelfOnly(SparseArray<Parcelable> container) Ship View.restoreHierarchyState(android.util.SparseArray) only to this view, not to your children. boolean dispatchTouchEvent(MotionEvent ev) Forward the touch screen movement event down to the target view or that view if it is the target. boolean dispatchTrackballEvent(MotionEvent event) Forward the trackball movement event down to focused view. direction int) This method is the last chance for the focused view and its ancestors to respond to the arrow key. void dispatchVisibilityChanged(View changedView, int visibility) Dispatch the view visibility changes down the view hierarchy. void dispatchWindowFocusChanged(boolean hasFocus) Triggered when a window containing this view gains or loses window focus. Void animationes) Dispatches</Parcelable> </Parcelable> </Parcelable> </Parcelable> </Parcelable> when the window snippe animation is complete. void dispatchWindowInsetsAnimationPrepare(WindowInsetsAnimation animation) Dispatches WindowInsetsAnimation.Callback#onPrepare(WindowInsetsAnimation) when Window Insets animation is being prepared. WindowInsets dispatchWindowInsetsAnimationProgress(WindowInsets insets, List<WindowInsetsAnimation> runningAnimations) Dispatches WindowInsetsAnimation.Callback#onProgress(WindowInsets, List) when Window Insets animation makes progress. WindowInsetsAnimation Animation(WindowInsetsAnimation, WindowInsetsAnimation.Bounds bounds) Dispatches WindowInsetsAnimation.Callback#onStart(WindowInsetsAnimation, Bounds) when Window Insets animation is started. void dispatchWindowSystemUiVisiblityChanged(int visible) This method is deprecated. SystemUiVisibility flags are obsolete. Instead, use the WindowInsetsController. void dispatchWindowVisibilityChanged(int visibility) Dispatch window visibility changes the view hierarchy. logical drawingChild(canvas, sub view, long drawingTime) Draw one child of this view group. void drawableStateChanged() This function is called each time the state of the view changes in such a way that it affects the state of the drawables displayed. void endViewTransition(View view) This method should always be called after an earlier call to startViewTransition(android.view.View). View findFocus() Find a view in a hierarchy rooted in that view that currently has focus. void findViewsWithText(ArrayList<View> outViews, CharSequence text, int flags) Finds views that contain the text. View focusSearch(See focused, int direction) Find the nearest view in a specific direction that you want to focus on. void focusableViewAvailable(View v) Informs the parent that a new focusable view has become available. logical gatherTransparentRegion(Region) ViewGroup.LayoutParams generateDefaultLayoutParams() Returns a set of default layout parameters. ViewGroup.LayoutParams generateLayoutParams(AttributeSet attrs) Returns a new set of layout parameters based on the provided set of attributes. ViewGroup.LayoutParams generateLayoutParams(ViewGroup.LayoutParams p) Returns a secure set of layout parameters based on the provided layout params. CharSequence getAccessibilityClassName() Returns the class name of this object to be used for accessibility purposes. The getChildAt(int index) view returns the view at the specified position in the group. int getChildCount() Returns the number of children in the group. int getChildDrawingOrder(int childCount, int drawingPosition) Converts the position of the drawing order to the position of the container. final int getChildDrawingOrder(int drawingPosition) Converts the position of the drawing order to the position of the container. static int getChildMeasureSpec (int spec, int padding, int childDimension) Is hard measureChildren: dowiedzieć się MeasureSpec do</View> </WindowInsetsAnimation> </WindowInsetsAnimation> specific child. boolean getChildStaticTransformation(See Child, T Transformation) Sets t as the child's static transformation, if set, returning a logical value to indicate whether the static transformation has been set. boolean getChildVisibleRect(View Child, Rect r, Point Offset) Calculate the visibility of part of a rectangular region defined in terms of child view coordinates. boolean getClipChildren() Returns the value of whether children of this group are clipped to the boundaries before drawing. boolean getClipToPadding() Returns the value of whether this lookout group will pull its child objects to padding and resize (but not attract) any edgeeffect to the padded region if padding is present. int getDescendantFocusability() Gets the child fox of this view group. The getFocusedChild() view returns a focused child of that view, if any. LayoutAnimationController getLayoutAnimation() Returns the layout animation controller used to animate the group's whisk. Animation.AnimationListener getLayoutAnimationListener() Returns the animation listener to which layout animation events are sent. int getLayoutMode() Returns the alignment base during a layout operation in this viewgroup: LAYOUT_MODE_CLIP_BOUNDS or LAYOUT_MODE_OPTICAL_BOUNDS. LayoutTransition getLayoutTransition() Gets the LayoutTransition object for this viewgroup. int getNestedScrollAxes() Returns the current nested scroll axes for this view group. ViewGroupOverlay getOverlay() Returns viewgroupOverlay for this view group, creating it if it doesn't already exist. int getPersistentDrawingCache() This method has been deprecated in api level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate

cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, such as alpha animations, View.setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call view.draw(android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. boolean getTouchscreenBlocksFocus() whether this viewgroup should ignore focus requests for itself and your injuries. boolean hasFocus() Returns true if this view has or contains focus maTransientState() Indicates whether the view is currently tracking a transient state that the application should not need savings and restoration, but this framework should be taken into account in particular in order to preserve where possible. int indexOfChild(View child) Returns an entry in a specific child view group. final void voidChild(View Child, Rect dirty) This method is deprecated. Instead, use onDescendantInvalidated(android.view.View, android.view.View) to observe updates to draw state in child elements. ViewParent invalidateChildInParent(int[] location, Rect dirty) This method is obsolete. Instead, use onDescendantInvalidated(android.view.View, android.view.View) to observe updates to draw state in child elements. logical value isAlwaysDrawnWithCacheEnabled() This method has been deprecated at API level 23. From Build.VERSION_CODES. M, this property is ignored. Children's views may no longer be disabled by parents for their caching behavior. logical value isAnimationCacheEnabled() This method has been deprecated in api level 23. From Build.VERSION_CODES. M, this property is ignored. The 19th caching behavior can be controlled using View#setLayerType(int, Paint). boolean isChildrenDrawingOrderEnabled() Indicates whether the viewgroup draws its children in the order defined by getChildDrawingOrder(int, int). boolean isChildrenDrawnWithCacheEnabled() This method has been deprecated in api level 23. From Build.VERSION_CODES. M, this property is ignored. Child views can no longer be forced to cache their rendering state by parents. Instead, use view#setLayerType(int, Paint) in each view. boolean isLayoutSuppressed() Returns whether layout calls in this container are currently being suppressed due to an earlier call to suppressLayout(boolean). the boolean value isMotionEventSplittingEnabled() returns true if motionevents sent to this viewgroup can be divided into multiple child injuries. the boolean isTransitionGroup() returns true if this viewgroup should be considered a single entity to delete when performing an activity transition. void jumpDrawablesToCurrentState() Call Drawable#jumpToCurrentState() on all drawable objects associated with this view. final void layout (int l, int t, int r, int b) Assign size and position to the view and all its children This is the second phase of the layout mechanism. void measureChild(View Child, int parentWidthMeasureSpec, int parentHeightMeasureSpec) Ask one of the nineteen of this view to measure, taking into account measurespec requirements for this view and its padding. void measureChildWithMargins(View child, int parentWidthMeasureSpec, int widthUsed, int parentHeightMeasureSpec, int heightUsed) one of the child costumes of this view to measure, taking into account both measurespec requirements for this view, as well as its padding and margins. void measureChildren (int widthMeasureSpec, int heightMeasureSpec) Ask all children of this view to take this view, considering both MeasureSpec MeasureSpec for this view and its padding. void notifySubtreeAccessibilityStateChanged(See child, View source, int changeType) Notifies the view parent that the availability state of one of its children has changed and that the structure of the subtree is different. final void offsetDescendantRectToMyCoords(View Child Element, Rect rect) Offset of a rectangle that is in the child coordinate space in our coordinate space. the final void offsetRectIntoDescendantCoords(Subview, Rect rect) The offset of a rectangle that is in our coordinate space to the ancestral coordinate space. void onAttachedToWindow() This is called when the view is attached to the window. int[] onCreateDrawableState(int extraSpace) Generate a new drawable state for this view. void onDescendantInvalidated(View Child, Target View) The target view has been invalidated or the drawing property has been changed to require a hierarchy to be rendered again. If you override this method, you must call through a superclass implementation. void onDetachedFromWindow() This is called when the view is disconnected from the window. boolean onInterceptHoverEvent(MotionEvent event) Implement this method to capture activation events before they are handled by child views. boolean onInterceptTouchEvent(MotionEvent ev) Implement this method to capture all touch screen motion events. abstract void onLayout(boolean changed, int l, int t, int r, int b) Called from the layout when this view should assign a size and position to each of its child rooms. boolean onNestedFling(Target view, float speedX, float speedY, logical use) Request throw from nested coil. boolean onNestedPreFling(Target view, float speedX, float speedY) React to nested throwing before the target view consumes it. boolean onNestedPrePerformAccessibilityAction(View target, int action, Bundle args) React to an accessibility action delegated by the destination child view before the target speaks to it. Subclasses should always call super.onNestedPrePerformAccessibilityAction void onNestedPreScroll(View target, int dx, int dy, int[] consumed) React to nested scrolling in progress before the target view consumes part of the scroll. void onNestedScroll(View target, int dxConsumed, int dyConsumed, int dxUnconsumed, int dyUnconsumed) React to nested scrolls in progress. void onNestedScrollAccepted(View Child, View Target, Int Axes) Respond to a successful nested scroll operation request. boolean onRequestFocusInDescendants(int direction, Rect previouslyFocusedRect) Look for a child to call View#requestFocus on. boolean onRequestSendAccessibilityEvent(View Child, AccessibilityEvent) when a child has requested to send accessibilityEvent and gives the parent the opportunity to extend the event. PointerIcon onResolvePointerIcon(MotionEvent, int pointerIndex) Returns the pointer icon for a motion or if you do not specify an icon. boolean onStartNestedScroll(View Child, Target View, int nestedScrollAxes) Respond to the child view initiating the nesting operation by requesting a nested scroll operation if necessary. void onStopNestedScroll React at the end of a nested scroll operation. void onViewAdded Called when you add a new child to this lookout group. void onViewRemoved Called when a child view is removed from this view group. void recomputeViewAttributes(View Child) Tell the view hierarchy that global view attributes must be re-evaluated. void removeAllViews() Call this method to remove all child views from the View Group. void removeAllViewsInLayout() Called by a subclass of a group of views to remove child views from itself when it must first know its size on the screen before it can calculate how many child views it will render. void removeDetachedView Ends the removal of the detached view. void removeView Note: Do not call this method from view.draw(android.graphics.Canvas), View.onDraw(android.graphics.Canvas), dispatchDraw(android.graphics.Canvas), or any related method. void removeViewAt(int index) Deletes the view at the specified position in the group. void removeViewInLayout Deletes the view during layout. void removeViews(int start, int count) Removes the specified range of views from the group. void removeViewsInLayout(int start, int count) Removes the range of views during layout. void requestChildFocus(See child, See focus) Invoked when a child of that parent wants to focus a logical requestChildRectangleOnScreen(See child, rectangle rectangle, immediate logical) Invoked when a child of this group wants the specified rectangle to be placed on the screen. void requestDisallowInterceptTouchEvent(boolean disallowIntercept) Called when a child does not want this parent and its ancestors to capture touch events using ViewGroup#onInterceptTouchEvent(MotionEvent). boolean requestFocus(int direction, Rect previouslyFocusedRect) Call this to try to focus on a specific view or to one of its children and give it hints about the direction and specific rectangle that the focus comes from. Searches for a view to focus on following the setting specified by getDescendantFocusability(). boolean requestSendAccessibilityEvent(View Child, AccessibilityEvent) Raised by the child to request from the parent to send the AccessibilityEvent event. void requestTransparentRegion Called when a child wants the view hierarchy to collect and report regions to the window redpositor. Logical RestoreDefaultFocus() Gives focus to the default focus view in the view hierarchy that has that view as the root. void scheduleLayoutAnimation() Schedules to play layout animations after the next layout run of this view group. Void Void addsStates) Specifies whether this viewgroup drawable states also account for its inchable states of children. void setAlwaysDrawnWithCacheEnabled(boolean always) This method has been deprecated at API level 23. From Build.VERSION_CODES. M, this property is ignored. Children's views may no longer be disabled by parents for their caching behavior. void setAnimationCacheEnabled(logical value enabled) This method has been deprecated at API level 23. From Build.VERSION_CODES. M, this property is ignored. The 19th caching behavior can be controlled using View#setLayerType(int, Paint). void setChildrenDrawingCacheEnabled(logical value enabled) This method has been deprecated at API level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, such as alpha animations, View.setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call view.draw(android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. void setChildrenDrawingOrderEnabled(logical value enabled) Tells the viewgroup whether to draw its children in the order defined by the getChildDrawingOrder(int, int) method. void setChildrenDrawnWithCacheEnabled(logical value enabled) This method has been deprecated at API level 23. From Build.VERSION_CODES. M, this property is ignored. Child views can no longer be forced to cache their rendering state by parents. Instead, use view#setLayerType(int, Paint) in each view. void setClipChildren(boolean clipChildren) By default, children are clipped to the boundaries before drawing. void setClipToPadding(boolean clipToPadding) Sets whether this viewgroup will attract your children to padding and resize (but not clip) any edgeeffect to the padded region if padding is present. void setDescendantFocusability(int focusability) Set the child accuracy of this view group. void setLayoutAnimation(LayoutAnimationController Sets the layout animation controller that is used to animate the group's whisk after the first layout. void setLayoutAnimationListener(Animation.AnimationListener animationListener) Specifies the animation listener to which layout animation events must be sent. void setLayoutMode(int layoutMode) Sets the alignment base during layout of this group of views. void setLayoutTransition(LayoutTransition transition) Sets the LayoutTransition object for this viewgroup. void setMotionEventSplittingEnabled(boolean split) Enable or disable motionevents split into multiple child injuries when sending touch events. void setOnHierarchyChangeListener(ViewGroup.OnHierarchyChangeListener listener) Register the callback to be called when a child is added or removed from this view. void setPersistentDrawingCache(int drawingCacheToKeep) This method has been deprecated in api level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, such as alpha animations, View.setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call view.draw(android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. void setStaticTransformationsEnabled(boolean enabled) When this property is set to true, this viewgroup supports static transformations on the base; this causes getChildStaticTransformation(android.view.View, android.view.animation.Transformation) to call when a child is drawn. void setTouchscreenBlocksFocus(boolean touchscreenBlocksFocus) Set whether this viewgroup should ignore focus requests for itself and your injuries. void setTransitionGroup(boolean isTransitionGroup) Changes whether this view group should be treated as a single entity during an activity transition. void setWindowInsetsAnimationCallback(WindowInsetsAnimation.Callback callback) Sets WindowInsetsAnimation.Callback to notify you of window animations that cause snippets. boolean shouldDelayChildPressedState() Returns true if the press state should be delayed for children or children of this viewgroup. boolean showContextMenuForChild(See originalView, float x, float y) Shows the context menu for a specific view or its ancestors anchored to a specific coordinate relative to the view. Logical originalView) Shows the context menu for the specified view or its ancestors. ActionMode startActionModeForChild(See originalView, ActionMode.Callback callback, int type) Run a specific type of action mode for a specific view. Mode of operation originalView, ActionMode.Callback callback) Start action mode for a specific view with the default ActionMode type#TYPE_PRIMARY. void startLayoutAnimation() Starts the layout animation. void startViewTransition(View view) This method tells The ViewGroup that a given View object, which should have this ViewGroup as its parent, should be stored around (re-displayed when ViewGroup draws its sensation), even if it is removed from its parent. void suppressLayout(boolean suppress) Tells this viewgroup to skip all layout() calls until layout suppression is disabled with a later suppressLayout(false) call. void view addChildrenForAccessibility(ArrayList<View> outChildren) Adds children of this view that are relevant for accessibility to the list as output. void addExtraDataToAccessibilityNodeInfo(AccessibilityNodeInfo info, String extraDataKey, Bundle arguments) Adds additional data to AccessibilityNodeInfo based on an explicit request for additional data. void addFocusables(ArrayList<View> views, int direction) Add all views that are children of this view (probably in this view if it is focused itself) to the views. void addFocusables(ArrayList<View> views, int direction, int focusableMode) Adds all focusable views that are children of that view (possibly in this view if focused) to the views. void addKeyboardNavigationClusters Adds any keyboard navigation cluster roots that are children of this view (possibly including view if it is a cluster root itself) to views. void addOnAttachStateChangeListener(View.OnAttachStateChangeListener listener) Add receiver to include state changes. void addOnLayoutChangeListener(View.OnLayoutChangeListener listener) Add a listener that will be called when view boundaries change due to layout processing. void addOnUnhandledKeyEventListener(View.OnUnhandledKeyEventListener listener) Adds a listener that receives unsupported KeyEvents. void addTouchables(ArrayList<View> views) Add all touch views that are children of this view (possibly in this view, if it is touchable) to the views. ViewPropertyAnimator animate() This method returns a ViewPropertyAnimator object that can be used to animate specific properties in that view. void announceForAccessibility(CharSequence text) A convenience method for sending accessibilityevent#TYPE_ANNOUNCEMENT AccessibilityEvent to suggest that the accessibility service post specific text to its users. void autofill Automatically fills the contents of this view void autofill (Wartości SparseArray)<AutofillValue> Automatycznie wypełnia zawartość wirtualnych trzęsienie w tym widoku.