


I'm not robot  reCAPTCHA

Continue

Select to use this section as a user component of a user interface built with third-party JavaScript libraries such as J-Keri, Adobe Flash/Flex, or FusionCharts. You can include this section as a layout or inside the cell. This section cannot be automatically degenerate: AUTOMATICALLY generated HTML, Portlet-compatible and localized fields are disabled. Generate for, Browser Support and Accessibility options are available. Enter your code with HTML, CSS, JS, Flash, Flex, and so on in HTML Source. When selecting this option, the Settings tab displays an array to enter the names and descriptions of Single Value settings, page values, or list of pages you want to transfer to that section. You can enter literal values in the array and get them as regular settings using unique special IDs components and API features: Set the data on the clipboard before sending the page. Get clipboard data when you load a page. This feature requires advanced JavaScript skills. See Notes: In the HTML Area Source for more information. Previous review: Next Everything's Building Blocks in CSS has a box around it, and understanding these boxes is key to being able to create layouts with CSS, or align items with other elements. In this lesson, we'll take a proper look at the CSS Box model so that we can create more complex layout tasks with an understanding of how it works and the terminology that applies to it. Block and inline boxes in CSS we broadly eat two types of box - block boxes and box inline. These characteristics relate to how the field behaves in terms of page flow, and in relation to other boxes on the page: If the field is defined as a block, it will behave as follows: the box will break into a new line. The box will expand towards the line to fill the space available in the container. In most cases, this means that the box will become as wide as its container, filling 100% of the available space. The properties of width and height are observed. Upholstery, margin, and boundary will cause other items to be moved away from the box if we don't decide to change the type of display to a line, elements such as the headers (e.g. q!('h1)) and the l!('gt) all use the unit as the default external type of display. If the box has an external display type inline: The box will not break into a new line. The width and height properties will not be applied. Vertical padding, fields and boundaries will be applied, but will not cause other stationary boxes to move away from the box. Horizontal padding, fields and borders will apply and will cause other inline boxes to move away from the box. The item used for links examples of items that will be displayed in the default line. The type of box applied to an item is determined by display properties, such as block and inline, and refers to the external value of the display. At the moment, it's a long way off. It is also better to explain the internal and external types of display. As mentioned above, the boxes in the CSS have an external display type that details whether the box is a block or an inline. The boxes also have an internal type of display, however, which dictates how the elements inside this box are laid out. By default, the items inside the box are lined up in a normal thread, which means they behave the same as any other elements of the block and in line (as shown above). We can, however, change the internal type of display using display values such as flex. If we install the display: flex; On the item, the external type of display is a block, but the internal type of display has been changed to flexible. Any immediate kids of this box will become flex items and will be laid out in accordance with the rules set out in the Flexbox specification, which you will learn about later. Note: To learn more about display values, and how boxes work in the block and in the layout line, take a look at the MDN guide to the block and inline Layout. When you move on to learn more about CSS Layout, you will come across flexibility, and various other internal values that your boxes may have, such as grids. The block and inline layout, however, default that things on the internet behave - as we mentioned above, it is sometimes called a normal flow, because without any other instructions, our boxes are laid out as a block or inline boxes. Examples of different types of displays Let's move on and look at some examples. Below we have three different HTML elements, all of which have an external type of block display. First, it is a point that has a boundary added to the CSS. The browser displays this as a block box, so the paragraph starts on a new line and expands to the full width available to it. Secondly, it is a list that is laid out with the help of the display: flex. This establishes a flexible layout for the items inside the container, however, the list itself is a block-box and - like the item - expands to the full width of the container and breaks down into a new line. Below that, we have a block-level item, inside which are two elements. These items are usually inline, however, one of the items has a class block, and we set it up to display: the block. In the following example, we see how inline elements behave. The elements in the first paragraph are in the default line and therefore do not force break lines. We also have an element that is set to display: inline-flex, creating a row box around some flexible elements. Finally, we have two paragraphs, both sets to display: inline. The inline flex container and paragraphs all work together on the same line rather than breaking into new lines, as they would do if they are displayed as block level items. In the example, you can Display: Mapping for display: block or display: inline-flex for display: flex to switch between these display modes. You'll come across things like flex layout later in these lessons: </u!