**Тест 💎**

1. Почему в большинстве ситуаций стоит использовать flexbox-позиционирование?

Flexbox is designed to create flexible layouts. With this technology you can very easily and flexibly arrange elements in the container, allocate the available space between them, and align them in one way or another even if they have no specific dimensions.

Flexbox makes creating an adaptive design much easier than with other types of positioning.

1. Самостоятельно изучите способ позиционирования через display: table и ответьте на вопрос, для каких ситуаций оно лучше всего подходит?

The positioning takes place with the help of a table, using display:table and display:table-cell. This method is used when we want to set all the columns in a row. All the columns will have the same width and no column will be moved to a new row, instead it will be merged. This method is good when we want to align the site content vertically in the center/bottom/top of a page.

In other words, CSS tables via display table bring the useful properties of HTML tables back to light and lend them to a simple div tag. For example, all elements within a table row have the same height and we can centre elements in table cells vertically, which otherwise only works with many little tricks.

One of the useful behaviours of tables: They only become as wide as their content and can be centred horizontally in their encompassing blocks with a simple margin-left: auto and margin-right: auto. This behaviour is inherently both dynamic and responsive.

table-row and table-cell reproduce the row of a table and place, for example, three columns next to each other on large monitors and below each other on small monitors. display:table-cell masters the small feat of centring an element vertically or placing it at the bottom of the comprehensive block.

display:table-row places columns of the same height next to each other, because in an HTML table row all cells are always the same height.

1. Какими способами можно сделать горизонтальное выравнивание по центру? Минимум 3 варианта, можно больше

* The most common and (therefore) easiest type of centering is to centre lines of text in a paragraph or heading. CSS has a 'text-align' property for this:

P { text-align: center }

H2 { text-align: center }

which displays each line in the P paragraph or H2 header centered between the margins

* It can be also done with the help of the flexbox, e.g. by using justify-content: center to align the elements along the main axis.

Sometimes it's not the text that needs to be centred, but the block as a whole. Or to put it another way: we want the right and left margins to be the same. This is done by setting the margins to 'auto'. This is usually used for fixed-width blocks, because if the block itself is flexible, it will just take up all the available width. Here's an example:

P.blocktext {

margin-left: auto;

margin-right: auto;

width: 6em

}

...

<P class="blocktext">This rather ...

Or, in other words, to horizontally center a block element (like <div>), we use margin: auto;

Setting the width of the element will prevent it from stretching out to the edges of its container.

The element will then take up the specified width, and the remaining space will be split equally between the two margins:

.center {  
  margin: auto;  
  width: 50%;  
  border: 3px solid green;  
  padding: 10px;  
}

* To center an image, set left and right margin to auto and make it into a block element:

img {  
  display: block;  
  margin-left: auto;  
  margin-right: auto;  
  width: 40%;  
}

* The default container for absolutely positioned elements is the viewport. (In the case of a browser, this is the browser window).

<html>

<style>

body {

background: white}

section {

background: black;

color: white;

border-radius: 1em;

padding: 1em;

position: absolute;

top: 50%;

left: 50%;

margin-right: -50%;

transform: translate (-50%, -50%)}

</style>

<section>

<h1> Nicely centered </h1>

<p> This text box is vertically centered.

<p> And horizontally if the window is wide enough.

</section>

The 'margin-right: -50%' rule is needed to compensate for 'left: 50%'. The 'left' rule reduces the width available for an element by 50%. Therefore the renderer will try to make the lines no longer than half the width of the container. Indicating that the right margin of the element is to the right by the same amount, we note that the maximum line length is again equal to the width of the container.

Try resizing the window: you will see that each sentence is on one line when the window is wide enough. Only when the window is too narrow for the whole sentence will that sentence be split into multiple lines. When you remove the 'margin-right: -50%' rule and resize the window again, you will notice that the sentences will already be split, although the window is still twice as wide as the lines of text.

1. Какие есть оси во флекс-верстке и как задается их направление?

If “regular” layout is based on both block and inline flow directions, the flex layout is based on “flex-flow directions”. Items will be laid out following either the main axis (from main-start to main-end) or the cross axis (from cross-start to cross-end). The main axis of a flex container is the primary axis along which flex items are laid out. But it is not necessarily horizontal; it depends on the flex-direction property. The axis perpendicular to the main axis is called the cross axis. Its direction depends on the main axis direction.

The direction of the main axis can be changed using the flex-direction CSS property.

css

flex-direction: row;

/\* row (left to right) - by default

row-reverse

column (top-down)

column-reverse (bottom to top) \*/

This property can be used to make flex-elements appear in columns instead of rows. This is done by using column or column-reverse.

1. Разберитесь, как работает свойство margin: auto во флекс-верстке, приведите пример использования

If you apply auto margins to a flex item, that item will automatically extend its specified margin to occupy the extra space in the flex container, depending on the direction in which the auto-margin is applied.