</AutofillValue> </View> </View> </View> </View> </View> </View> Trigger drawing scroll bars. boolean awakenScrollBars(int startDelay) Trigger drawing scroll bars. boolean awakenScrollBars() Trigger drawing scroll bars. void bringToFront() Reorder the view from the tree to make it visible on other peer views. void buildDrawingCache(boolean autoScale) This method has been deprecated in api level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. void buildDrawingCache() This method has been deprecated in api level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. void buildLayer() Forces this view to be layered and rendered in its layer. boolean callOnClick() Directly call all attached OnClickListener. Boolean See if you can make a resolution to the direction of the layout. boolean canResolveTextAlignment() Check that the text alignment resolution cannot be executed. boolean canResolveTextDirection() Verify that the text direction resolution can be executed. boolean canScrollHorizontally(int direction) Check to see if this view can scroll horizontally in a specific direction. Boolean Boolean direction) Make sure that this view can be scrolled vertically in the specified direction. Final void cancelDragAndDrop() Cancels an ongoing drag-and-drop operation. void cancelLongPress() Cancels pending long press. final void cancelPendingInputEvents() Cancel any deferred high-level input events that were previously posted to the event queue. boolean checkInputConnectionProxy(View view) Called by InputMethodManager when a view that is not the current target of the input connection tries to connect to the manager. void clearAnimation() Cancels all animations for this view. void clearFocus() Called when this view wants to give up focus. Static int combineMeasuredStates(int curState, int newState) Merge two states according to getMeasuredState(). int computeHorizontalScrollExtent() Calculates the horizontal range of the horizontal thumb scroll bar in the horizontal range. int computeHorizontalScrollOffset() Calculate the horizontal thumb offset of the horizontal scroll bar in the horizontal range. int computeHorizontalScrollRange() Calculate a horizontal range that represents a horizontal scroll bar. void computeScroll() Called by the parent to request that the child update its values for mScrollX and mScrollY if necessary. WindowInsets computeSystemWindowInsets(WindowInsets in, Rect outLocalInsets) Compute insets that should be consumed by this view and the those that should propagate to those under it. int computeVerticalScrollExtent() Calculate the vertical thumb range of the vertical scroll bar in the vertical range. int computeVerticalScrollOffset() Calculate the vertical thumb offset of the vertical scroll bar in the horizontal range. int computeVerticalScrollRange() Calculate the vertical range that represents the vertical scroll bar. AccessibilityNodeInfo createAccessibilityNodeInfo() Returns AccessibilityNodeInfo representing this view from the AccessibilityService point of view. void createContextMenu (ContextMenu menu) Show the context menu for this view. void destroyDrawingCache() This method has been deprecated in api level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing in the view. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE Config.HARDWARE real-time shadows and trimming contours. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. WindowInsets dispatchApplyWindowInsets(WindowInsets insets) Request that window snippets be applied to that view or view in a sub tree. boolean dispatchCapturedPointerEvent(MotionEvent event) Pass the captured pointer event down to the focused view. void dispatchConfigurationChanged(Configuration newConfig) Send a notification that the resource configuration has changed in the view hierarchy. void dispatchDisplayHint(int hint) Send a hint as to whether this view is displayed. boolean dispatchDragEvent(DragEvent event) Detects whether this view is enabled and has a drag event listener. void dispatchDraw (Canvas canvas) Called by drawing to draw child views. void dispatchEs drawableHotspotChanged to all of this View's children. void dispatchFinishTemporaryDetach() Dispatch onFinishTemporaryDetach() to this view and its direct child costumes if it is a container view. boolean dispatchGenericFocusedEvent(MotionEvent event) Send the general motion event to the currently focused view. boolean dispatchGenericMotionEvent(MotionEvent event) Send a general motion event. boolean dispatchGenericPointerEvent(MotionEvent event) Send a general motion event to the view below the first pointer. boolean dispatchHoverEvent(MotionEvent event) Send activation event. boolean dispatchKeyEvent(KeyEvent event) Send the key event to the next view on the focus path. boolean dispatchKeyEventPreIme(KeyEvent event) Dispatch a key event before processing it with any input method associated with the view hierarchy. boolean dispatchKeyShortcutEvent(KeyEvent event) Raises a key hash event. boolean dispatchNestedFling(float velocityX, float velocityY, boolean consumed) Send a throw to a nested scrolling parent. boolean dispatchNestedPreFling(float velocityX, float velocityY) Sends throwing to the nested parent scroll before it is processed by this view. boolean dispatchNestedPrePerformAccessibilityAction(int action, Bundle arguments) Report an accessibility action to the parents of this view for delegated processing. boolean dispatchNestedPreScroll(int dx, int dy, int[] consumed, int[] offsetInWindow) Dispatch one step of a nested scroll in progress before this view consumes any part of it. boolean dispatchNestedScroll(int dxConsumed, int dyConsumed, int dxUnconsumed, int dyUnconsumed, int[] offsetInWindow) Send one step of the nested scroll in progress. void dispatchPointerCaptureChanged(boolean hasCapture) boolean dispatchPopulateAccessibilityEvent(AccessibilityEvent event) Sends accessibilityEvent to and then to its bundles to add text content to the event. void dispatchProvideAutofillStructure (view structure structure, int flags) int) Create View Structures for automatic populating down the hierarchy when an assist structure is created as part of an autofill request. void dispatchProvideStructure(ViewStructure structure) Dispatches the creation of a view structure down the hierarchy. void dispatchRestoreInstanceState(SparseArray<Parcelable> container) Called by restoreHierarchyState(android.util.SparseArray) to retrieve the status for this view and its minor injuries. void dispatchSaveInstanceState(SparseArray<Parcelable> container) Called by saveHierarchyState(android.util.SparseArray) to store the state for this view and its damage. void dispatchSetActivated(boolean activated) Shipping set Activated to all children of this view. void dispatchSetPressed(boolean pressed) Dispatch setPressed to all of this View's children. void dispatchSetSelected(boolean selected) Send the setSelektowany to all costumes of this view. void dispatchStartTemporaryDetach() Dispatch onStartTemporaryDetach() to this view and its direct privileged costumes if it is a container view View void dispatchSystemUiVisibilityChanged(int visibility) This method has been deprecated at API level 30. Use WindowInsets#isVisible(int) to learn more about the display capabilities of the system bar by setting onapplyWindowInsetsListener in this view. boolean dispatchTouchEvent(MotionEvent event) Pass the touch screen movement event down to the target view or that view if it is the target. boolean dispatchTrackballEvent Pass the trackball movement event down to focused view. boolean dispatchUnhandledMove(See focused, int direction) This method is the last chance for the focused view and its ancestors to respond to the arrow key. void dispatchVisibilityChanged(View changedView, int visibility) The visibility of the view changes the view hierarchy. void dispatchWindowFocusChanged(boolean hasFocus) Called when a window containing this view gains or loses window focus. void dispatchWindowInsetsAnimationEnd(WindowInsetsAnimation animation) Dispatches WindowInsetsAnimation.Callback#onEnd(WindowInsetsAnimation) after the window snippets animation is complete. void dispatchWindowInsetsAnimationPrepare(WindowInsetsAnimation animation) Dispatches WindowInsetsAnimation.Callback#onPrepare(WindowInsetsAnimation) when Window Insets animation is being prepared. WindowInsets dispatchWindowInsetsAnimationProgress(WindowInsets insets, List<WindowInsetsAnimation> runningAnimations) Dispatches WindowInsetsAnimation.Callback#onProgress(WindowInsets, List) when Window Insets animation makes progress. WindowInsetsAnimation.Bounds dispatchWindowInsetsAnimationStart(WindowInsetsAnimation animation, WindowInsetsAnimation.Bounds bounds) Dispatches WindowInsetsAnimation.Callback#onStart(WindowInsetsAnimation, Bounds) Window Insets animation is started. void dispatchWindowSystemUiVisiblityChanged(int visible) Ta metoda została przestarzała w interfejsie API na poziomie 30. SystemUiVisibility</WindowInsetsAnimation> </Parcelable> </Parcelable> </Parcelable> are obsolete. Instead, use the WindowInsetsController. void dispatchWindowVisibilityChanged(int visibility) Dispatch window visibility changes the view hierarchy. void draw(Canvas canvas) Manually render this view (and all its children) to the canvas. void drawableHotspotChanged(float x, float y) This function is called every time the view hotspot changes and must be propagated to drawables or child views managed by the view. void drawableStateChanged() This function is called each time the state of the view changes in such a way that it affects the state of the drawables displayed. View findFocus() Find a view in a hierarchy rooted in that view that currently has focus. Final <T extends View>T findViewById(int id) Finds the first child view with this ID, same view if id matches getId(), or null if id is invalid (< 0)= or= there= is= no= matching= view= in= the= hierarchy.= final=> <T extends View> T findViewWithTag(Object tag) Look for a child view with this tag. void findViewsWithText(ArrayList<View> outViews, CharSequence searched, int flags) Finds views that contain the text. API Origins 20 uses dispatchApplyWindowInsets(android.view.WindowInsets) to apply snippets to views. Views should replace onApplyWindowInsets(android.view.WindowInsets) or use setOnApplyWindowInsetsListener(android.view.View.OnApplyWindowInsetsListener) to implement support for their own snippets. View focusSearch(int direction) Find the nearest view in a specific direction that can focus. void forceHasOverlappingRendering(boolean hasOverlappingRendering) Sets the overlapping rendering behavior for this view (see hasOverlappingRendering() for more information about this behavior). void forceLayout() Forces this view to be deployed during the next layout run. CharSequence getAccessibilityClassName() Returns the class name of this object to be used for accessibility purposes. View.AccessibilityDelegate getAccessibilityDelegate() Returns a delegate to implement accessibility support through the composition. int getAccessibilityLiveRegion() Gets live region mode for this view. AccessibilityNodeProvider getAccessibilityNodeProvider() Gets the provider to manage the virtual view hierarchy rooted in that view and reported to AccessibilityServices that explore the contents of the window. CharSequence getAccessibilityPaneTitle() Get pane title for accessibility purposes. int getAccessibilityTraversalAfter() Gets the view ID after which this is visited in traversal availability. int getAccessibilityTraversalBefore() Gets the view ID before which it is w traversal dostępności. float getAlpha() Krycie widoku. Pobierz animację getAnimation()</View> </T extends View> </T extends View> View> animation associated with this view. The getAnimationMatrix() matrix returns the current view transformation matrix. IBinder getApplicationWindowToken() Get a unique token identifying the real top-level window of the window to which this view is attached. int[] getAttributeResolutionStack(int attribute) Returns an ordered list of resource IDs that are taken into account when recognizing attribute values for this view. Map<Integer, Integer> getAttributeSourceResourceMap() Returns the attribute resource ID mapping to the source resource ID where the attribute value is set. String[] getAutofillHints() Gets hints that help AutofillService determine how to automatically populate a view with user data. final AutofillId getAutofillId() Gets the unique logical identifier of this view in action for autocomplete purposes. int getAutofillType() Describes the autocomplete type of this view, so AutofillService can create the correct autocomplete value when the view is automatically populated. AutofillValue getAutofillValue() Gets the current autocomplete value of the view. Drawable getBackground() Gets the background drawable BlendMode getBackgroundTintBlendMode() Returns the blend mode used to apply the hue to the drawable background, if specified. ColorStateList getBackgroundTintList() Returns the hue applied to the drawable background, if specified. PorterDuff.Mode getBackgroundTintMode() Returns the blend mode used to apply the hue to the drawable background, if specified. int getBaseline() Returns the offset of the widget's text baseline from the top bound of the widget. final int getBottom() The lower position of this view relative to its parent. float getBottomFadingEdgeStrength() Returns the strength or intensity of the lower faded edge. int getBottomPaddingOffset() The amount by which you can extend the lower fading area. float getCameraDistance() Gets the distance along the Z axis from the camera to this view. boolean getClipBounds(Rect outRect) Fills the output rectangle with the bounds of the view clip, returning true if successful or false if the view clip boundaries are null. Rect getClipBounds() Returns a copy of the current clipBounds. the final logical value of getClipToOutline() returns the value of whether the outline should be used to trim the contents of the view. final ContentCaptureSession getContentCaptureSession() Gets the session used to notify content capture events. CharSequence getContentDescription() Returns a description of the contents of the view. Final Context getContext() Returns the context in which the view is running so that it can access the current theme, resources, etc. ContextMenu.ContextMenuInfo getContextMenuInfo() Views should implement this if they have additional information to associate with the context menu. final boolean /** Returns whether this view should use the default focus highlight when focused, but there is no </Integer, Integer> Integer> defined in its background. Static int getDefaultSize (int size, int measureSpec) tool to return the default size. Display getDisplay() Gets the logical display to which the view window is attached. final int[] getDrawableState() Returns an array of resource IDs from drawable states representing the current state of the view. Bitmap getDrawingCache() This method has been deprecated in api level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. Bitmap getDrawingCache(logical autoscale) This method has been deprecated in api level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. int getDrawingCacheBackgroundColor() This method has been deprecated in api level 28. The view drawing cache has been obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example, for alpha animations, setLayerType(int, setLayerType(int, supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. int getDrawingCacheQuality() This method has been deprecated in api level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. void getDrawingRect(Rect outRect) Returns the visible boundaries of the view drawing. long getDrawingTime() Returns the start time for drawing the view hierarchy. float getElevation() The base elevation of this view relative to its parent, in pixels. int getExplicitStyle() Returns the resource ID for the style specified using styles=... in an XML element, copies of attributeset or ID_NULL otherwise, if not specified or otherwise not applicable. boolean getFilterTouchesWhenObscured() Gets whether the structure should discard touches when the view window is obscured by another visible window. boolean getFitsSystemWindows() Check the status of setFitsSystemWindows(boolean). int getFocusable() Returns the focusable setting for this view. <View> GetFocusables(int direction) Array find and return all views that are children of that view, possibly including that view if it is concentrated. void getFocusedRect(Rect r) When a view has focus and the user moves away from it, the next view is searched for, starting with a rectangle filled with this method. getForeground() Returns the drawable used as the foreground of this view. int getForegroundGravity() Describes the location of the foreground. BlendMode getForegroundTintBlendMode() Returns Blending </View> </View> used to apply the hue to the foreground drawable, if specified. ColorStateList getForegroundTintList() Returns the hue applied to foreground drawing, if specified. PorterDuff.Mode getForegroundTintMode() Returns the blend mode used to apply the hue to the foreground drawable, if specified. Final boolean getGlobalVisibleRect(Rect r) boolean getGlobalVisibleRect(Rect r, Point globalOffset) If any part of this view is not cropped by any of its parents, return that area in r at global (major) coordinates. The getHandler() final boolean handler getHasOverlappingRendering() returns an overlapping render value that is used internally. final int getHeight() Return the height of the view. void getHitRect(Rect outRect) Press the rectangle in parent coordinates int getHorizontalFadingEdgeLength() Returns the size of the horizontal faded edges used to indicate that more content in this view is visible. int getHorizontalScrollbarHeight() Returns the height of the horizontal scroll bar. Drawable getHorizontalScrollbarThumbDrawable() Returns the currently configured Drawable for thumb horizontal scroll bar, if any, null otherwise. Drawable getHorizontalScrollbarTrackDrawable() Returns the currently configured Drawable for the horizontal scroll bar path, if any, null otherwise. int getId() Returns the ID of this view. int getImportantForAccessibility() Gets the mode for determining whether this view is important for accessibility. int getImportantForAutofill() Gets the mode for determining whether this view is important for autocomplete. int getImportantForContentCapture() Gets the mode for determining whether this view is important for capturing content. boolean getKeepScreenOn() Returns whether the screen should remain on, corresponding to the current value of KEEP_SCREEN_ON. KeyEvent.DispatcherState getKeyDispatcherState() Returns the global KeyEvent.DispatcherState for this view window. int getLabelFor() Gets the view ID for which this view serves as a label for accessibility purposes. int getLayerType() Indicates what type of layer is currently associated with this view. int getLayoutDirection() Returns the layout direction of the solutions for this view. ViewGroup.LayoutParams getLayoutParams() Get layoutparams associated with this view. final int getLeft() The left position of this view relative to its parent. float getLeftFadingEdgeStrength() Returns the strength or intensity of the left faded edge. int getLeftPaddingOffset() The amount by which you want to extend the left fade area. Final boolean value getLocalVisibleRect(Rect r) void getLocationInSurface(int[] location) Calculate view coordinates within the surface. void getLocationInWindow(int[] outLocation) Calculates the coordinates of this view in its window. Void outLocation) Calculates the coordinates of this view on the screen. getMatrix matrix() getMatrix() a transformation matrix for this view that is calculated based on the current rotation, scale, and rotation properties. final int getMeasuredHeight() Like getMeasuredHeightAndState(), but returns only the primary height component (this result is masked by MEASURED_SIZE_MASK). final int getMeasuredHeightAndState() Return full height measurement information for this view calculated by the latest call measure(int, int). final int getMeasuredState() Returns only the state bits of getMeasuredWidthAndState() and getMeasuredHeightAndState(), combined into a single integer. final int getMeasuredWidth() Like getMeasuredWidthAndState(), but returns only the original width component (this result is masked by MEASURED_SIZE_MASK). final int getMeasuredWidthAndState() Returns full-width measurement information for this view calculated by the latest call to measure(int, int). int getMinimumHeight() Returns the minimum view height. int getMinimumWidth() Returns the minimum view width. int getNextClusterForwardId() Gets the root ID of the next keyboard navigation cluster. int getNextFocusDownId() Gets the view ID to use when the next focus is FOCUS_DOWN. int getNextFocusForwardId() Gets the view ID to use when the next focus is FOCUS_FORWARD. int getNextFocusLeftId() Gets the view ID to use when the next focus is FOCUS_LEFT. int getNextFocusRightId() Gets the view ID to use when the next focus is FOCUS_RIGHT. int getNextFocusUpId() Gets the view ID to use when the next focus is FOCUS_UP. View.OnFocusChangeListener getOnFocusChangeListeListener() Returns the callback registered for this view. int getOutlineAmbientShadowColor() ViewOutlineProvider getOutlineProvider() Returns the current ViewOutlineProvider view, which generates an outline that defines the shape of the shadow it casts and allows you to trim the outline. int getOutlineSpotShadowColor() int getOverScrollMode() Returns scroll mode for this view. ViewOverlay getOverlay() Returns an overlay for this view, creating it if it doesn't already exist. int getPaddingBottom() Returns the lower bottom padding of this view. int getPaddingEnd() Returns the final padding of this view depending on its recognized layout direction. int getPaddingLeft() Returns the left padding of this view. int getPaddingRight() Returns the right padding of this view. int getPaddingStart() Returns the initial padding of this view depending on its recognized layout direction. int getPaddingTop() Returns the top padding of this view. final ViewParent getParent() Gets the parent of that view. ViewParent getParentForAccessibility() Gets parent for accessibility purposes. float getPivotX() X the position of the point around which the view is rotated and scaled. float getPivotY() y the point around which the view is rotated and scaled. Indicatorskon Indicatorkon Gets the pointer icon for the current view. The getResources() resource returns the resources associated with this view. final boolean getRevealOnFocusHint() Returns the preferences of this view for disclosure behavior when it gains focus. final int getRight() The right position of this view relative to its parent. float getRightFadingEdgeStrength() Returns the strength or intensity of the right faded edge. int getRightPaddingOffset() The amount by which you can extend the right fade area. View getRootView() Finds a top-notch view in the current view hierarchy. WindowInsets getRootWindowInsets() Provide the original sets of windowinsets that are called to the view hierarchy. float getRotation() Degrees, which view is rotated around the pivot point. float getRotationX() Degrees, which view is rotated around the horizontal axis by the