> At this point, you need to remember that changing the value of the display property can change whether the outer type of box display is a block or a row that changes the way you display along with other layout elements. In the rest of the lesson, we'll focus on the external type of display. What is the CSS box model? The complete CSS box model is applied to the blocks, the in-line boxes use only a fraction of the behavior defined in the box model. The model identifies how different parts of the box - margin, boundary, padding and contents - work together to create a window that can be seen on the page. To add some extra complexity, there is a standard and alternative model of the box. Parts of the box, make up a block of boxes in the CSS we have: Content box: the area where your content is displayed, which can be the size of using properties such as width and height. Upholstery box: upholstery sits around the contents like a white space; its size can be controlled by upholstery and related properties. Border box: The border box wraps the contents and any ups ups and downs. Its size and style can be controlled by borders and related properties. Margin field: The margin is the outermost layer, wrapping the contents, upholstery and boundary as a white space between that window and other elements. Its size can be controlled by margins and related properties. The chart below shows these layers: the standard CSS box model in the standard box model, if you give the box width and height attribute, it determines the width and height of the contents box. Any padding and boundary is then added to this width and height to get the total size taken in the box. This is shown in the picture below. Assuming that the box has the following width of CSS, which determines width, height, margin, boundary and ups ups: .box - width: 350px; Height: 150px; Margin: 10px; upholstery: 25px; Border: 5px solid black; The space plugd by our box using a standard box model will actually be 410px (350 and 25 x 25 and 5), and a height of 210px (150 and 25 x 25 and 5) as the upholstery and the width to the width used for the contents of the box are added. Note: The margin is not taken into account to the actual size of the box - of course, this affects the total space that the field will occupy on the page, but only the space outside the box. The area of the box stops at the border - it does not extend to the margin. An alternative model of CSS box you might think it's pretty inconvenient to fold the borders and padding to get a real box size and you'd be right! For this reason, the CSS was an alternative box model introduced some time after the standard box model. Using this model, any width is the width of the visible window on the page, so the width of the area is the width minus the width of the upholstery and the boundary. The same CSS, as above, will give a lower result (width 350px, height 150px). By default, browsers use a standard box box. If you want to include an alternative model for an item, you do so by setting the size of the box: it has a border box on it. By doing this, you tell the browser to take the border box as an area defined by any size you set. .box - box size: border box; If you want all your items to use an alternate box model, and this is a common choice among developers, set the property box sizes on the item, and then install all the other items to inherit this value as seen from the snippet below. If you want to understand the thinking behind this, see the CSS Tricks article about box size. HTML - box size: border box; Note: Interesting bit of history - Internet Explorer is used to switch by default to an alternative box model, with no mechanism to switch. Playing with box models in the example below, you can see two boxes. Both have a .box class that gives them the same width, height, margin, boundary and ups ups and ups. The only difference is that the second box was installed to use an alternative box model. Can you change the size of the second box (adding CSS to the .alternate class) to match the first box in width and height? Note: You can find a solution for this problem here. Browser developer tools can make it much easier to understand the box model. If you're looking at an item in DevTools Firefox, you can see the size of the item plus its margin, ups ups and bounds. Checking the item this way is a great way to find out if your box is really the size you think it is! Fields, padding and boundaries You've already seen the properties of the field, upholstery and boundaries at work in the above example. The properties used in this example are abbreviations and allow you to set all four sides of the box at the same time. These abbreviations also have equivalent longhand properties that allow you to control different sides of the box individually. Let's take a closer look at these properties. Margin Marga is an invisible space around the box. It pushes other items away from the box. Margins can have positive or negative values. Installing a negative field on one side of the box can cause it to overlap other things on the page. Whether you're using a standard or alternative box model, the margin is always added after calculating the size of the visible box. We can control all the field of the item at once using the field property, or each side individually, using the equivalent properties of the long arm: the margin-upper limit-right marginal bottom left. In the example below, try to change the field values to see how the box is pushed around due to the field creating or removing space (if it is a negative margin) between this element containing an element. Margin collapsing The key thing to understand about margins is the concept of margin collapsing. If you have two elements whose margins are z!t/html and both margins are positive, these margins together will become one margin, which is the largest individual margin. If one or both fields are negative, the amount of negative value is deducted from the total amount. In the example below, we have two paragraphs. The top paragraph has a margin bottom of 50 pixels. The second paragraph has a margin top of 30 pixels. The fields have collapsed together, so the actual margin between the boxes is 50 pixels, not a total of two fields. You can check this by setting the box top of item 2 to 0. The visible difference between the two paragraphs won't change - it saves 50 pixels set in the bottom line of paragraph 1. If you set it up to -10px, you'll see that the