Let’s pick that apart a bit and say we have a simple parent div with a child div inside it:

<div class="parent">

<div class="child"></div>

</div>

And let’s assume we’re using the following CSS to style those divs:

.parent {

display: flex;

height: 400px;

background-color: #222;

}

.child {

background-color: red;

width: 100px;

height: 100px;

}

When we set all margins to auto:

.child {

background-color: red;

width: 100px;

height: 100px;

margin: auto;

}

Then the red box will appear in the center of the parent. It’s like we’re using a popular centering trick by setting justify-content and align-items to center because the child decides to rest in the center of the parent, both horizontally and vertically.

Setting the margin property on a flex child will push the child away from that direction. Set margin-left to auto, the child will push right. Set margin-top to auto and the child will push to the bottom.

1. В чем преимущества box-sizing?

So far the block model has been additive (stackable). We have to add the values of the `width`, `padding` and `border` properties together to get the actual full width of the element.

The block model, however, can be modified so that the calculation is done differently. CSS3 introduced the `box-sizing` property, which allows us to change exactly how the block model works and how element sizes are calculated. The box-sizing property allows you to change this algorithm so that the width and height properties set the size of the box rather than the content.

The `border-box` value changes the block model so that any values of the `border` or `padding` properties are included within the `width` and `height` of the element.

1. Чем отличается flex-grow от flex-shrink?

flex-shrink determines the factor by which a flex item shrinks compared to the other flex items when the space of the flex container becomes too small.

The default value is 1 - proportionally.

flex-shrink : 0 prevents the flex item from shrinking.

flex-grow is the factor for dividing the free space (when the flex container is larger than the contents): flex-grow is the growth factor compared to the other flex items.

flex-grow:2 does not say that an item will grow twice as big as items with flex-grow:1 - only its share of growth is twice the share of items with flex-grow:1 (the default value).

0: Do not widen. Either the width of the item or flex-basis.

1: Default: Same width as all flex items within the row.

≥2: Widen by a factor of 2 compared to all other flex items noted with flex-grow:1.

1. Как можно добиться следующего позиционирования элементов:

<div class="container">

<div class="red"></div>

<div class="green"></div>

<div class="blue"></div>

<div class="purple"></div>

</div>

Then we add the following features:

.container {

height: 200px;

width: 100%;

display: flex;

justify-content: space-between

}

1. Какой в итоге будет размер у элемента (можно округлить)?

Width: 1.989 + 5 + 134.545 + 5 + 1.989 = 148.5

Height: 1.989 + 2 + 11.818 + 2 = 17.8

1. Самостоятельно разберитесь, зачем нужно свойство order?

order:

Allows you to change the order (position, position) of an element in the general row.

Default: order: 0

By default elements are in order: 0 and are put in the order in which they appear in HTML code and the direction of the row. But if you change the value of the order property, the elements will be put in order of values: -1 0 1 2 3 .... For example, if you specify an order: 1 for one of the elements, they will go all zeros first, and then all elements with 1.

This CSS property arranges the flex elements in the flex-container in ascending order of number.

This way you can, for example, move the first element to the end without changing the direction of the rest of the elements or the HTML code.

1. Каким кодом можно сделать такую таблицу?

<table border="1" style="width=50%">

<tr>

<th>Column1</th>

<th>Column2</th>

<th>Column3</th>

</tr>

<tr>

<td rowspan="2">Row 1 Cell 1</td>

<td>Row 1 Cell 2</td>

<td>Row 1 Cell 3</td>

</tr>

<tr>

<td>Row 2 Cell 2</td>

<td>Row 2 Cell 3</td>

</tr>

<tr>

<td colspan="3">Row 3 Cell 1</td>

</tr>

</table>

1. Изучите материалы и найдите, для каких ситуаций подходит position: fixed?

When the page is scrolled, the fixed element stays in place and does not scroll with the page.

The position of the item is always counted from the browser viewport, regardless of the position of the parent.

If the element falls outside the visible area to the right or below it, there are no scroll bars and the element is simply 'cropped'.

Otherwise, fixed is similar to absolute.

This type of positioning is used to create menus, tabs, headers, in general, any elements that should be fixed to the web page and always be visible to the visitor. For example we can set a header fixed and it stays in one place regardless of the amount of information on the site.

Fixed, like absolute, can also be used to display dialogs and messages, but the fixed message cannot be scrolled up or down, it always stays in place.

1. Каким способом лучше всего верстать большие блоки текста?

The best way to create a website with a big amount of information is by using boxes. Every HTML element is, at its most basic form, a box. The box model is used to describe an element’s dimensions and structure. It is made up of four boxes: content box, padding box, border box, and margin box.

1. Как рассчитывается размер flex-контейнера?

The dimensions of the flex elements are calculated as in the normal block model: \*frames + indents + content dimensions. The `box-sizing` works. Flex-elements are compressed to fit the content by default. The final width of a flex-element inside a flex-box will be determined depending on the values of the flex-grow and flex-shrink CSS properties that are set not only for this element, but also for the other flex-elements in this flex-container.