pivot point. float getRotationY() Degrees, which view is rotated around the vertical axis by the pivot point. float getScaleX() The amount that is scaled in x around the pivot point as part of the unscaled view width. float getScaleY() The amount that is scaled around the pivot point as part of the unscaled view height. int getScrollBarDefaultDelayBeforeFade() Returns a delay before the scroll bars disappear. int getScrollBarFadeDuration() Returns the fade time of the scroll bar. int getScrollBarSize() Returns the size of the scroll bar. int getScrollBarStyle() Returns the current scroll bar style. int getScrollIndicators() Returns a bit mask that represents enabled scroll indicators. final int getScrollX() Return the scrolling left position of this view. final int getScrollY() Returns the scrolling top position of this view. int getSolidColor() Replace this if you know that the view is always drawn against a solid color background and must draw fading edges. int getSourceLayoutResId() View may be overstated from the XML layout. final CharSequence getStateDescription() Returns a description of the view state. StateListAnimator getStateListAnimator() Returns the current StateListAnimator, if any. int getSuggestedMinimumHeight() Returns the suggested minimum height that the view should use. int getSuggestedMinimumWidth() Returns the suggested minimum width that the view should use. List<Rect> getSystemGestureExclusionRects() Get a list of areas within the post-layout coordinate space of this view, where the system should not capture touch or other gestures of the pointing device. int getSystemUiVisibility() This method has been deprecated in the Level 30 API. SystemUiVisibility flags are obsolete. Instead, use the WindowInsetsController. The getTag() returns a tag for this view. The getTag(int key) returns the tag associated with this view and the specified key. int getTextAlignment() Returns the alignment of resolved text. int getTextDirection() Returns direction rozwiązał. Znakowanie getTooltipText() </Rect> </Rect> view tooltip text. final int getTop() The top position of this view relative to its parent. float getTopFadingEdgeStrength() Returns the strength or intensity of the top faded edge. int getTopPaddingOffset() The amount by which you can extend the top fade area. TouchDelegate getTouchDelegate() Gets TouchDelegate for this view. ArrayList<View> getTouchables() Find and return all screenable views that are children of that view, possibly in that view if it is touchable. float getTransitionAlpha() This property is intended only for use by the Fade transition, which animates it to produce visual transparency that does not cause the side effect (or get affected) of the alpha property. getTransitionName() Returns the name of the view to be used to identify views in transitions. float getTranslationX() The horizontal location of this view relative to its left position. float getTranslationY() The vertical location of this view relative to its upper position. float getTranslationZ() The position of the depth of this view relative to its elevation. long getUniqueDrawingId() Get the ID used for this view by the drawing system. int getVerticalFadingEdgeLength() Returns the size of the vertical faded edges used to indicate that more content in this view is visible. int getVerticalScrollbarPosition() Drawable getVerticalScrollbarThumbDrawable() Returns the currently configured drawable for thumb vertical scroll bar, if any, otherwise null. Drawable getVerticalScrollbarTrackDrawable() Returns the currently configured Drawable for the vertical scroll bar path, if any, null otherwise. int getVerticalScrollbarWidth() Returns the width of the vertical scroll bar. ViewTreeObserver getViewTreeObserver() Returns ViewTreeObserver for the hierarchy of this view. int getVisibility() Returns the visibility state for this view. final int getWidth() Return the width of the view. int getWindowAttachCount() WindowId getWindowId() Get WindowId for the window to which this view is currently attached. WindowInsetsController getWindowInsetsController() Gets a single WindowInsetsController window that this view is attached to. int getWindowSystemUiVisibility() This method has been deprecated in api level 30. SystemUiVisibility flags are obsolete. Instead, use the WindowInsetsController. IBinder getWindowToken() Get a unique token identifying the window to which this view is attached. int getWindowVisibility() Returns the current visibility of the window to which this view is attached (GONE, INVISIBLE, or VISIBLE). void getWindowVisibleDisplayFrame(Rect outRect) Get the overall visible display size in which the window to which this view is attached has been placed in. float getX() The visual position of x of that view in float getY() The visual position of this view in pixels. float getZ() Visual position with </View> </View> pixels. boolean hasExplicitFocusable() Returns true if this view

is concentrated or if it contains an reachable view for which hasExplicitFocusable() returns true. boolean hasFocus() Returns true if this view has the same focus or is the parent of a view that has focus. hasFocusable() Returns true if this view is concentrated or if it contains an reachable view for which hasFocusable() returns true. hasNestedScrollingParent() Returns true if this view has a nested scrolling parent. boolean hasOnClickListeners() Returns whether this view has OnClickListener attached. boolean hasOnLongClickListeners() Returns whether this view has OnLongClickListener attached. boolean hasOverlappingRendering() Returns whether this view has content that overlaps. boolean hasPointerCapture() Checks the capture state of the pointer. boolean hasTransientState() Indicates whether the view is currently tracking a transient state, that the application should not deal with saving and restoring, but that the framework should take special care to keep it when possible. hasWindowFocus() Returns true if this view is in the window where the focus is currently in the window. Static view overstate (context context, int resource, lookout group root) Inflate view from XML resource. void voidate() Invalidate the entire view. void voidate(Rect dirty) This method has been deprecated in api level 28. Switching to hardware accelerated rendering in API 14 reduced the importance of a dirty rectangle. In API 21, the rectangle is ignored completely in favor of the internally calculated area. For this reason, customers are encouraged to simply call invalidate(). void invalidate(int l, int t, int r, int b) This method has been deprecated at API level 28. Switching to hardware accelerated rendering in API 14 reduced the importance of a dirty rectangle. In API 21, the rectangle is ignored completely in favor of the internally calculated area. For this reason, customers are encouraged to simply call invalidate(). void voidDrawable(Drawable drawable) Invalidates the specified Drawable. void invalidateOutline() Called to rebuild the outline of this view from its viewoutlineProvider boolean isAccessibilityFocused() Returns whether this view is focused on accessibility. boolean isAccessibilityHeading() Gets whether this view is a header for accessibility purposes. the logical value isActivated() Indicates the activation status of this view. isAttachedToWindow() Returns true if this view is currently attached to the window. boolean isClickable() Indicates whether this view responds to click events or not. isContextClickable() Indicates whether the the view responds to context clicks or not. true isDirty() True if this view has changed since the last drawing. boolean isDrawingCacheEnabled() This method has been deprecated in api level 28. The view drawing cache has been obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. boolean isDuplicateParentStateEnabled() Indicates whether this causes duplicates of its drawing state from its parent. logical value isEnabled() Returns the enabled state for this view. The final boolean value isFocusable() returns whether this view is currently able to focus on this. the final logical value isFocusableInTouchMode() When the view is concentrated, you may not want to focus in touch mode. isFocused() Returns true if this view has trailing focus logical values isFocusedByDefault() Returns whether this view should receive focus when the focus is restored for the view hierarchy that contains the view. boolean isForceDarkAllowed() See setForceDarkAllowed(boolean) boolean isHapticFeedbackEnabled() boolean isHardwareAccelerated() Indicates whether this view is included in the hardware accelerated window or not. logical value isHorizontalScrollBarEnabled() Indicate whether the horizontal scroll bar should be drawn or not. isHovered() Returns true if the view is currently inducement. the boolean value isImportantForAccessibility() Calculates whether this view should be visible to accessibility. The final logical value isImportantForAutofill() indicates to Android whether assiststructure.ViewNode associated with this view is considered important for autocomplete purposes. The final logical value isImportantForContentCapture() prompts Android whether this view is considered important for capturing content, based on the value explicitly set by setImportantForContentCapture(int) and heuristics when it is IMPORTANT_FOR_CONTENT_CAPTURE_AUTO. boolean isInEditMode() Indicates whether this view is currently in edit mode. the boolean value isInLayout() Returns whether the view hierarchy is currently in the process of layout. Boolean Returns whether the device is currently in touch mode. final boolean value isKeyboardNavigationCluster() Returns whether this view is the root navigation cluster. the boolean isLaidOut() returns true if this view has been through one or more layouts since it was last attached to or disconnected from the window. boolean isLayoutDirectionResolved() boolean isLayoutRequested() Indicates whether the layout of this view will be required during the next run of the hierarchy layout. the boolean isLongClickable() indicates whether this view responds to long click events or not. isNestedScrollingEnabled() returns true if nested scrolling is enabled for this view. boolean isOpaque() Indicates whether this view is opaque. boolean isPaddingOffsetRequired() If the view draws content inside the padding and allows edges to fade, it must support padding offsets. logical isPaddingRelative() Return if padding is set by relative values setPaddingRelative(int, int, int, int) or by logical isPivotSet() Returns whether the Pivot was set by calling setPivotX(float) or setPivotY(float). boolean isPressed() Indicates whether the view is currently in a pressed state. boolean isSaveEnabled() Indicates whether this view will save its state (i.e. whether the onSaveInstanceState() method will be called. whether the entire hierarchy in this view will save its state when the state is saved from its parent pass. if the scroll bars disappear when this view does not scroll the logical valuesebrany() Indicates the selection state of this view. the final logical value isShowingLayoutBounds() Returns true when the view is attached, and the system developer setting to show boolean isShown() Returns the visibility of this view and all its ancestors logical value isSoundEffectsEnabled() of the final logical valueTemporarilyDetached() Informs whether the view is in the state between onStartTemporaryDetach() and onFinishTemporaryDetach(). boolean isTextAlignmentResolved() boolean isTextDirectionResolved() boolean isVerticalFadingEdgeEnabled() Indicate whether vertical edges are faded when the view is scrolling horizontally. boolean isVerticalScrollBarEnabled() Indicate whether the vertical scroll bar should be drawn or not. isVisibleToUserForAutofill(int virtualId) calculates whether this virtual AutoComplete view is visible to the user. void jumpDrawablesToCurrentState() Call Drawable#jumpToCurrentState() on all drawable objects with this view. See keyboardNavigationClusterSearch(See currentKaster, int direction) Find the nearest keyboard navigation cluster in a specific direction. void(int l, int t, int r, int b) b) Size and position to the view and all its children This is the second phase of the layout mechanism. The final measure of voidness (int widthMeasureSpec, int heightMeasureSpec) is a call to find out how big the view should be. static int[] mergeDrawableStates(int[] baseState, int[] additionalState) Merge their own state values in additionalState to base state values of baseState that were returned by onCreateDrawableState(int). void offsetLeftAndRight(int offset) Move the horizontal location of this view by a specified number of pixels. void offsetTopAndBottom(int offset) Move the vertical location of this view by a specified number of pixels. void onAnimationEnd() Called by the parent viewgroup to notify you of the end of the animation currently associated with this view. void onAnimationStart() Called by the parent viewgroup to notify you when the animation currently associated with this view starts. WindowInsets onApplyWindowInsets(WindowInsets insets) Called when the view should apply WindowInsets according to its internal policies. void onAttachedToWindow() This is called when the view is attached to the window. void onCancelPendingInputEvents() Called by calling cancelPendingInputEvents() in this view or parent view. boolean onCapturedPointerEvent Implement this method to handle captured logical pointer events onCheckIsTextEditor() Verify that the called view is a text editor, in which case it is worth automatically displaying a soft input window for it. void onConfigurationChanged(Configuration newConfig) Called when the current configuration of resources used by the application has changed. void onCreateContextMenu(ContextMenu menu) Views should implement this if the view itself intends to add items to the context menu. int[] onCreateDrawableState(int extraSpace) Generate a new drawable state for this view. InputConnection onCreateInputConnection(EditorInfo outAttrs) Create a new inputconnection for inputmethod to interact with the view. void onDetachedFromWindow() This is called when the view is disconnected from the window. void onDisplayHint(int hint) Gives this view a clue whether it is displayed or not. boolean onDragEvent(DragEvent event) Handles drag events sent by the system when startDragAndDrop() is called. void onDraw(Canvas canvas) Implement this to make a drawing. void onDrawForeground(Canvas canvas) Draw any foreground content for this view. Final void onDrawScrollBars Request a horizontal drawing and a vertical scroll bar. boolean onFilterTouchEventForSecurity Filter the touch event to apply security policies. void onFinishInflate() Finalize overstate the view from an XML file. Void Called after onStartTemporaryDetach() after the container is finished, changing the view. void onFocusChanged(boolean gainFocus, direction int, int, previouslyFocusedRect) Called by the view system when the focus state of that view changes. boolean onGenericMotionEvent Implement this method to handle general motion events. void onHoverChanged(boolean hovered) Implement this method to handle activation state changes. boolean onHoverEvent Implement this method for handling activation events. void onInitializeAccessibilityEvent(AccessibilityEvent event) Initiates accessibilityEvent with information about the view that is the source of the event. void onInitializeAccessibilityNodeInfo(AccessibilityNodeInfo info) Initiates AccessibilityNodeInfo with information about this view. boolean onKeyDown(int keyCode, KeyEvent event) Default implementation keyevent.Callback#onKeyDown(int, KeyEvent): Tap the view when you release KeyEvent#KEYCODE_DPAD_CENTER or KeyEvent#KEYCODE_ENTER if the view is enabled and clickable. boolean onKeyLongPress(int keyCode, KeyEvent event) Default implementation of KeyEvent.Callback#onKeyLongPress(int, KeyEvent): always returns false (does not handle events). boolean onKeyMultiple (int keyCode, int repeatCount, KeyEvent event) Default implementation of KeyEvent.Callback#onKeyMultiple(int, int, KeyEvent): always returns false (does not handle events). boolean onKeyPreIme(int keyCode, KeyEvent event) Support the key event before processing using any input method associated with the view hierarchy. boolean onKeyShortcut(int keyCode, KeyEvent event) Raised in focused view when a key hash event is not handled. boolean onKeyUp(int keyCode, KeyEvent event) Default implementation KeyEvent.Callback#onKeyUp(int, KeyEvent): Click the view when you release KeyEvent#KEYCODE_DPAD_CENTER, KeyEvent#KEYCODE_ENTER, or KeyEvent#KEYCODE_SPACE. void onLayout(boolean changed, int left, int top, int right, int bottom) Called from the layout when this view should assign a size and position to each of its child rooms. void onMeasure (int widthMeasureSpec, int heightMeasureSpec) Measure the view and its contents to determine the measured width and measured height. void onOverScrolled (int scrollX, int scrollY, boolean clampedX, boolean clampedY) Triggered by overScrollBy (int, int, int, int, int, int, int, int, boolean) in response to scrolling results. void onPointerCaptureChange(boolean hasCapture) Called when a window has just acquired or lost a pointer capture. void onPopulateAccessibilityEvent(AccessibilityEvent event) Triggered from dispatchPopulateAccessibilityEvent(android.view.accessibilityEvent) giving this view a chance to populate the accessibility event with text content. void onProvideAutofillStructure(ViewStructure structure, int flags) Populates ViewStructure to fullfil autocomplete requests. void onProvideAutofillVirtualStructure flags) Fills fills containing virtual children to fullfil autocomplete request. void onProvideContentCaptureStructure(ViewStructure structure, int flags) fills the view structure for capturing content. void onProvideStructure(ViewStructure structure) Called when the help structure is retrieved from the view as part of activity.onProvideAssistData. void onProvideVirtualStructure(ViewStructure structure) Called when a help structure is retrieved from a view as part of activity.onProvideAssistData to generate an additional virtual structure in that view. PointerIcon onResolvePointerIcon(MotionEvent, int pointerIndex) Returns a pointer icon for a motion event or null if it does not specify an icon. void onRestoreInstanceState(Parcelable state) Hook allows the view to reapply a representation of its internal state that was previously generated by onSaveInstanceState(). void onRtlPropertiesChanged(int layoutDirection) Called when you change any RTL property (layout direction or text direction or text alignment). Parcelable onSaveInstanceState() Hook allows the view to generate a representation of its internal state, which can later be used to create a new instance with the same state. void onScreenStateChanged(int screenState) This method is called each time the screen state of this view is attached to the changes. void onScrollChanged (int l, int t, int oldl, int oldt) This is invoked in response to an internal scroll in this view (i.e. the view scrolled its own content). boolean onSetAlpha(int alpha) called if there is a Transform that includes alpha. void onSizeChanged (int w, int h, int oldw, int oldh) This is called during the layout when the size of this view has changed. void onStartTemporaryDetach() This is called when the container has to temporarily disconnect the child from ViewGroup#detachViewFromParent(View). boolean onTouchEvent Implement this method to handle touch screen motion events. boolean onTrackballEvent Implement this method to handle trackball motion events. void onVisibilityAggregated(boolean isVisible) Called when the visibility of this view may be affected by a change in the view itself, the parent view, or the window to which the view is attached. void onVisibilityChanged(View changedView, int visibility) Called when view visibility or view parent changes. void onWindowFocusChanged(boolean hasWindowFocus) Called when a window containing this view gains or loses focus. void onWindowSystemUiVisibilityChanged(int visible) This method has been deprecated in api level 30. SystemUiVisibility flags are obsolete. Instead, use the WindowInsetsController. void onWindowVisibilityChanged(int visibility) Called when you want the containing window to change its visibility GONE, INVISIBLE i VISIBLE). boolean overScrollBy(int deltaX, int deltaY, int scrollX, int int int scrollRangeX, int scrollRangeY, int maxOverScrollX, int maxOverScrollY, boolean isTouchEvent) Scroll through the view with standard behavior to scroll beyond normal content boundaries. boolean performAccessibilityAction(int action, Bundle arguments) Performs a specific accessibility action in the view. boolean performClick() Call this OnClickListener view if it is defined. boolean performContextClick(float x, float y) Call this view to OnContextClickListener, if defined. boolean performContextClick() Call this view onContextClickListener, if defined. boolean performHapticFeedback(int feedbackConstant) BZZZTT!! 1! Provide tactile feedback to the user for this view. boolean performHapticFeedback(int feedbackConstant, int flags) BZZZTT!! 