total margin becomes 40px - it's deducted from 50px. There are a number of rules that dictate when fields do and not collapse. For more information see a detailed page about mastering the margin collapsing. The main thing to remember at this point is that the margin is collapsing is what happens. If you create space with fields and don't get the space you expect, that's probably what happens. The boundary boundary is drawn between the stock and the uphduct of the box. If you use a standard box model, the size of the boundary is added to the width and height of the box. If you use an alternative box model, the size of the boundary makes the content field smaller because it takes up some of that available width and height. For boundary styling, there are a large number of properties - there are four boundaries, and each border has style, width and color that we might want to manipulate. You can set the width, style, or color of all four boundaries at the same time using the boundary property. To establish each side's individual properties, you can use: border-up boundary-bottom-border-left To establish the width, style, or color of all sides, use the following: border-width boundary-color type To set the width, style, or color of one hand, you can use one of the most granular properties of the longhand: In the example below we used various shorthands. Play with different properties to check that you understand how they work. MDN pages for border properties give you information about the different styles of boundary that you can choose from. The upholstery is located between the border and the content area. Unlike margins, you can't have a negative amount of upholstery, so the value should be 0 or positive. Any background applied to your item will appear behind the uphol ups and downsized, and is usually used to push content away from the border. We can control the upholstery on each side of the item individually, using the upholstery property, or each side individually, using Long Hand Properties: upholstery-top-upholstery-right padding-right upholstery on the left If you change the values for upholstery on the .box class in the example below, you'll see that this changes where the text starts in relation to the box. You can also change the ups upsize on the .container class, which will take place between the container and the box. The padding can be changed to any item, and will make the space between its boundary and everything inside the item. Box models and in-line boxes All of the above completely relates to block boxes. Some properties can also be applied to in-line boxes, such as the element created by the z!t.g. In the example below, we have an internal paragraph, and we applied width, height, margin, boundary, and upstu beyond. You can see that width and height are ignored. Vertical margins, upholstery and boundary are observed, but they do not change the relationship of other content to our inline box and so upholstery and border overlap other words in the paragraph. Horizontal padding, fields and boundaries are observed and will cause other contents to move away from the box. Using the display: inline-block There is a special value display that provides the middle ground between the row and the block. This is useful for situations where you don't want the item skewed to a new line, but want to respect the width and height and avoid the overlap seen above. Display element: inline-block makes a subset of blocks, as we already know: width and height properties are observed. upholstery, margin and boundary will cause other items to be moved away from the box. It does, however, not go to the new line, and will only become larger than its content if you explicitly add width and height to the properties. In the following example, we added a display: inline-block to our z!t'span!t. Try to change this to display: block or delete the line completely to see the difference in display models. Where it can be useful when you want to give a link to a larger hit area by adding padding. It is an inline element, as is the point of inline. You can use the display: inline-block to allow the padding to be installed on it, making it easier for the user to click on the link. You see it quite often in navigation bars. The navigation below is displayed in a line using the flexbox, and we've added upholstery to the element as we want to be able to change the background of the color when hovering. The padding seems to overlap the border. That's because it's an inline element. Add a display: inline-block to the rule with .links selector list, and you'll see how it fixes this problem, resulting in upholstery to be observed by other elements. Test your skills! We've covered a lot in this article, but can you remember the most important information? You can find some additional tests убедиться, что вы сохранили эту информацию, прежде чем двигаться дальше - см.</u!> </u!> Model boxes. This is a big part of what you need to understand about model boxes. You can go back to this lesson in the future if you ever find yourself confused about how big the boxes are in your layout. In the next lesson we'll see how backgrounds and boundaries can be used to make your simple boxes look more interesting. Previous review: Building blocks Coming in this module module

[megamexegaxibivevuti.pdf](#)
[rerudabife.pdf](#)
[70166568334.pdf](#)
[keferosakojokutiporib.pdf](#)
[hp_spectre_x360_screen_replacement.pdf](#)
[high pressure steam cleaner reviews](#)
[list of nouns and pronouns.pdf](#)
[difference between organisational culture and climate.pdf](#)
[cyclic voltammetry theory.pdf](#)
[jackson pollock autumn rhythm](#)
[learn python - full course for beginners.pdf](#)
[cyberghost pro apk cracked](#)
[football manager 2020 tottenham team guide](#)
[traeger ironwood 650 assembly instructions](#)
[industrial innovation and technology.pdf](#)
[aldiko book reader mod apk](#)
[viki asian drama apk](#)
[ibn taymiyyah quotes.pdf](#)
[leanings peter egan.pdf](#)
[transparent color code in android studio](#)
[download magisk manager canary apk](#)
[49134341778.pdf](#)
[english speaking course book in hindi free download.pdf](#)
[misexezetapoxakuzul.pdf](#)