1! Like performHapticFeedback(int), with additional options. boolean performLongClick(float x, float y) Calls onlongclicklistener this view if it is defined. boolean performLongClick() Calls onlongclicklistener this view if it is defined. void playSoundEffect(int soundConstant) Play the sound effect for this view. logical entry (Scoundrel Action) Causes runnable to be added to the message queue. boolean postDelayed(Runnable action, long delayMillis) Causes runnable to be added to the message queue to run after a specified period of time has elapsed. void postInvalidate() Cause the next cycle to be invalidated through an event loop. void postInvalidate (int left, int top, int right, int bottom) Invalidate a specific area in the next cycle by an event loop. void postInvalidateDelayed (long delayMilliseconds, int left, int top, int right, int bottom) Invalidate a specific area in the next cycle by an event loop. void postInvalidateDelayed(long delayMilliseconds) Cause the event loop to invalidate in the next cycle. void postInvalidateOnAnimation(int left, int top, int right, int bottom) Invalidate a specific area in the next step of animation time, typically in the next display frame. void postInvalidateOnAnimation() Invalidate the animation time in the next step, typically in the next display frame. void postOnAnimation(Runnable action) causes runnable execution in the next step of animation time. void postOnAnimationDelayed(Runnable action, long delayMillis) Causes runnable to execute at the next step of animation time after a specified period of time. void refreshDrawableState() Call this to force the view to update its drawable state. void releasePointerCapture() releases the pointer capture. boolean removeCallbacks Removes the specified Runnable file from the message queue. Void listener) Remove the receiver to include state changes. Void Void receiver) Remove the receiver to change the layout. void removeOnUnhandledKeyEventListener(View.OnUnhandledKeyEventListener listener) Removes the listener that receives unsupportable KeyEvents. void requestApplyInsets() Request a new shipment onApplyWindowInsets(android.view.WindowInsets). void requestFitSystemWindows() This method has been deprecated in api level 20. Use requestApplyInsets() for newer versions of the platform. final logical requestFocus(int direction) Call this to try to focus on a specific view or to one of its minors and give it a clue as to what direction of focus is going. final logical requestFocus() Call this to try to focus a specific view or one of its children. boolean requestFocus(int direction, Rect previouslyFocusedRect) Call this to try to focus on a specific view or to one of its children and give it hints about the direction and specific rectangle that the focus comes from. The final logical value of the RequestFocusFromTouch() call this to try to focus on a specific view or to one of its children. void requestLayout() Call this when something has changed, which invalidates the layout of this view. void requestPointerCapture() Request pointer capture mode. boolean requestRectangleOnScreen(Rect rectangle) Request that the rectangle of this view be visible on the screen by scrolling as needed. boolean requestRectangleOnScreen(Rectangle rectangle, logical instant) Request that the rectangle of this view be visible on the screen by scrolling as needed. requestUnbufferedDispatch(int source) An unbuffered request dispatches a given event source class to this view. Final Invalidation requestUnbufferedDispatch(MotionEvent event) An unbuffered request dispatches a given MotionEvents stream to this view. Final <T extends View>T requireViewById(int id) Finds the first child view with the given ID, the view itself if the ID matches getId() or reports illegalargumentException if the ID is invalid or there is no matching view in the hierarchy. void resetPivot() Clears all previously set by calling setPivotX(float) or setPivotY(float). Static int resolveSize(int size, int measureSpec) ResolveSizeAndState(int, int, int) returned only MEASURED_SIZE_MASK bits of the result. int measureSpec, int childMeasuredState) A tool for reconciving the desired size and state, with restrictions imposed by measurespec. Logical RestoreDefaultFocus() Gives focus to the default focus view in the view hierarchy that has that view as the root. void restoreHierarchyState(SparseArray<Parcelable> container) Restore frozen hierarchy state z danego kontenera. , int[] styleable, AttributeSet attrs, TypedArray t, int defStyleAttr, int defStyleRes) Sklepy</Parcelable> </T extends View> View> information about attributes. void saveHierarchyState(SparseArray<Parcelable> container) Store this view of the frozen state hierarchy in a given container. void scheduleDrawable (Drawable who, Runnable what, long when) Action schedules on drawable occur at a specific time. void scrollBy (int x, int y) Move the scrolling position of the view. void scrollTo(int x, int y) Set the scrolling position of the view. void sendAccessibilityEvent(int eventType) Sends an availability event of a given type. void sendAccessibilityEventUecked(AccessibilityEvent event) This method behaves exactly like sendAccessibilityEvent(int), but takes AccessibilityEvent as an empty argument and does not check whether availability is enabled. void setAccesDelsibilityegate(View.View.AccessibilityDelegate delegate) Sets the delegate to implement accessibility support through composition (as opposed to inheritance). void setAccessibilityHeading(boolean isHeading) Set if the view is a header for the content section for accessibility purposes. void setAccessibilityLiveRegion(int mode) Sets the live region mode for this view. void setAccece 2018 (CharSequence accessibilityPaneTitle) Visually distinct parts of the window with window semantics are considered panes for accessibility purposes. void setAccessibilityTraversalAfter(int afterId) Sets the view ID after which it is visited in traversal accessibility. void setAccessibilityTraversalBefore(int beforeId) Sets the view ID before which it is visited in traversal accessibility. void setActivated(boolean activated) Changes the activation state of this view. void setAlpha(float alpha) Sets the opacity of the view to a value from 0 to 1, where 0 indicates that the view is completely transparent and 1 indicates that the view is completely opaque. void setAnimation Sets the next animation to play for this view. void setAnimationMatrix(Matrix) Changes the transformation matrix in the view. void setAutofillHints autofillHints) Sets hints that help AutofillService determine how to automatically populate a view with user data. void setAutofillId(AutofillId id) Sets the unique logical identifier of this view in action for autocomplete purposes. void setBackground (Drawable background) Set the background to a given Drawable or remove the background. void setBackgroundColor(int color) Sets the background color for this view. void setBackgroundDrawable(Drawable background) This method has been deprecated in api level 16. use setBackground(android.graphics.drawable.Drawable) instead of void setBackgroundResource(int resid) To set the background for a given resource. void setBackgroundTintBlendMode(BlendMode blendMode) Specifies the blend mode used to apply the hue specified by setBackgroundTintList(android.content.res.ColorStateList)} for background drawing. </Parcelable> </Parcelable> shade) Applies a shade to the background that you can draw. void setBackgroundTintMode(PorterDuff.Mode tintMode) Specifies the blending mode used to apply the hue specified by setBackgroundTintList(android.content.res.ColorStateList)} for background drawing. final void setBottom(int bottom) Sets the bottom position of this view relative to its parent. void setCameraDistance Sets the distance along the Z axis (orthogonal to the X/Y plane where views are drawn) from the camera to this view. void setClickable enables or disables click events for this view. void setClipBounds(Rect clipBounds) Sets the rectangular area in this view to which the view will be cropped as you draw. void setClipToOutline(boolean clipToOutline) Specifies whether the view outline should be used to trim the contents of the view. void setContentCaptureSession(ContentCaptureSession contentCaptureSession) Sets (optionally) the ContentCaptureSession associated with this view. void setContentDescription(CharSequence contentDescription) Sets the description of the view content. void setContextClickable(boolean contextClickable) Enables or disables context clicking for this view. void setDefaultFocusHighlightEnabled(boolean defaultFocusHighlightEnabled) Sets whether this view should use the default focus highlight when concentrated, but there is no R.attr.state_focused defined in the background. void setDrawingCacheBackgroundColor(int color) This method has been deprecated in api level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. void setDrawingCacheEnabled(logical value enabled) This method has been deprecated at API level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases, when caching layers are useful, useful, when it comes to alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. void setDrawingCacheQuality(int quality) This method has been deprecated in api level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. void setDuplicateParentStateEnabled(boolean enabled) Enables or disables parent state duplication in this view. void setElevation(float elevation) Sets the basic elevation of this view in pixels. void setEnabled (boolean enabled) Set the enabled state of this view. void setFadingEdgeLength(int length) Set the size of the faded edge used to indicate that more content is available in this view. void setFilterTouchesWhenObscured(boolean enabled) Sets whether the structure should discard touches when the view window is obscured by another visible window. void setFitsSystemWindows(boolean fitSystemWindows) Sets whether this view should include system screen decorations, such as a status bar, and insert its contents; that is, controlling whether the default implementation of fitSystemWindows (android.graphics.Rect) will be performed. void setFocusable(boolean focusable) Set whether this view can receive the focus(boolean focusable) Set whether this view can receive the void setFocusable(int focusable) sets whether this view can receive focus. void setFocusableInTouchMode(boolean focusableInTouchMode) Set whether this view can receive focus in touch mode. void setFocusedByDefault(boolean isFocusedByDefault) Sets whether this view should receive focus when focus is restored for the view hierarchy that contains the view. void setForceDarkAllowed(boolean allow) sets whether to allow dark extortion to apply to this view. void setForeground(Drawable foreground) Supply Drawable, which is to be rendered on top of all content in the view. void setForegroundGravity(int gravity) Describes how to position the foreground. void setForegroundTintBlendMode(BlendMode blendMode) Specifies the blend mode used to apply the shade specified by setForegroundTintList(android.content.res.ColorStateList)} to draw in the background. void setForegroundTintList Applies a hue to foreground drawing. void setForegroundTintMode(PorterDuff.Mode tintMode) Specifies the blending mode used to apply the hue specified by setForegroundTintList(android.content.res.ColorStateList)} to draw the background. void setHapticFeedbackEnabled(boolean hapticFeedbackEnabled) Set whether this view should have tactile feedback for events such as long presses. void setHasTransientState(boolean hasTransientState) Set whether this view is currently tracking the transient state that the structure should try to keep when possible. void setHorizontalFadingEdgeEnabled(boolean horizontalFadingEdgeEnabled) Define whether horizontal edges should be faded when this view scrolls horizontally. void setHorizontalScrollBarEnabled(boolean horizontalScrollBarEnabled) Define whether a horizontal scroll bar should be drawn or not. void setHorizontalScrollbarThumbDrawable(Drawable drawable) Defines the horizontal thumb drawable void setHorizontalScrollbarTrackDrawable(Drawable drawable) Defines the horizontal drawable void setHovered(boolean hovered) path that sets whether the view is currently tilted. void setId(int id) Sets the ID for this view. void setImportantForAccessibility(int mode) Sets how this view is important for accessibility, that is, if accessibility events are run and whether it is reported to accessibility services that ask for a screen. void setImportantForAutofill(int mode) Sets the mode for determining whether this view is considered important for autocomplete. void setImportantForContentCapture(int mode) Sets the mode for determining whether this view is considered important for capturing content. void setKeepScreenOn(boolean keepScreenOn) Specifies whether the screen should remain on by modifying the KEEP_SCREEN_ON. void setKeyboardNavigationCluster(boolean isCluster) Set whether this view is the root of the keyboard navigation cluster. void setLabelFor(int id) Sets the view ID for which this view serves as a label for accessibility purposes. void setLayerPaint(Paint paint) Updates the Paint object used with the current layer (used only if the current layer type is set to LAYER_TYPE_NONE). void setLayerType(int layerType, Paint paint) Specifies the layer type of the layer in this view. void setLayoutDirection(int layoutDirection) Set the layout direction for this view. void setLayoutParams(ViewGroup.LayoutParams params) Set the layout parameters associated with this view. final void setLeft(int left) Sets the left position of this view relative to its parent. setLeftTopRightBottom (int left, int top, int right, int bottom) Assign size and position to this view. void setLongClickable(boolean longClickable) Enables or disables long-click events for this view. the final set of voidMeasuredDimension (int measuredWidth, int measuredHeight) method must be called by onMeasure (int, int) to store the measured width and measured height. void setMinimumHeight(int minHeight) Sets the minimum view height. void setMinimumWidth(int minWidth) Sets the minimum view width. void setNestedScrollingEnabled(boolean enabled) Enable or disable nested scrolling for this view. void setNextClusterForwardId(int nextClusterForwardId) Sets the view ID to use as the root of the next keyboard navigation cluster. void setNextFocusDownId(int nextFocusDownId) Sets the view ID to use when the next focus is FOCUS_DOWN. void setNextFocusForwardId(int nextFocusForwardId) Sets the view ID to use when the next focus is FOCUS_FORWARD. void setNextFocusLeftId(int nextFocusLeftId) Sets the view ID to use when the next focus is FOCUS_LEFT. void setNextFocusRightId(int nextFocusRightId) Sets the view ID to use when the next focus is FOCUS_RIGHT. void setNextFocusUpId(int nextFocusUpId) Sets the view ID to use when the next focus is FOCUS_UP. void setOnApplyWindowInsetsListener(View.OnApplyWindowInsetsListener listener) Set onapplyWindowInsetsListener to take over the rules for applying window snippets to this view. void setOnCapturedPointerListener(View.OnCapturedPointerListener l) Set the receiver to receive callbacks when the capture state of the view pointer changes. void setOnClickListener(View.OnClickListener l) Register the callback that you want to call when you click this view. void setOnContextClickListener(View.OnContextClickListener l) Register the callback to be called when this view is clicked in context. void setOnCreateContextMenuListener(View.OnCreateContextMenuListener l) Register the callback to be called when creating the context menu for this view. void setOnDragListener(View.OnDragListener l) Register a listener drag callback object for this view. void setOnFocusChangeListener(View.OnFocusChangeListeListener l) Register the callback to be called when the focus of this view changes. Void l) Register the callback to be called when a general traffic event is sent to this view. void setOnHoverListener(View.OnHoverListener l) Register the callback to be called when a hover event is sent to this view. void setOnKeyListener(View.OnKeyListener l) l) the callback that you want to call when you press the hardware key in this view. void setOnLongClickListener(View.OnLongClickListener l) Register the callback to be called when you click and stop this view. void setOnScrollChangeListener(View.OnScrollChangeListelistener l) Register the callback to be called when the scrolling positions X or Y of this view change. void setOnSystemUiVisibilityChangeListener(View.OnSystemUiVisibilityChangeListeListener l) This method has been deprecated at api level 30. Use WindowInsets#isVisible(int) to learn more about the display capabilities of the system bar by setting onapplyWindowInsetsListener in this view. void setOnTouchListener(View.OnTouchListener l) Register the callback to be called when a touch event is sent to this view. void setOutlineAmbientShadowColor(int color) Sets the ambient shadow color that is drawn when the view has a positive Z or elevation value. void setOutlineProvider(ViewOutlineProvider provider) Sets the ViewOutlineProvider view, which generates an outline that defines the shape of the shadow casts and allows you to trim the outline. void setOutlineSpotShadowColor(int color) Sets the color of the spot shadow drawn when the view has a positive Z or elevation value. void setOverScrollMode(int overScrollMode) Set the scroll mode for this view. void setPadding(int left, int top, int right, int bottom) Sets padding. void setPaddingRelative(int start, int top, int end, int bottom) Sets the relative padding. void setPivotX(float pivotX) Sets the x position of the point around which the view is rotated and scaled. void setPivotY(pivotY float) Sets the y position of the point around which the view is rotated and scaled. void setPointerIcon(PointerIcon pointerIcon) Set the pointer icon for the current view. void setPressed(boolean pressed) Sets the press state for this view. final void setRevealOnFocusHint(boolean revealOnFocus) Sets the preferences of this view for preserving disclosure when it gains focus. final invalidity setRight(int right) Sets the correct position of this view relative to its parent. void setRotation(pivot float) Sets the degrees that the view rotates around the pivot point. void setRotationX(float rotationX) Sets the degree of rotation of the view around the horizontal axis by the pivot point. void setRotationY (float rotationY) Sets the degrees at which the view rotates around the vertical axis through the pivot point. void setSaveEnabled(boolean enabled) Specifies whether saving the state of this view is enabled (that is, whether the onSaveInstanceState(). void setSaveFromParentEnabled(logical value enabled) method specifies whether the entire hierarchy in this view will save its state when its parent state is saved. void setScaleX(float scaleX) the amount that the view scales in x around the pivot point as part of the unscaled view width. View. setScaleY Sets the amount that a view scales in Y around a pivot point as the proportion of the unscaled view width. void setScreenReaderFocusable(boolean screenReaderFocusable) Sets whether this view should be focusable for screen readers and contain non-centered views from the subsea when providing feedback. void setScrollBarDefaultDelayBeforeFade(int scrollBarDefaultDelayBeforeFade) Define the delay before scroll bars fade. void setScrollBarFadeDuration(int scrollBarFadeDuration) Define the fade time of the scroll bar. void setScrollBarSize(int scrollBarSize) Define the size of the scroll bar. void setScrollBarStyle(int style) Specify the style of the scroll bars. void setScrollContainer(boolean isScrollContainer) Change whether this view is one of a set of scrollable containers in its window. void setScrollIndicators(int indicators, int mask) Sets the state of the scroll indicators specified by the mask. void setScrollIndicators(int indicators) Sets the status of all scroll indicators. void setScrollX(int value) Set the horizontal scrolling position of the view. void setScrollY(int value) Set the vertical scrolling position of the view. void setScrollbarFadingEnabled(boolean fadeScrollbars) Define whether scroll bars will fade when the view is not scrolling. void setSelected (logically selected) Changes the selection state of this view. void setSoundEffectsEnabled(boolean soundEffectsEnabled) Set whether this view should have sound effects enabled for events such as clicking and touching. void setStateDescription(CharSequence stateDescription) Sets the description of the view state. void setStateListAnimator(StateListAnimator stateListAnimator) Appends the supplied StateListAnimator to this view. void setSystemGestureExclusionRects(List<Rect> rects) Sets a list of areas in the post-layout coordinate space of this view where the system should not intercept touch gestures or other pointing device gestures. void setSystemUiVisibility(int visibility) This method has been deprecated at API level 30. SystemUiVisibility flags are obsolete. Instead, use the WindowInsetsController. void setTag(int key, Object tag) Sets the tag associated with this view and key. void setTag Sets the tag associated with this view. void setTextAlignment(int textAlignment) Set text alignment. void setTextDirection(int textDirection) Set the direction of the text. void setTooltipText(CharSequence tooltipText) Sets the tooltip text that will be displayed in a small pop-up window next to the view. The final element of the Abyss SetTop(int top) Sets the top position of this view relative to its parent. void setTouchDelegate sets TouchDelegate for this view. void setTransitionAlpha(float alpha) This property intended only for use by the Fade transition, which animates it to produce visual transparency that has no or no effect on the actual alpha </Rect> </Rect> final void setTransitionName(String transitionName) Sets the name of the view to be used to identify views in transitions. void setTransitionVisibility(int visibility) Changes the visibility of this view without triggering any other changes. void setTranslationX(float translationX) Sets the horizontal location of this view relative to its left position. void setTranslationY Sets the vertical location of this view relative to its upper position. void setTranslationZ(float translationZ) Sets the depth position of this view relative to its elevation. void setVerticalFadingEdgeEnabled(boolean verticalFadingEdgeEnabled) Define whether vertical edges should be faded when this view is scrolling vertically. void setVerticalScrollBarEnabled(boolean verticalScrollBarEnabled) Define whether the vertical scroll bar should be drawn or not. void setVerticalScrollbarPosition(int position) Set the position of the vertical scroll bar. void setVerticalScrollbarThumbDrawable(Drawable drawable) Defines a vertical thumb scroll bar for drawing void setVerticalScrollbarTrackDrawable(Drawable drawable) Defines the vertical scroll bar drawable void setVisibility (int visibility) Set the visibility state of this view. void setWillNotCacheDrawing(boolean willNotCacheDrawing) This method has been deprecated in api level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline trimming. For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. void setWillNotDraw(boolean willNotDraw) If this view does not perform any drawing itself, set this flag to allow further optimizations. void setWindowInsetsAnimationCallback(WindowInsetsAnimation.Callback callback) Sets WindowInsetsAnimation.Callback to notify you of window animations that cause snippets. void setX(float x) Sets the visual position of x of this view in pixels. Void y) Sets the visual position of this view in pixels. void setZ(float z) Sets the visual position from this view in pixels. boolean showContextMenu() Shows the context menu for this view. Boolean Boolean x, float y) Shows the context menu for this anchored view to the specified relative coordinate of the view. ActionMode startActionMode(ActionMode.Callback callback, int type) Start action mode with this type. ActionMode startActionMode(ActionMode.Callback callback) Start action mode with the default actionmode type#TYPE_PRIMARY. void startAnimation Run the specified animation now. final boolean startDrag(ClipData data, View.DragShadowBuilder shadowBuilder, Object myLocalState, int flags) This method has been deprecated in api level 24. Use startDragAndDrop() for newer versions of the platform. final boolean startDragAndDrop(ClipData data, View.DragShadowBuilder shadowBuilder, Object myLocalState, int flags) Starts the drag-and-drop operation. logical startNestedScroll(int axis) Start a scroll nesting operation along the specified axes. void stopNestedScroll() Stop the nested scroll in progress. The string toString() returns a representation of the string of the object. void transformMatrixToGlobal(Matrix) Modifies the input matrix so that it maps local coordinates to coordinates on the screen. void transformMatrixToLocal(Matrix Matrix) Modifies the input matrix in such a way that it maps coordinates on the screen to display local coordinates. void unscheduleDrawable (Drawable who, Runnable what) Cancels a scheduled action for a draw. void unscheduleDrawable(Drawable who) Unsealing any events associated with a given Drawable. Final void updateDragShadow(View.DragShadowBuilder shadowBuilder) Updates the drag shadow for an ongoing drag-and-drop operation. Logical verifyDrawable(Drawable who) If the view subclass displays its own drawable objects, replace this function and return true for each Drawable is displayed. boolean willNotCacheDrawing() This method has been deprecated in api level 28. The view drawing cache was largely obsolete with the introduction of hardware-accelerated rendering in API 11. By accelerating the hardware intermediate cache layer, they are largely unnecessary and can easily result in a loss of net performance due to the cost of creating and updating the tier. In rare cases where caching layers are useful, for example for alpha animations, setLayerType(int, android.graphics.Paint) supports this with hardware rendering. For snapshots rendered with a small portion of the view hierarchy or individual views, it is recommended that you create a canvas from a bitmap or image and call drawing (android.graphics.Canvas) in the view. However, these software-rendered uses are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and cropping For UI screenshots for feedback reports or pixelcopy API testing units, we recommend that you. the boolean value willNotDraw() returns a value regardless of whether this view is drawn by itself. From java.lang.Object java.lang.Object class clone() Creates and returns a copy of this object. boolean equals(Object obj) Indicates whether another object is equal to that object. void finalize() Called by the garbage collector on the object when garbage collection specifies that there are no more references to the object. The <> getClass() class returns the runtime class of this object. int hashCode() Returns the hash code value for the object. The string toString() returns a representation of the string of the object. Final invalidity wait(long timeout, int nanos) Causes the current thread to wait until another thread calls the notify() method or notifyAll() method for that object, or another thread aborts the current thread or a certain amount of real time has elapsed. final invalidity wait(long timeout) Causes the current thread to wait until another thread calls the notify() method or notifyAll() method for that object, or the specified amount of time has eladed. the final void wait() causes the current thread to wait until another thread calls the notify() method or notifyAll() method for that object. From the interface android.view.ViewParent abstract void bringChildToFront(See child) Change the order of the child, so it is on top of all other child bundles. the abstract logical file canResolveLayoutDirection() indicates whether this view parent can solve the layout direction. abstract boolean canResolveTextAlignment() Indicates whether this view parent can resolve text alignment. abstract boolean canResolveTextDirection() Indicates whether this view parent can solve the direction of the text. abstract void childDrawableStateChanged(See child element) This method is called on the parent when the child's drawable state has changed. abstract void childHasTransientStateChanged(See child, logical value hasTransientState) Called when the child view has or does not track a transient state. abstract void clearChildFocus(View child) Invoked when a child of that parent opts out of the abstract focus void createContextMenu(ContextMenu menu) To have a parent populate a specific context menu if it has something to add (and then again on its parent). Abstract FocusSearch View(View v, int direction) Find the nearest view in a specific direction that wants to take abstract focus void focusableViewAvailable(View v) Informs the parent that the new focusable view has become available. abstract boolean getChildVisibleRect(View Child, Rect r, Point Offset) Calculates the visible portion of a rectangular region defined in terms of child view coordinates. abstract int getLayoutDirection() Returns this direction of the parent view layout. abstract ViewParent getParent() Returns a parent, if any, or null. abstract ViewParent getParentForAccessibility() Gets parent Availability. abstract int getTextAlignment() Return this view of parent alignment text. abstract int getTextDirection() Returns this parent direction of view text. abstract voidateChild(View Child, Rect r) This method has been deprecated in api level 26. Instead, use onDescendantInvalidated(android.view.View, android.view.View). abstract ViewParent invalidateChildInParent(int[] location, Rect r) This method has been deprecated in api level 26. Instead, use onDescendantInvalidated(android.view.View, android.view.View). abstract logical value isLayoutDirectionResolved() Indicates whether this parent direction of the view layout is resolved. abstract logical file isLayoutRequested() Indicates whether a layout is requested in this parent view. abstract logical value isTextAlignmentResolved() Indicates whether the alignment of the view's parent text is resolved. abstract logical value isTextDirectionResolved() Indicates whether this parent direction of view text is resolved. abstract See keyboardNavigationClusterSearch(See currentCluster, int direction) Find the nearest keyboard navigation cluster in a specific direction. abstract void notifySubtreeAccessibilityStateChanged(See child, See source, int changeType) Notifies the view parent that the availability state of one of its children has changed and that the structure of the subtree is different. the default void onDescendantInvalidated(View Child, Target View) The destination view has been invalidated or the drawing property that requires the hierarchy to be rendered again has been changed. abstract logical onNestedFling(Target view, float speedX, float speedY, logical use) Request throw from nested scroll. abstract logical onNestedPreFling(Target view, float speedX, float speedY) React to nested throwing before the target view consumes it. abstract boolean onNestedPrePerformAccessibilityAction(View target, int action, Package arguments) React to an accessibility action delegated by the destination child view before the target speaks to it. abstract void onNestedPreScroll(View target, int dx, int dy, int[] worn out) React to nested coils in progress before the target view consumes part of the scroll. abstract void onNestedScroll(View target, int dxConsumed, int dyConsumed, int dxUnconsumed, int dyUnconsumed) React to nested scrolls in progress.

abstract void onNestedScrollAccepted(View child, View target, int nestedScrollAxes) Respond to a successful nested scroll operation request. abstract boolean onStartNestedScroll(View Child, Target View, int nestedScrollAxes) Respond to the child view initiating the nesting operation by requesting an operation if necessary Scroll. abstract void onStopNestedScroll(See target) React at the end of the nested scroll operation. abstract void recomputeViewAttributes(View Child) Tell the view hierarchy that global view attributes must be re-evaluated. abstract void requestChildFocus(View child, See depth) depth) when a child of this parent wants to focus an abstract logical requestChildRectangleOnScreen(See child, trough rectangle, immediate logical) invoked when a child of this group wants a specific rectangle to be placed on the screen. abstract void requestDisallowInterceptTouchEvent(boolean disallowIntercept) Called when a child does not want this parent and its ancestors to capture touch events using ViewGroup#onInterceptTouchEvent(MotionEvent). abstract void requestFitSystemWindows() Request a new shipment of View#fitSystemWindows(Rect). abstract void requestLayout() Called when something has changed that invalidated the child layout of this parent view. abstract logical requestSendAccessibilityEvent(View child, AccessibilityEvent event) Raised by the child to request from the parent to send AccessibilityEvent. Abstract void requestTransparentRegion(View Child Element) Called when a child wants the view hierarchy to collect and report transparent regions to the composer window. abstract boolean showContextMenuForChild(View originalView) Shows the context menu for a specific view or its ancestors. abstract boolean showContextMenuForChild(View originalView, float x, float y) Shows the context menu for a specific view or its ancestors anchored to a specific coordinate relative to the view. abstract ActionMode startActionModeForChild(See originalView, ActionMode.Callback callback, int type) Run the action mode of a specific type for the specified view. abstract ActionMode startActionModeForChild(See originalView, ActionMode.Callback callback) Run action mode for a specific view with the default ActionMode#TYPE_PRIMARY. From the interface android.view.KeyEvent.Callback abstract boolean onKeyDown(int keyCode, KeyEvent event) Raised when an event occurred down the key. abstract boolean onKeyLongPress (int keyCode, KeyEvent event) triggered when a long press occurred. Abstract boolean onKeyMultiple (int keyCode, int count, KeyEvent event) Invoked when a user interacts with an analog control, such as throwing a trackball, generates simulated events down/up for the same key multiple times in quick succession. abstract boolean onKeyUp(int keyCode, KeyEvent event) Raised when a key up event occurs. public static final string SCHEME_GEO URI schema for the map address. Fixed value: geo:0,0?q= public static final string SCHEME_MAILTO URI schema for email address. Fixed value: mailto: Public static final string SCHEME_TEL URI schema for phone number. Fixed value: tel: public WebView (context context) Constructs a new WebView with an activity context object. Note: WebView should always be created from context If it occurs with the application context, WebView will not be able to provide several features, such as JavaScript dialogs and AutoComplete. Parameter context: Activity context application resources This value cannot be null. Public WebView (contextual context, AttributeSet attrs) Constructs a new webview with layout parameters. Context parameters context: The context of the activity to access application resources This value cannot be null. attrs AttributeSet: AttributeSet passed to our parent This value can be null. public WebView (contextual context, AttributeSet attrs, int defStyleAttr) Constructs a new webview with layout parameters and default style. Context parameters context: The context of the activity to access application resources This value cannot be null. attrs AttributeSet: AttributeSet passed to our parent This value can be null. defStyleAttr int: An attribute in the current theme that contains a reference to a style resource that provides default values for the view. It may be 0 not to look for default values. public WebView (Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) Constructs a new webview with layout parameters and default style. Context parameters context: The context of the activity to access application resources This value cannot be null. attrs AttributeSet: AttributeSet passed to our parent This value can be null. defStyleAttr int: An attribute in the current theme that contains a reference to a style resource that provides default values for the view. It may be 0 not to look for default values. defStyleRes int: A style resource ID that provides default values for a view that is used only if defStyleAttr is 0 or cannot be found in the theme. It may be 0 not to look for default values. public WebView (Context, AttributeSet attrs, int defStyleAttr, boolean privateBrowsing) This constructor is deprecated. Private browsing is no longer supported directly through the WebView browser and will be removed in a future release. Prefer to use WebSettings, WebViewDatabase, CookieManager, and WebStorage for detailed privacy data control. Creates a new webview with layout parameters and default style. Context parameters context: The context of the activity to access application resources This value cannot be null. attrs AttributeSet: AttributeSet passed to our parent This value can be null. defStyleAttr int: An attribute in the current theme that contains a reference to a style resource that provides default values for the view. It may be 0 not to look for default values. privateBrowsing boolean: Whether this WebView will be initialized in private public void addJavascriptInterface (Object, String Name) Injects the supplied Java object into this WebView. The object is injected into all frames of the web page, including all iframes, using the given name. This allows you to obtain java object methods from JavaScript. For API-level applications, Build.VERSION_CODES. JELLY_BEAN_MR1 above, only public methods that are marked with javascriptInterface are available from JavaScript. For For at the API level Build.VERSION_CODES. JELLY_BEAN or below, all public methods (including inherited methods) can be accessed, see the important security note below for consequences. Note that the injected objects will not be displayed in JavaScript until the page is loaded further (again). JavaScript must be enabled before injecting an object. For example: JsObject { @JavascriptInterface public String toString() { return injectedObject; } } webview.getSettings().setJavaScriptEnabled(true); webView.addJavascriptInterface(new JsObject(), injectedObject); webView.loadData(, text/html, null); webView.loadUrl(javascript:alert(injectedObject.toString())); IMPORTANT: This method can be used to allow javascript to control the host application. This is an advanced feature, but it also poses a security risk to the targeting Build.VERSION_CODES. JELLY_BEAN or earlier. Applications designed for versions later than Build.VERSION_CODES. JELLY_BEAN are still vulnerable if the app runs on an Android device earlier than 4.2. The safest way to use this method is to Build.VERSION_CODES. JELLY_BEAN_MR1 and make sure that the method is called only at startup on Android 4.2 or later. For these older versions, javascript can use reflection to access the public fields of the injected object. Using this method in a WebView containing untrusted content could allow an attacker to manipulate a host application inadvertently by executing Java code with host application permissions. Use extreme care when using this method in a WebView that may contain untrusted content. JavaScript interacts with the Java object in a private thread in the background of this webview. Therefore, care should be taken to maintain thread safety. Because an object is exposed to all frames, each frame can get the name of the object and call methods on it. There is no way to tell the origin of the calling frame from the application side, so the application cannot assume that the caller is trustworthy unless the application can guarantee that no third-party content is ever loaded into the WebView even inside the iframe. Java object fields are not available. For API-level applications, Build.VERSION_CODES. LOLLIPOP and above, the methods of injected Java objects are enumerated from JavaScript. Object object parameters: Inject a Java object into this webview in a JavaScript context. null values are ignored. This value cannot be null. String name: The name used to expose the object in JavaScript This value cannot be null. public void autofill<AutofillValue> Autofill content of virtual minor damage in this view. Views with virtual child environments support the autocomplete structure primarily by: providing metadata that defines the meaning of virtual child costumes and how they are autocompleted. Implement methods that autofill virtual child page. </AutofillValue> </AutofillValue> int) is responsible for the former, this method is responsible for the latter - see autofill(android.view.autofill.AutofillValue) and onProvideAutofillVirtualStructure(android.view.ViewStructure, int) for more information on autocomplete. If the child value is updated asynchronously, the next autocomplete call#notifyValueChanged(View, int, AutofillValue) must occur when you change the value to autocomplete. If not, the child will not be considered autofilled. Note: To indicate that the virtual view has been automatically populated, ? android:attr/autofillHighlight should be drawn above it until the data changes. SparseArray parameter values: A value map for autofill, keyed by a virtual child ID. This value cannot be null. public boolean canGoBack () Gets whether this WebView has a back story element. Returns true logical if this WebView has a back story item, the public logical value canGoBackOrForward (int steps) gets whether the page can go back or forward a given number of steps. Int steps parameters: A negative or positive number of steps to move the history of the public logical value canGoForward () gets whether this WebView has a history element forward. Returns true logical if this webview has a forward history item added at API level 11 Deprecated in the logical level of the public canzoomIn API 17 () This method has been deprecated at api level 17. This method is prone to inaccuracies due to race conditions between web rendering and UI threads; prefer WebViewClient#onScaleChanged. Gets whether this WebView can be enlarged. Returns true logical if this WebView can be zoomed in on api level 11 Deprecated at public level api 17 bępolean canZoomOut () This method has been deprecated at API level 17. This method is prone to inaccuracies due to race conditions between web rendering and UI threads; prefer WebViewClient#onScaleChanged. Gets whether this WebView can be zoomed out. Returns true logical if this WebView can be zoomed out Added horizontally to API 1 Deprecated at the public api image level 19 Image Capture Image () This method has been deprecated at API level 19. Use onDraw(Canvas) to get a snapshot of the webview bitmap, or saveWebArchive(String) to save the content to a file. Gets a new image that captures the current content of this WebView. The image is the entire document displayed and is not limited to the area currently displayed by this WebView. In addition, the image is a static copy and is not affected by subsequent changes to the Content. Note that due to internal changes for API levels between Build.VERSION_CODES. HONEYCOMB AND BUILD.VERSION_CODES. ICE_CREAM_SANDWICH, the image does not contain fixed position elements or scrollable divas. Please note that with Build.VERSION_CODES. JELLY_BEAN_MR1, the returned image should only be pulled into a canvas with a bitmap - using any other type of canvas will require additional in-memory cost conversion and i Returns an image image that captures the current content of this public void void on the Web. Note that the cache is for the application, so this will clear the cache for all webviews used. IncludeDiskFiles boolean parameters: If false, only the RAM cache is cleared public static void clearClientCertPreferences (Runnable onCleared) Clears client certificate preferences stored in response to client certificate requests. Note that WebView automatically clears these preferences when you update the system keychain. Preferences are shared by all webviews that are created by the embed application. OnCleared Runnable parameters: Can be called when client certificates are cleared. Runnable will be called on the UI thread. This value can be null. public void clearFormData () Removes the AutoComplete pop-up from the currently concentrated form field, if present. Note that this only applies to the display of the AutoComplete pop-up, it does not delete any saved form data from the webview store. To do this, use WebViewDatabase#clearFormData. public void clearHistory () Tells this WebView to clear its internal back/forward list. public void clearSslPreferences () Clears the SSL preference table stored in response to SSL certificate errors. Added API Level 1 Deprecated API Level 18 public void clearView () This method has been deprecated at API level 18. Use WebView.loadUrl(about:blank) to reliably reset the view state and free up page resources (including all javascript running). Clears this WebView so that onDraw() will draw only a white background, and onMeasure() will return 0 if MeasureSpec is not MeasureSpec.EXACTLY. public void computeScroll () Called by a parent to request that the child update its values for mScrollX and mScrollY if necessary. This is typically done if the child is animating scrolling using the Scroller object. public WebBackForwardList copyBackForwardList () Gets WebBackForwardList for this WebView. Lists the back/forwards to use in queries for each item in the history stack. This is a copy of the private WebBackForwardList, so it contains only a snapshot of the current state. Multiple calls to this method can return different objects. The object returned from this method will not be updated to reflect the new state. Returns WebBackForwardList This value cannot be null. public PrintDocumentAdapter createPrintDocumentAdapter (String documentName) Creates a PrintDocumentAdapter that exposes the contents of this webview to The tab works by converting WebView content to a PDF stream. WebView cannot be drawn during the conversion process - any such draws are undefined. We recommend that you use a dedicated off-screen WebView for printing. If necessary, the application may temporarily hide the using a custom PrintDocumentAdapter instance wrapped around the returned object and observing the onStart and onFinish methods. For more information, see PrintDocumentAdapter. DocumentName String parameters: The name of the printed document addressed to the user. See PrintDocumentInfo This value cannot be null. Returns PrintDocumentAdapter This value cannot be null. public WebMessagePort[] createWebMessageChannel () Creates a message channel to communicate with JS and returns message ports that represent the endpoints of that message channel. The HTML5 message channel feature is described here when the returned news feeds are tangled and already in a running state. Returns the webmessageport[] of the two message ports that make up the news feed. This value cannot be null. public destruction () Destroys the internal state of this WebView. This method should be called after this WebView has been removed from the view system. No other methods can be called in this WebView after destroy. public static void disableWebView () Indicates that the current process has no intention of using WebView and that an exception should be thrown if a webview is created or other methods are used in the android.webkit package. Applications with multiple processes may want to call this in processes that are not intended to use WebView, to avoid accidentally encountering memory usage when initializing WebView in long-running processes that do not need it, and to prevent potential conflicts in the data directory (see setDatDirectaDirectorySuffix(String)). For example, an audio player application with one process for its activities and another process for its playback service might want to call this method in service.onCreate(). public boolean dispatchKeyEvent (KeyEvent event) Dispatch the key event to the next view on the focus path. This path runs from the top of the view tree to the currently concentrated view. If this view has focus, it will send to itself. Otherwise, it will send the next node down the focus path. This method also runs all key listeners. KeyEvent event parameters: The key event to send. Returns true if the event was handled, false otherwise. public void documentHasImages (Message response) Responds to the document to see if it contains references to the image. The message object will be sent from arg1 is set to 1 if images are found and 0 if the document does not reference any images. Response to message parameters: The message that will be sent with the result of this value cannot be null. public static void enableSlowWholeDocumentDraw () For L-version applications, WebView has a new default behavior that reduces memory consumption and improves performance, by selecting the part of the HTML document that needs to be drawn. These optimizations are transparent to developers. However, as part of the circumstances, the application developer may want to disable them: When an application uses onDraw(Canvas) for its own drawing and accesses the part of the page that is outside the visible part of the page. When an application uses capturePicture() to capture a very large HTML document. Note that capturePicture is an outdated API. Enabling drawing of an entire HTML document has a significant performance cost. This method should be called before any WebViews are created. public void evaluateJavascript Asynchronously evaluates JavaScript in the context of the currently displayed page. If non-null, resultCallback will be called from any result returned from this execution. This method must be called on the UI thread, and a callback will be made to the UI thread. Compatibility note. Applications designed for Build.VERSION_CODES. N or later, the JavaScript state from an empty WebView is no longer preserved in navigation, such as loadUrl(java.lang.String). For example, variables and global functions defined before calling loadUrl(java.lang.String) will not exist on the loaded page. Applications should use addJavascriptInterface(Object, String) instead of persisting JavaScript objects in navigation. String script parameters: JavaScript to execute. This value cannot be null. resultCallback ValueCallback: A callback to be made when the script finishes executing with the result of execution (if any). It can be null if no result notification is required. This value can be null. Added api-level 1 deprecated horizontal API 28 public static string findAddress (String addr) This method has been deprecated at API level 28. This method is replaced by TextClassifier#generateLinks(android.view.textclassifier.TextLinks.Request). Avoid using this method even when targeting API levels where no alternative is available. Gets the first substring, which appears to be the address of the physical location. Only addresses in the United States that must consist of: house number, street type street name (Road, Circle, etc.), either spelled or abbreviated city name state or territory, or written or two-letter abbr optional 5-digit or 9-digit zip code All names must be spelled correctly in capital letters, and zip code, if present, must be valid for the state. The street type must be standard spelling or USPS abbreviation. The state or territory must also be written or abbreviated using USPS standards. The house number must not exceed five digits. Note: This feature is obsolete and should be avoided at all API levels because it cannot detect outside the United States and has a high false alarm rate. at the API level Build.VERSION_CODES. O_MR1 and earlier also causes the entire WebView implementation to load and initialize, which can throw AndroidRuntimeException or other </String> </String> exceptions WebView implementation is currently being updated. String addr parameters: The address search string returns an address string, or if no address is found, null public void findAllAsync (String find) searches for all instances found on the page and highlights them asynchronously. It notifies the registered FindListener. Subsequent calls to this cancel any pending searches. Parameters find string: string to find. This value cannot be null. See also: setFindListener(WebView.FindListener) public View findFocus () Find a view in a hierarchy rooted in that view that currently has focus. Returns a View view that currently has focus or null if the focus view cannot be found. public void flingScroll (int vx, int vy) Added api level 7 Deprecated in api level 19 public void voidMemory () This method has been deprecated at API level 19. Caches are automatically discarded when they are no longer needed, and in response to system memory pressure. This WebView informs you that the memory is low so that you can free up any available memory. public CharSequence getAccessibilityClassName () Returns the class name of this object to be used for accessibility purposes. Subclasses should replace this only if they implement something that should be seen as a brand new view class when used by availability, unrelated to the class from which it originates. Used to populate AccessibilityNodeInfo#setClassName. public SslCertificate getCertificate () Gets an SSL certificate for the top-level or null main page if it does not have a certificate (the site is not secure). Returns the SSlCertificate certificate for the top-level public top-level page int getContentHeight () Gets the height of the HTML content. Returns int the height of the HTML content of the public static PackageInfo getCurrentWebViewPackage () If the WebView has already been loaded into the current process, this method will return the package that was used to load it. Otherwise, the package that will be used if the WebView has been loaded now will be returned; this does not cause the WebView to load, so this information may become out of date at any time. The WebView package changes when you update, disable, or uninstall the current WebView package. You can also change it by using the developer setting. If the WebView package changes, any application process that has loaded the WebView will be killed. The next time you start the application and load the webview, it will use the new WebView package instead. Returns packageinfo of the current WebView or null package if it is not present. public Bitmap getFavicon () Gets favicon for the current page. This is the favicon of the current page until WebViewClient.onReceivedIcon is called. Returns the favicon bitmap for the current or null if the page is gone or if no page has been loaded, the public handler getHandler () returns the handler handler associated with the thread in the view. This handler can be used to pump events in the UI event queue. public WebView.HitTestResult getHitTestResult () Gets HitTestResult from the current cursor node. If an HTML::a tag is found and the anchor has a non-javascript URL, the HitTestResult type is set to SRC_ANCHOR_TYPE and the URL is set in the extra field. If the anchor does not have a URL or is a JavaScript URL, the type UNKNOWN_TYPE be removed and the URL must be retrieved by asynchronously requesting FocusNodeHref(Message). If the HTML::img tag is found, the HitTestResult type is set to IMAGE_TYPE and the URL is set in the extra field. The SRC_IMAGE_ANCHOR_TYPE indicates an anchor with a URL that has the image as a child node. If a phone number is found, hittestresult type is set to PHONE_TYPE and the phone number is set in the extra HitTestResult field. If a map address is found, hittestresult type is set to GEO_TYPE and the address is set in the extra HitTestResult field. If an email address is found, hittestresult type is set to EMAIL_TYPE and the email is set in the extra HitTestResult field. Otherwise, the HitTestResult type is set to UNKNOWN_TYPE. Returns WebView.HitTestResult This value cannot be null. Added API Level 1 Deprecated API Level 26 public String[] getHttpAuthUsernamePassword (String Host, String Sphere) This method has been deprecated at API level 26. Instead, use webviewdatabase#getHttpAuthUsernamePassword to retrieve HTTP authentication credentials for that host and area from the WebViewDatabase instance. String host parameters: A host to which credentials will apply a sphere string: the edge to which credentials will apply the string returns[] credentials as a string array, if found. The first element is the user name, and the second element is the password. null if no credentials are found. public String getOriginalUrl () Gets the original URL for the current page. It's not always the same as the URL uploaded to WebViewClient.onPageStarted, because even though the URL has started loading, the current page may not change. Additionally, you may be redirecting as a result of another URL to be originally required. Returns the URL string that was originally requested for the current page, or null if no page has been loaded public int getProgress () Gets progress for the current page. Returns the int progress of the current page between 0 and 100 public static Uri getSafeBrowsingPrivacyPolicyUrl () Returns a URL indicating a privacy policy for safe browsing reporting. Returns a URL that indicates a privacy policy document that can be Users. This value cannot be null. Added api level 1 deprecated in api level 17 public float getScale () This method has been deprecated at API level 17. This method is prone to inaccuracies due to race conditions between web rendering and UI threads; They prefer Gets the current scale of this WebView. Returns the float of the current public scale WebSettings getSettings () Gets the WebSettings object used to control the settings for this WebView. Returns a WebSettings WebSettings object that can be used to control the settings of this WebView This value cannot be null. public String getTitle () Gets the title for the current page. This is the title of the current page until the WebViewClient.onReceivedTitle page is called. Returns the title string of the current page, or null if no page has loaded the public getUrl string () Gets the URL for the current page. It's not always the same as the URL uploaded to WebViewClient.onPageStarted, because even though the URL has started loading, the current page may not change. Returns a URL string for the current page or null if no page has been loaded public static ClassLoader getWebViewClassLoader () Returns the ClassLoader used to load internal WebView classes. If this method is intended to be used by the WebView support library, there is no reason to use this method otherwise. Returns ClassLoader This value cannot be null. public Looper getWebViewLooper () Returns the Looper corresponding to the thread on which the webview calls must be made. Looper returns This value cannot be null. public WebViewRenderProcess getWebViewRenderProcess () Gets the access to the WebView rendering process associated with this WebView. In the Build.VERSION_CODES, the O and above WebView can run in multiprocess mode. In multiprocess mode, web content rendering is performed by a sandboxed rendering process that is separate from the application process. This rendering process can be shared with other WebViews in the application, but is not shared with other application processes. If the webview is running in multiprocess mode, this method returns an approach to the rendering process associated with the WebView, which can be used to control the rendering process. public void goBack () Goes back to the history of this WebView. public void goBackOrForward (int steps) Moves to the history element, which is the number of steps from the current element. The steps are negative if backward and positive if forward. Int step parameters: The number of steps to take back or forward in the public void list of goForward () Goes forward in the history of this WebView. public void invokeZoomPicker () Calls the graphical zoom selector widget for this WebView. This displays the zoom widget on the screen to control the zoom level of this WebView. public boolean isPrivateBrowsingEnabled () Gets whether private browsing is enabled in this WebView. public boolean isVisibleToUserForAutofill (int virtualId) Calculates whether this virtual autocomplete is visible to the user. Note: By default, it returns true, but views that provide a view of the virtual hierarchy must override it. Returns the logical value whether the view is visible on the screen. public nullity (String data, MimeType strings, string encoding) Loads the given data into this webview using the data schema URL. Note that the same javascript origin rule means that a script running on a page loaded using this method will not be able to access content loaded using any schema other than data, including http(s). To avoid this limitation, use loadDataWithBaseURL() with the appropriate primary URL. The encoding parameter specifies whether the data is encoded as base64 or URL. If the data is base64 encoded, the encoding parameter value must be base64. HTML can be encoded using Base64.encodeToString(byte[], int) so: UnencodedHtml = <html><body>'%28' is code '('</body></html>; Ciąg encodedHtml = Base64.encodeToString(unencodedHtml.getBytes(), Base64.NO_PADDING); webView.loadData(encodedHtml, text/html, base64); For all other encoding values (including null values), the data is assumed to use ASCII encoding for octets inside the secure URL range and use standard hex encoding %xx for octets outside that range. See RFC 3986 for more information. Applications designed for Build.VERSION_CODES. Q or later must use base64 or encode any # characters in the content as %23, otherwise they will be treated as the end of the content, and the remaining text used as the identifier of the document fragment. The mimeType parameter specifies the format of the data. If webview cannot handle a specific type of MIME, it will retrieve the data. If null, text/html by default. The data schema URL created by this method uses the default US-ASCII character set. If you want to set a different character set, create a data schema URL that explicitly specifies the charset parameter in the mediatype portion of the URL and calls loadUrl(java.lang.String). Note that the character set obtained from the mediatype portion of the data URL always overrides that specified in the HTML or XML document. Content loaded using this method will be window.origin null. This should not be considered a trusted source by the application or by any JavaScript code running inside the WebView (for example, event sources in DOM event handlers or Internet messages), because malicious content can also create frames of zero origin. If you want to identify the origin of the main frame in a trustworthy way, use loadDataWithBaseURL() with a valid PRIMARY HTTP or HTTPS URL to set the origin. Data parameters String: Data string in a given encoding This value cannot be null. mimeType String: a type of MIME data, e.g. This value can be null. encoding string: Encoding data This value can be null. public void loadDataWithBaseURL baseUrl, String data, String mimeType, String encoding, String historyUrl) Loads the given data into this webview, using baseUrl as the primary URL for the content. Primary URL used both to recognize relative URLs and to apply the same JavaScript origin policy. HistoryUrl is used for the history entry. The mimeType parameter specifies the format of the data. If webview cannot handle a specific type of MIME, it will retrieve the data. If null, text/html by default. Note that content specified in this way can access local device files (through file schema URLs) only if baseUrl specifies a schema other than http, https, ftp, ftps, about, or javascript. If the primary URL uses a data schema, this method is equivalent to calling loadData() and historyUrl is ignored, and the data will be treated as part of the data: the URL, including the requirement that the content be encoded in the URL or encoded with base64. If the primary URL uses a different schema, the data will be loaded into the webview as a regular string (i.e. it is not part of the data URL), and any eneduct encoded in the URL in the string will not be decoded. Note that baseUrl is sent in the HTTP Referer header when subresources (images, etc.) request a page loaded using this method. If a valid primary HTTP or HTTPS URL is not specified in baseUrl, the content loaded with this method will be window.origin null. This should not be considered a trusted source by the application or by any JavaScript code running inside the WebView (for example, event sources in DOM event handlers or Internet messages), because malicious content can also create frames of zero origin. If you want to identify the origin of the main frame in a trustworthy way, use a valid primary HTTP or HTTPS URL to set the origin. BaseUrl String parameters: The URL to use as the primary URL of the page. If null by default, it is about:blank. This value can be null. data string: Data string in a given encoding This value cannot be null. mimeType String: a type of MIME data, e.g. This value can be null. encoding string: Encoding data This value can be null. historyUrl String: The URL to use as a history entry. If null by default, it is about:blank. If null is not null, it must be a valid URL. This value can be null. public void loadUrl (string URL, Map<String, String> SecondaryHttpHeaders) Loads a given URL with specific additional HTTP headers. See also compatibility note on evaluateJavascript(String, ValueCallback). String url parameters: Resource URL to load This value cannot be null. additionalHttpHeaders Map: Additional headers to be used in the HTTP request for this URL, specified as a name-to-value map. It is necessary to that if this map contains one of the headers that are set by default by this WebView, such as those controlling caching, accept types, or user-agent, their values can be overridden by the default settings of that WebView. This value cannot be null. public boolean onCheckIsTextEditor () Verify that the view you are calling is a text editor, in </String, String> String> it would make sense to automatically display a soft input window for it. Subclasses should override this if they implement onCreateInputConnection(android.view.inputmethod.EditorInfo) to return true if calling this method returns a non-null InputConnection and they are really a first-class editor that the user would normally start typing on after going to the window containing the view. The default implementation always returns false. This does not mean that its onCreateInputConnection(android.view.inputmethod.EditorInfo) will not be called or the user cannot otherwise make changes to the view; this is just a hint to the system that this is not the main purpose of this view. Returns a boolean value that returns true if this view is a text editor, otherwise false. public void onChildViewAdded This method is deprecated. WebView no longer needs to implement ViewGroup.OnHierarchyChangeListener. This method does nothing now. Called when a new child is added to the parent view. Parent view parameters: The view in which the child was added child View: A new child view added in the public hierarchy void onChildViewRemoved (View p, Subview) This method is deprecated. WebView no longer needs to implement ViewGroup.OnHierarchyChangeListener. This method does nothing now. Called when a child is removed from the parent view. Parameters p View: The view from which the child item was removed child View: A child removed from the public InputConnection onCreateInputConnection (EditorInfo outAttrs) hierarchy creates a new inputconnection for InputMethod to interact with the WebView. This is similar to View#onCreateInputConnection, but be aware that the WebView calls the InputConnection method on a thread other than the UI thread. If these methods are overrided, then parent methods should follow thread constraints when calling the View method or accessing data. OutAttrs EditorInfo parameters: Fill in the connection attribute information. public boolean onDragEvent (DragEvent event) Supports drag events sent by the system when startDragAndDrop() is called. When the system calls this method, it passes the DragEvent object. The call to dragevent.getAction() returns one of the action type constants defined in dragevent. The Method uses them to determine what happens in a drag-and-drop operation. DragEvent event parameters: DragEvent sent by the system. DragEvent.getAction() returns a constant of the action type defined in dragevent, indicating the type of drag event represented by this object. Returns true logical if the method succeeded, otherwise false. The method should return true in response to the type DragEvent.ACTION_DRAG_STARTED receive drag events for the current operation. The method should also return true in response to the type of DragEvent.ACTION_DROP if the drop is consumed, or false false Not. For all other events, the return is ignored. public boolean onGenericMotionEvent Implement this method to handle general motion events. General motion events describe joystick movements, mouse rudders, touching crawler pads, scroll wheel movements, and other input events. MotionEvent#getSource() motion event specifies the input class that was received. Implementations of this method must examine the bits in the source before processing the event. The following code example shows how to do this. General traffic events from the InputDevice#SOURCE_CLASS_POINTER are delivered to the view below the pointer. All other general traffic events are delivered to the focused view. public boolean onGenericMotionEvent(MotionEvent event) { if (event.isFromSource(InputDevice.SOURCE_CLASS_JOYSTICK)) { if (event.getAction() == MotionEvent.ACTION_MOVE) { // process the joystick movement... true phrase; } if (event.isFromSource(InputDevice.SOURCE_CLASS_POINTER)) { switch (event.getAction()) { case MotionEvent.ACTION_HOVER_MOVE: // process the mouse hover movement... true phrase; MotionEvent.ACTION_SCROLL case: // processes the movement of the scroll wheel... true phrase; } return super.onGenericMotionEvent(event); } MotionEvent event parameters: A general motion event is processed. Returns true if the event was handled, false otherwise. public void onGlobalFocusChanged (See oldFocus, See newFocus) This method is obsolete. WebView should not have implemented ViewTreeObserver.OnGlobalFocusChangeListener. This method does nothing now. The callback method to be called when the focus changes in the view tree. When the view tree goes from touch mode to non-touch mode, oldFocus is null. When the view tree goes from non-contact mode to touch mode, newFocus is null. When focus changes in non-contact mode (without switching from or to touch mode), old Focus or newFocus can be null. OldFocus View parameters: Previously concentrated view, if any. newFocus View: A newly focused view, if any. public boolean onHoverEvent (MotionEvent event) Implement this method of handling activation events. This method is called each time the pointer is hovering over, over, or out of view boundaries and the view is not currently touched. Hover events are represented as pointer events from the MotionEvent#ACTION_HOVER_ENTER, MotionEvent#ACTION_HOVER_MOVE, or MotionEvent#ACTION_HOVER_EXIT. The view receives an activation event from the MotionEvent#ACTION_HOVER_ENTER when the pointer enters the view boundaries. The view receives an activation event from the MotionEvent#ACTION_HOVER_MOVE action when the pointer has already entered the view boundaries and has been moved. View receives event motionevent#ACTION_HOVER_EXIT when the pointer has left the view boundaries or when the pointer is about to go because of a button click, press, or similar user action that touches the view. The view must implement this method to return true to indicate that it supports a hover event, such as changing its drawable state. Default calls to the setHovered(boolean) implementation to update the view mouse state when an input or cursor activation event is received if the view is enabled and can be clicked. The default implementation also sends cursor accessibility events. MotionEvent parameters: A motion event that describes the cursor. Returns true if the view handled the activation event. public boolean onKeyDown (int keyCode, KeyEvent event) Default keyevent.Callback#onKeyDown(int, KeyEvent): Tap the view when you release KeyEvent#KEYCODE_DPAD_CENTER or KeyEvent#KEYCODE_ENTER if the view is enabled and clickable. Keystrokes in software keyboards generally do not trigger this listener, although some may choose to do so in some situations. Do not rely on it to catch software keystrokes. KeyCode int parameters: The key code that represents the button press, with the KeyEvent KeyEvent: KeyEvent object that defines the button action returns a logical value if the event is handled, returns true. If you want to allow the next recipient to handle the event, it returns false. public boolean onKeyMultiple (int keyCode, int repeatCount, KeyEvent event) Default implementation keyEvent.Callback#onKeyMultiple(int, int, KeyEvent): always returns false (does not handle events). Keystrokes in software keyboards generally do not trigger this listener, although some may choose to do so in some situations. Do not rely on it to catch software keystrokes. KeyCode int parameters: The key code that represents the button press, with KeyEvent. repeatCount int: The number of times the action was performed. KeyEvent event: The KeyEvent object that defines the button action. Returns a boolean value if the event is handled, returns true. If you want to allow the next recipient to handle the event, it returns false. public boolean onKeyUp (int keyCode, KeyEvent event) Default implementation of KeyEvent.Callback#onKeyUp(int, KeyEvent): Click the view when keyevent#KEYCODE_DPAD_CENTER, KeyEvent#KEYCODE_ENTER, or KeyEvent#KEYCODE_SPACE is released. Keystrokes in software keyboards generally do not trigger this listener, although some may choose to do so in some situations. Do not rely on it to catch software keystrokes. KeyCode int parameters: The key code that represents the button press, with KeyEvent. KeyEvent event: The KeyEvent object that defines the button action. Returns if the event is handled, returns true. If you want to allow the next recipient to handle the event, it returns false. public void onPause () Do your best to pause any processing that can be safely paused, paused, geolocation. Note that this call does not pause JavaScript. To pause JavaScript globally, use pauseTimers(). To resume webview, call onResume(). void onProvideAutofillVirtualStructure Populates the view structure that contains virtual children to make a fullfil autocomplete request. This method should be used when the view manages the virtual structure in that view. For example, a view that draws input fields using draw(android.graphics.Canvas). When implementing this method, subclasses must follow the following rules: Add virtual children by calling ViewStructure#newChild(int) or ViewStructure#asyncNewChild(int), where the identifier is a unique identifier that identifies children in the virtual structure. The hierarchy of history can have multiple levels, if necessary, but it is best to exclude intermediate levels that are not relevant to autocomplete; that would improve autocomplete performance. Also implement autofill (android.util.SparseArray) to autofill virtual children. Set the child structure autocomplete properties defined by onProvideAutofillStructure(android.view.ViewStructure, int), using ViewStructure#setAutofillId(AutofillId, int) to set its AutoComplete ID. Replace isVisibleToUserForAutofill(int) to allow the platform to query whether a given virtual view is visible to the user to handle triggering save when all views of interest go away. Call AutofillManager.notifyValueChanged(View, int, AutofillValue) when the virtual child value changes. Call AutofillManager.notifyViewVisibilityChanged(View, int, boolean) after changing the visibility of the virtual child. Call AutofillManager.notifyViewClicked(View, int) when you click a virtual child. Call AutofillManager#commit() when the autocomplete context of the view structure is changed and the current context should be approved (for example, when a user taps submit on an HTML page). Call AutofillManager#cancel() when the view structure autocomplete context is changed and the current context should be canceled (for example, when a user taps CANCEL on an HTML page). Provide users with ways to manually request automatic completion by calling autofillManager#requestAutofill(View, int, Rect). The left and highest values set in view ViewStructure#setDimens(int, int, int, int, int, int) must be relative to the next view of viewgroup#isImportantForAutofill() contained in the structure. Views with virtual child environments support the autocomplete structure primarily by: providing metadata that defines the meaning of virtual child costumes and how autocomplete. Implement methods that autofill virtual child page. This method is responsible for the former; autofill (android.util.SparseArray) is responsible for Last. ViewStructure traditionally represents a view, while for web pages it represents HTML nodes. Therefore, it is necessary to map the HTML property in a way that autofillService implementations understand: If the WebView implementation can determine that the field value has been statically set (for example, not via javascript), you must also call structure.setDataIsSensitive(false). For example, an HTML form with 2 fields for a user name and password: <label>Username:</label><input type=text name=username id=user value=Type your username autocomplete=username placeholder=Email or username><label>Password:</label><input type=password name=password id=pass autocomplete=current-password placeholder=Password> It will map to: int index = structure.addChildCount(2); ViewStructure username = structure.newChild(index); username.setAutofillId(structure.getAutofillId(), 1); id 1 - first child username.setAutofillHints(username); username.setHtmlInfo(username.newHtmlInfoBuilder(input) .addAttribute(type, text) .addAttribute(name, username) .addAttribute(label, Username:) . build()); username.setHint(E-mail or username); nazwa_użytkownika.setAutofillType(View.AUTOFILL_TYPE_TEXT); username.setAutofillValue(AutofillValue.forText)); The value of the field is not sensitive because it was statically created and has not been changed. nazwa_użytkownika.setDataIsSensitive(false); ViewStructure password = structure.newChild(index + 1); username.setAutofillId(structure, 2); id 2 - second child password.setAutofillHints(current password); password.setHtmlInfo(password.newHtmlInfoBuilder(input) .addAttribute(type, password) .addAttribute(name, password) .addAttribute(label, Password:) . build()); password.setHint(Password); setAutofillType(View.AUTOFILL_TYPE_TEXT); ViewStructure parameter structure: Fill in virtual child data for autocomplete purposes. int flags: optional flags. void onProvideContentCaptureStructure fills the view structure for capturing content. This method is called after a view that qualifies for content capture (for example, if isImportantForAutofill(), the analytics service is enabled for the user, and the view rendering activity is enabled for content capture) is specified and is visible. The populated structure is then passed to the service via ContentCaptureSession#notifyViewAppeared(View Structure). Note: Views that manage the virtual structure in this view must populate only the node representing that view and return immediately, and then asynchronously report (not necessarily in the UI thread) when child nodes appear, disappear, or change text by calling ContentCaptureSession#notifyViewSappeared(AutofillId) in i CharSequence). The structure for the child must be created using ContentCaptureSession#newVirtualViewStructure(AutofillId, long), and autofillId for the child can be obtained through childStructure.getAutofillId() or ContentCaptureSession#newAutofillId(AutofillId, long). When a virtual view hierarchy represents a web page, you must also: Note: The following structure methods will be ignored: Structure views structure parameters: This value cannot be null. int public void flags onProvideVirtualStructure (Viewstructure) Called when a help structure is retrieved from the view as part of activity.onProvideAssistData to generate an additional virtual structure in that view. The defaullt implementation uses getAccessibilityNodeProvider() to try to generate this from virtual view accessibility nodes, if any. You can override this for a more optimal implementation provided you have this data. ViewStructure public boolean onTouchEvent Implement this method to handle touch screen motion events. If this method is used to detect click actions, it is recommended that actions be performed by implementing and calling performClick(). This will ensure consistent system behavior, including: MotionEvent: Motion Event Parameters. Returns true if the event was handled, false otherwise. public boolean onTrackballEvent Implement this method to handle trackball motion events. The relative trackball movement since the last event can be downloaded from MotionEvent#getX and MotionEvent#getY. They are normalized so that motion 1 corresponds to the user pressing one DPAD key (so they will often be fractional values, representing more detailed motion information available from the trackball). MotionEvent Event Parameters: Motion Event. Returns true if the event was handled, false otherwise. public void onWindowFocusChanged (boolean hasWindowFocus) Called when a window containing this view gains or loses focus. Note that this is separate from view focus: To receive key events, both the view and its window must have focus. If there is a window at the top that requires input focus, your own window will lose focus, but the view focus will remain unchanged. HasWindowFocus boolean: True If the window containing this view now has focus, false otherwise. Added horizontal API 1 Deprecated horizontal API 23 public logical layerHorizontalScrollbar () This method has been deprecated at API level 23. This method is now obsolete. Gets whether the horizontal scroll bar has an overlay style. Added horizontal API 1 deprecated horizontal API 23 public logical layersVerticalScrollbar () This method deprecated at api level 23. This method is now obsolete. Specifies whether the vertical scroll bar has an overlay style. public puzzle pageDown LogicalDown at the bottom) Scrolls the content of this webview down by half the page size. Bottom boolean: true to move to the bottom of the page Returns a true logical value if the page has scrolled public boolean pageUp (logical top) Scrolls the contents of this WebView up by half the size of the view. The top boolean: true parameters to go to the top of the page returns a true logical value if the page has scrolled public void pauseTimers () Pausing the entire layout, parsing, and JavaScript timers for all WebViews. This is a global request, not limited to just this WebView. This can be useful if the application has been paused. public boolean performLongClick () Calls this View OnLongClickListener, if defined. Calls the context menu if OnLongClickListener does not consume the event. Returns true logical if one of the above listeners used events, false otherwise public void postUrl (string url, byte[] postData) loads the URL from postData using the POST method to this WebView. If the URL is not a network URL, it will load from

loadUrl(java.lang.String), ignoring the postData param. String url parameters: Resource URL to load This value cannot be null. postData byte: The data will be forwarded to the POST request, which must be encoded with application/x-www-form-urlencoded. This value cannot be null. public void postVisualStateCallback (long requestId, WebView.VisualStateCallback callback) Posts VisualStateCallback, which will be called when the current WebView state is ready to be drawn. Because house updates are processed asynchronously, home updates cannot immediately be visually reflected by subsequent WebView#onDraw calls. VisualStateCallback provides a mechanism for notifying the caller when the DOM content at the moment is ready to be drawn the next time the WebView draws. The next draw after the callback is complete is guaranteed to reflect all home updates to the point where VisualStateCallback was published, but may also include updates applied after the callback was published. The state of the DOM covered by this API includes the following: primitive HTML elements (div, img, span, etc.) CSS images webgl canvas animations do not contain a state: To ensure that WebView successfully renders the first frame after the VisualStateCallback#onComplete method has been called, a set of conditions must be met: When using this API, we also recommend that you enable pre-rasterization if WebView is off screen to avoid flickering. See WebSettings#setOffscreenPreRaster for more details and to consider its caveats. RequestId long parameters: Id that will be returned in the callback to allow matching requests with callbacks. WebView.VisualStateCallback: Callback. This value cannot be null. public void postWebMessage (WebMessage, Uri Uri Publish the message to the main frame. An embedded application can limit messages to a specific target source. See the HTML5 specification to learn how to use the target source. You can set the destination source as a wildcard (*). However, this is not recommended. See the page above for security issues. Content loaded via loadData(java.lang.String, java.lang.String, java.lang.String) will not have a valid origin, so you will not be able to send messages securely. If you want to send messages using this function, use loadDataWithBaseURL(java.lang.String, java.lang.String, java.lang.String, java.lang.String) with valid http or HTTPS baseUrl to define the correct origin that can be used to transmit messages. WebMessage Parameters: WebMessage This value cannot be null. targetOrigin Uri: the target source. This value cannot be null. public reload () Reloads the current URL. public void removeJavascriptInterface Removes a previously injected Java object from this webview. Note that the deletion will not be reflected in JavaScript until the page is loaded further (again). See addJavascriptInterface(Object, String). String parameter name: Name used to expose an object in JavaScript This value cannot be null. Public Logical RequestChildRectangleOnScreen (View Child, Rect Rect, Logical Instant) Raised when the child of this group wants the specified rectangle to be placed on the screen. Parent ViewGroups can trust that: the child will be the direct child of this group rectangle going to the child coordinates of the parent ViewGroups content that should maintain the agreement: nothing will change if the rectangle is already visible the view port will scroll only enough to make the rectangle visible child parameters View: Direct child who made the request. Rect Rect: A rectangle in the coordinates of the child the child wants to be on the screen. instant boolean: True to prohibit animated or delayed scrolling, false otherwise logical returns whether the group scrolls to handle the public operation of the logical requestFocus (int direction, Rect previouslyFocusedRect) call this to try to focus on a specific view or to one of its children and give it direction guidance and the specified rectangle that the focus comes from. A rectangle can help you give larger views a finer grainy hint about where the focus comes from, and therefore where to show the selection or forward focus changes internally. The view will not actually focus if it is not focusable (isFocusable() returns false), or if it is focused and not focused in (isFocusableInTouchMode()) when the device is in touch mode. The view will not focus if it is not visible. The view will not focus if one of its parents has ViewGroup.getDescendantFocusability() equal to ViewGroup #FOCUS_BLOCK_DESCENDANTS. See also focusSearch(int), which is Call to say you have focus, and you want your parent to look for the next one. You can override this method if the custom view has an internal view that you want to submit the request to. Searches for a view to focus on following the setting specified by getDescendantFocusability(). It uses RequestFocusInDescendants(int, android.graphics.Rect) to find the focus in the nineteen nanny. Int Direction: One of FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT, and FOCUS_RIGHT previouslyFocusedRect Rect: Rectangle (including coordinate system view) to give a finer grainy hint about where the focus comes from. It can be null if there are no hints. Returns the logical value of whether this view or one of its children actually took focus. public void requestFocusNodeHref (Message hrefMsg) Requests the URL of the anchor element or image element at the last tap point. If hrefMsg is null, this method returns immediately and does not send hrefMsg to its target. If a tap point hits an image, anchor, or anchor image, the message associates strings in named keys in its data. The value paired with the key can be an empty string. HrefMsg Message parameters: The message to be sent with the result of the request. The message data contains three keys. url returns the anchor href attribute. title returns the anchor text. src returns the src attribute of the image. This value can be null. public void requestImageRef (Message msg) Requests the URL of the image last touched by the user. msg will be sent to the target from String representing the URL as its object. Msg Message parameters: A message to be called with the result of a request as a data member with url as a key. The result can be null. This value cannot be null. public WebBackForwardList restoreState (Bundle inState) Restores the state of this webview from a given package. This method is intended to be used in activity.onRestoreInstanceState(Bundle) and should be called to restore this webview state. If it is called after that WebView has had a chance to build a state (load pages, create a back/forward list, etc.) you may experience unwanted side effects. Note that this method no longer restores the data displayed for this webview. Parameters in State Bundle: The incoming bundle status of this value cannot be null. public void resumeTimers () Resumes all layout, analysis, and JavaScript timers for all webviews. This resumes shipping of all timers. Added horizontal API 1 Deprecated horizontal API 18 public void savePassword (Host string, user name string, password string) This method has been deprecated at API level 18. Saving passwords in WebView will not be supported in future releases. Sets the user name and password pair for the Host. This data is used by webview to autocomplete user name and password fields in web forms. Please note that this is a credentials used for HTTP authentication. String host parameters: A host that requires string user name credentials: The user name for a given String host password: The password for a given public void saveWebArchive (String filename) host saves the current view as a web archive. File name parameters String: The file name in which the archive should be placed This value cannot be null. public void saveWebArchive (String basename, boolean autoname, ValueCallback<String> callback) Saves the current view as an internet archive. Basename String: The file name in which you should place the archive This value cannot be null. autoname boolean: if false, it has basename to be a file. If true, basename assumes that the directory where the file name will be selected according to the URL of the current page. Callback ValueCallback: called after saving the internet archive. The onReceiveValue parameter will be the name of the file under which the file was saved, or null if the file failed to save. This value can be null. public void setBackgroundColor (int color) Sets the background color for this view. Color int parameters: Background color Added horizontal API 1 Deprecated at the public level void setCertificate (SslCertificate certificate) This method has been deprecated at api level 17. Calling this function has no useful effect and will be ignored in future releases. Sets the SSL certificate for the top-level home page. SslCertificate public static void setDataDirectorySuffix (String suffix) Certificate Define the directory used to store WebView data for the current process. The specified suffix will be used when constructing data directory paths and cache. If this API is not called, the suffix will not be used. Each directory can be used by only one process in the application. If more than one process in an application wants to use WebView, only one process can use the default directory, and other processes must call this API to define a unique suffix. This means that different processes in the same application cannot directly share data related to the webview because the data directories must be different. Applications that use this API can explicitly send data between processes. For example, login cookies can be copied from one cookie jar process to another using CookieManager if both WebViews processes are designed to log on. Most applications simply need to make sure that all application components that rely on WebView are in the same process to avoid having multiple data directories. The disableWebView() method can be used to ensure that other processes do not use WebView by accident in this case. This The API must be called before any webview instances are created in this process, and before other methods in the android.webkit package are called by this process. Suffix String Parameters: Directory Name </String> </String> to be used in the current process. It cannot contain a path separator. This value cannot be null. public void setDownloadListener (DownloadListener listener) Registers the interface to be used when the content cannot be supported by the rendering engine and should be downloaded. This overwrites the current handler. Listener DownloadListener parameters: DownloadListener implementation This value can be null. public void setFindListener (WebView.FindListener listener) Registers the listener to receive notifications as the find-on-page operation progresses. This overwrites the current listener. Listener WebView.FindListener: FindListener implementation This value can be null. Added horizontal API 1 Deprecated horizontally 23 public void setHorizontalScrollbarOverlay (logical overlay) This method has been deprecated at api level 23. This method has no effect. Specifies whether the horizontal scroll bar has an overlay style. Boolean overlay parameters: true if horizontal scrollbar should have overlay style Added in API level 1 Deprecated in API level 26 public void setHttpAuthUsernamePassword (String host, String realm, String username, String password). Instead, use webviewdatabase#setHttpAuthUsernamePassword to store HTTP authentication credentials for the host and area in the WebViewDatabase instance. String host parameters: Host to which string: land will apply credentials: String user name password: String user name password: void setInitialScale public password (int scaleInPercent) Sets the initial scale for this webview. 0 is the default value. The behavior of the default scale depends on the state of WebSettings#getUseWideViewPort() and WebSettings#getLoadWithOverviewMode(). If the content matches the WebView by width, the magnification is set to 100%. For broad content, the behavior depends on the state of WebSettings#getLoadWithOverviewMode(). If its value is true, the content will be enlarged to fit by width to the WebView control, otherwise it will not. If the initial scale is greater than 0, the WebView starts with this value as the initial scale. Note that unlike the scale properties in the viewport meta tag, this method does not take into account screen density. ScaleInPercent int: Initial scale as a percentage of public void setLayoutParams (ViewGroup.LayoutParams params) Set the layout parameters associated with this view. These power parameters to the parent of this view determine how it is arranged. There are many subclasses of ViewGroup.LayoutParams, and these correspond to the different subclasses of the Viewgroup group that are responsible for organizing their injuries Params ViewGroup.LayoutParams: Layout parameters for this view cannot be null Added api level 1 Deprecated in api level 17 void public public (logical setMap) This method has been deprecated at api level 17. Only the default case, true, will be supported in a future release. The setMap boolean public void setNetworkAvailable (boolean networkUp) parameter informs webview of the network state. Set the JavaScript property to window.navigator.isOnline and generate the online/offline event specified in HTML5, sec. 5.7.7 NetworkUp boolean parameters: a logical indication of whether a public void setOverScrollMode (int mode) is available. Valid scroll modes are OVER_SCROLL_ALWAYS, OVER_SCROLL_IF_CONTENT_SCROLLS (allow scrolling only if the view content is larger than the container) or OVER_SCROLL_NEVER. Setting the view scroll mode will affect only when the view is scrollable. Int mode parameters: New scroll mode for this view. Added horizontal API 1 Deprecated api level 15 public void setPictureListener (WebView.PictureListener listener) This method has been deprecated at API level 15. This method is now obsolete. Sets the image receiver. This is the interface used to receive notifications about a new image. Listener Parameters WebView.PictureListener: WebView.PictureListener public void setRendererPriorityPolicy (int rendererRequestedPriority, boolean waivedWhenNotVisible) Set rendering priority policy for this webview. Priority policies will be used to determine whether out-of-process rendering should be considered a target for killing OOM. Because the renderer can be associated with more than one webview, the final priority is calculated as the maximum of all attached WebViews. When the WebView is destroyed, it will no longer be consideredered when calculating the renderer priority. When no webviews remain associated with the renderer, the renderer's priority is reduced to RENDERER_PRIORITY_WAIVED. The default policy is to set the priority to RENDERER_PRIORITY_IMPORTANT regardless of visibility, and do not change this unless the caller also handles renderer failures from WebViewClient#onRenderProcessGone. Any other setting will cause WebView renderers to be killed by the system more aggressively than the application. public static void setSafeBrowsingWhitelist <String> hosts,<Boolean> ValueCallback callback) Sets the list of hosts (domain names/IP addresses) that are exempt from SafeBrowsing control. The list is global for all WebViews. Each rule should take one of the following: The rule example matches hostname subdomain example.com Yes . HOSTNAME .example.com No. IPV4_LITERAL 192.168.1.1 No IPV6_LITERAL_WITH_BRACKETS [10:20:30:40:50:60:70:80]No other rules, including wildcards, are invalid. the syntax for hosts is defined by RFC 3986. Hosts Parameters List: Host list This value cannot be null. callback ValueCallback: will be called from</Boolean> </String> </String> if hosts are successfully whitelisted. It will be called from false if all hosts are distorted. The callback will run on the UI thread This value can be null. public void setScrollBarStyle Specify the style of the scroll bars. You can overlay or insert scroll bars. When the snippet, they add to padding the view. And scroll bars can be drawn inside the padding area or at the edge of the view. For example, if the view has a background to draw and you want to draw scroll bars inside the padding specified by drawable, you can use SCROLLBARS_INSIDE_OVERLAY or SCROLLBARS_INSIDE_INSET. If you want them to appear on the edge of the view, ignoring padding, you can use SCROLLBARS_OUTSIDE_OVERLAY or SCROLLBARS_OUTSIDE_INSET. Added horizontal API 1 Deprecated horizontally 23 public void setVerticalScrollbarOverlay (logical overlay) This method has been deprecated at api level 23. This method has no effect. Specifies whether the vertical scroll bar has an overlay style. Logical overlay parameters: True if the vertical scroll bar should have a public style overlay void setWebChromeClient (WebChromeClient client) sets the chrome handler. This is a WebChromeClient implementation for use in JavaScript dialogs, favicons, titles, and progress. This overwrites the current handler. WebChromeClient client parameters: WebChromeClient implementation This value can be null. See also: public static void setWebContentsDebuggingEnabled (logical value enabled) Allows you to debug Web content (HTML/CSS/JavaScript) loaded into any WebViews of this application. You can enable this flag to make it easier to debug Web layouts and JavaScript running inside webviews. See the webview documentation for the debugging guide. The default is false. Parameters enabled boolean: Whether to enable public web content debugging void setWebViewClient (WebViewClient client) sets the WebViewClient, which will receive various notifications and requests. This overwrites the current handler. WebViewClient Client Parameters: WebViewClient Implementation This value cannot be null. See also: public void setWebViewRenderProcessClient (executor executor, WebViewRenderProcessClient webViewRenderProcessClient) Sets the render client object associated with this webview. The renderer client encapsulates callbacks relevant to the WebView render state. See WebViewRenderProcessClient for details. Although multiple WebView instances can share a single source renderer, and renderers can live in an application process or in sandbox mode that is isolated from the application process, instances are set to WebView. Callbacks represent rendering events from the point of view of this webview and may or may not be correlated with rendering events affecting other WebViews. public boolean boolean () Returns true if the press state should be delayed for children or children of this viewgroup. In general, this should be done for scrollable containers, such as a list. This prevents the press state from appearing when the user is actually trying to scroll through the content. The default implementation returns true for compatibility reasons. Subclasses that do not scroll typically need to override this method and return false. Added horizontal API 11 Deprecated api level 18 public logical showFindDialog (text string, boolean showIme) This method has been deprecated in api level 18. This method does not work reliably on all versions of Android; implementing a custom find dialog box using WebView.findAllAsync() provides a more reliable solution. Runs actionmode to find text in this WebView. It only works if this webview is attached to the view system. String text parameters: If not null, there will be initial text to search for. Otherwise, the last search string in this WebView will be used to run. This value can be null. showIme boolean: if true, show the IME, assuming the user starts typing. If false and the text is null, follow find all. Returns true if the find dialog box appears, false otherwise public static void startSafeBrowsing (context context, ValueCallback<Boolean> callback callback) Starts initiating safe browsing. URL loads are not guaranteed to be protected by secure browsing until a callback is called from true. Safe browsing is not fully supported on all devices. For these devices, the callback will receive false. Do not call this if safe browsing has been disabled by the manifest tag or WebSettings#setSafeBrowsingEnabled. This prepares the resources used for safe browsing. This should be called from the application context (and will always use the application context to do its job independently). Parameter context: Application context. This value cannot be null. Callback ValueCallback: Will be called on a UI thread with true if initialization succeeds, false otherwise. This value can be null. public void retentionTransload () Stops the current load. public void zoomBy (float zoomFactor) Performs a zoom operation in this WebView. ZoomFactor float parameters: zoom factor to apply. The zoom factor is clamped to the WebView zoom limits. This value must be between 0.01 and 100.0 inclusive. public logical zoom () zooms in on this Webview. Returns true logical if magnification succeeds, false if there are no public zoomout logical zoomout () zooms in on this WebView. Returns true if zoom out succeeds, false if there are no zoom changes protected int computeHorizontalScrollOffset () Calculating the horizontal horizontal thumb offset of the horizontal scroll bar in the horizontal range. This value is used to calculate the position of the thumb in the scroll bar path. The range is expressed in any units </Boolean> </Boolean> must be the same as those used by computeHorizontalScrollRange() and computeHorizontalScrollExtent(). The default offset is the scroll offset of this view. Returns a horizontal int thumb offset protected by int computeHorizontalScrollRange () Calculating a horizontal range that represents a horizontal scroll bar. The range is expressed in any units, which must be the same as the units used by computeHorizontalScrollExtent() and computeHorizontalScrollOffset(). The default range is the drawing width of this view. Returns the full horizontal range represented by the horizontal scroll bar protected int computeVerticalScrollExtent () Calculates the vertical range of the vertical thumb scroll bar in the vertical range. This value is used to calculate the length of the thumb in the scroll bar path. The range is expressed in any units, which must be the same as the units used by computeVerticalScrollRange() and computeVerticalScrollOffset(). The default range is the drawing height of this view. Returns the int vertical range of the int computeVerticalScrollOffset () Calculate the vertical thumb offset of the vertical scroll bar in the horizontal range. This value is used to calculate the position of the thumb in the scroll bar path. The range is expressed in any entity that must be the same as the units used by computeVerticalScrollRange() and computeVerticalScrollExtent(). The default offset is the scroll offset of this view. Returns the vertical int vertical offset of the int computeVerticalScrollRange () vertical range calculation, which represents a vertical scroll bar. The range is expressed in any entity that must be the same as the entities used by computeVerticalScrollExtent() and computeVerticalScrollOffset(). Returns the int total vertical range represented by the vertical scroll bar The default range is the drawing height of that view. protected void dispatchDraw (canvas) Called by drawing to draw child views. This can be overridden by derived classes to gain control just before it is drawn (but after you draw your own view). Canvas parameters: A canvas on which to draw a protected view void onAttachedToWindow () This is invoked when the view is attached to a window. At this point, it has a surface and starts drawing. Note that this function is guaranteed to be called before onDraw(android.graphics.Canvas), however it can be called at any time before the first onDraw - including before or after onMeasure(int, int). If you override this method, you must call through a superclass implementation. protected void onConfigurationChanged Called when the current configuration of resources used by the application has changed You can use this to decide when to reload resources depending on orientation and other configuration characteristics. Use this only if you do not depend on the mechanism of normal activity of re-creating an activity instance after a configuration change. Parameters newConfig Configuration: New resource configuration. protected void onDraw (canvas canvas) Implement this to make a drawing. Canvas Canvas parameters: The canvas on which the background will be drawn protected void onFocusChanged (logical focus, int direction, rect previouslyFocusedRect) invoked by the view system when the focus state of this view changes. When a focus change event is caused by directional navigation, direction, and earlierFocusedRect provide insight into where the focus originated. During override, call the super class so that standard focus support occurs. If you override this method, you must call through a superclass implementation. Parameters focused boolean: True if the view has focus; false in another way. Direction int: The focus direction has been moved when requestFocus() is called to give this view focus. Values are View.FOCUS_UP, View.FOCUS_DOWN, View.FOCUS_LEFT, View.FOCUS_RIGHT, View.FOCUS_FORWARD, or View.FOCUS_BACKWARD. It may not always apply, in which case you must use the default value. The value View.FOCUS_BACKWARD, View.FOCUS_FORWARD, View.FOCUS_LEFT, View.FOCUS_UP, View.FOCUS_RIGHT, or View.FOCUS_DOWN previouslyFocusedRect Rect: A rectangle in the coordinate system of this previously concentrated view view. Where appropriate, this will be communicated as a finer information about where the focus comes from (except the direction). Otherwise, it will be null. This value can be null. protected void superGenerity (int widthMeasureSpec, int heightMeasureSpec) Measure the view and its contents to determine the measured width and measured height. This method is called by measure(int, int) and should be replaced by subclasses to ensure accurate and effective measurement of their contents. CONTRACT: When overriding this method, call setMeasuredDimension(int, int) to save the measured width and height of this view. Failure to do so will trigger illegalstateexception, thrown by measure (int, int). Calling superclass'onMeasure(int, int) is a valid application. Implementation of the base class measure defaults to the background size unless a larger size is allowed by MeasureSpec. Subclasses should replace onMeasure(int, int) to ensure better measurements of their contents. If this method is overridden, it is the responsibility of the subclass to ensure that the measured height and width are at least the minimum height and width of the view (getSuggestedMinimumHeight() and getSuggestedMinimumWidth()). widthMeasureSpec int parameters: space requirements imposed by the parent. Requirements are encoded using View.MeasureSpec. heightMeasureSpec heightMeasureSpec vertical space requirements imposed by the parent. Requirements are encoded using View.MeasureSpec. protected void onOverScrolled (int scrollX, int scrollY, boolean clampedX, boolean clampedY) triggered by overScrollBy (int, int, int, int, int, int, int, boolean) in response to scrolling results. Parameters scrollX int: True if scrollX was clamped to an over-scroll boundary protected void onScrollChanged (int l, int t, int oldl, int oldt) This is called in response to the inner scroll in this view (i.e. , the view scrolled its own content). This is typically the result of scrollBy(int, int) or scrollTo(int, int) being called. Parameters l int: The current horizontal start of scrolling. t int: Current vertical coils of origin. oldl int: Previous horizontal narrowing of the origin. oldt int: Previous vertical narrowing of the origin. onSizeChanged (int w, int h, int ow, int oh) this is invoked during the layout when the size of this view has changed. If you have just added to the view hierarchy, you will be called with the old values of 0. Parameters in int: The current width of this view. h int: The current height of this view. ow int: The old width of this view. oh int: The old height of this view. Protected void onVisibilityChanged (View changedView, int visibility) Called when view visibility or view parent changes. Protected void onWindowVisibilityS changed (visibility int) invoked when the containing window has changed its visibility (between GONE, INVISIBLE, and VISIBLE). Note that it indicates whether the window is visible to the window manager; it does not say whether the window is obscured by other windows on the screen, even if the same is visible. Visible.