# Верстка. HTML. CSS.

1. Что такое HTML? В чем его назначение? +
2. Что такое CSS? В чем его назначение? +
3. Что такое DOCTYPE? На что влияет? Какой нужно использовать? +
4. Что такое тэг? На какие группы их можно разделить? Назовите несколько тэгов каждой группы +
5. Что такое атрибут элемента? Назовите несколько стандартных атрибутов. +
6. Какие виды списков вы знаете? Как они обозначаются? +
7. Какие виды input бывают и для чего они используются? Что такое textarea? +
8. Как создать таблицу с наполнением как у таблицы умножения до 5? +
9. Как создать гипертекстовую ссылку, открывающуюся в новой вкладке? +
10. Что такое iframe? Для чего они используются? +
11. Как подключить файл со стилями? +
12. Как подключить javascript файл ? +
13. **Как управлять переносом текста и разбиением слов?**
14. Что делает свойство overflow и какие значения оно может принимать? +
15. **Как сделать так, чтобы текст, который не поместился, заменялся на точки в конце? А как сделать это для многострочного текста?**
16. Какие блочные и строчные элементы вы знаете? Для чего они нужны? +
17. Что такое css-селектор? Напишите селектор для всех элементов одного типа(например для всех span), для элемента с id=’test’, для элемента с атрибутом data-value=’45’, для элемента с классом abc. +
18. Напишите селектор для элемента с классом child, который вложен в элемент с классом parent. Напишите селектор для элемента с двумя классами text и active. Напишите селектор для второго элемента списка внутри маркированного списка с id=’my-list’. +
19. Что такое псевдокласс? Какие псевдоклассы вы знаете? +
20. Как рассчитывается весомость css-селекторов? +
21. Расскажите о блочной модели CSS. Что делает свойство box-sizing, какие значения оно может принимать, что они означают? +
22. Для чего нужно свойство float? Какие проблемы могут возникнуть при его использовании и как их решать? +
23. За что отвечает свойство display, какие значения оно может принимать, и что они означают? +
24. За что отвечает свойство position, какие значения оно может принимать, и что они означают? +
25. Задайте элементу .block следующие отступы одним css свойством: слева 0px, справа 10px, сверху и снизу по 5px. +
26. Как работает функция calc()?  Для чего она нужна? +
27. В каких единицах можно задавать размеры шрифта? Как они работают? +
28. Есть блочный элемент с классом parent, в который вложен элемент с классом child. Оба имеют динамическую ширину и высоту. Напишите стили для этих элементов, такие, чтобы вложенный блок располагался по центру (и по горизонтали и по вертикали). Необходимо 2 варианта один с flexbox, один - без. +
29. **Свойства для выравнивания (align-items, justify-items, place-items, align-content, justify-content, place-content, align-self, justify-self, place-self). Как они ведут себя в flex и grid?**
30. Flexbox-свойства и их значения для контейнера и потомков (display, flex-direction, flex-wrap, flex-flow, gap, row-gap, column-gap, order, flex-grow, flex-shrink, flex-basis, flex). +
31. Расскажите  что такое блок, элемент, модификатор в БЭМ? Как они записываются? +
32. Какие html5 тэги вы знаете?
33. Как писать кроссбраузерную верстку? +
34. Что такое адаптивная верстка? Как ее писать? +
35. Расскажите про любую grid-систему, например про grid-систему bootstrap 4 или  css-grid. +
36. Что пишут в теге head?+
37. **Что такое HTML? В чем его назначение?**

HTML (от английского HyperText Markup Language) — это язык гипертекстовой разметки текста. Он нужен, чтобы размещать на веб-странице элементы: текст, картинки, таблицы и видео.

Когда вы заходите на сайт, браузер подгружает HTML-файл с информацией о структуре и контенте веб-страницы. HTML как бы выстраивает визуальный фундамент сайта, но не «запускает» сайт самостоятельно. Он всего лишь указывает, где располагаются элементы, какой у них будет базовый дизайн, откуда брать [стили для элементов](https://skillbox.ru/media/code/chto_takoe_css/) и скрипты

1. **Что такое CSS? В чем его назначение?**

CSS (Cascading Style Sheets, каскадные таблицы стилей) — язык описания внешнего вида HTML-документа. Это одна из базовых технологий в современном интернете. Практически ни один сайт не обходится без CSS, поэтому HTML и CSS действуют в единой связке.

Каскадные таблицы стилей работают с [HTML](https://skillbox.ru/media/code/chto_takoe_html/), но это совершенно другой язык. HTML структурирует документ и упорядочивает информацию, а CSS взаимодействует с браузером, чтобы придать документу оформление.

На языке HTML мы создаем размеченный текст — документ с гиперссылками, таблицами, маркированными списками, разными начертаниями шрифтов, заголовками, подзаголовками и так далее. Получаем «простыню» текста с таблицами и иллюстрациями. Интернет изобрели ученые, и для них такое положение вещей было приемлемым. Но все изменилось, когда WWW пошел в массы и свои странички начали создавать простые пользователи, которые хотели индивидуальности и самовыражения, а также коммерческие компании со своими корпоративными стандартами оформления. В общем, веб-страницам понадобилось индивидуальное оформление: стиль.

Поэтому берем HTML-основу — и подключаем к ней стиль CSS. С помощью CSS красиво оформляем существующий текст, то есть прописываем уникальные свойства элементам HTML.

1. **Что такое DOCTYPE? На что влияет? Какой нужно использовать?**

Элемент <!doctype> — DTD (document type definition, описание типа документа) указывает тип документа, который используется при написании html кода. Это непарный тег, который должен стоять самым первым в коде каждой страницы вашего сайта.

Директива doctype необходима, чтобы браузер понимал как обрабатывать текущую веб-страничку, так как существует несколько версий языка HTML, а еще имеется XHTML (EXtensible HyperText Markup Language, расширенный язык разметки гипертекста), который похож на HTML, но различается с ним по синтаксису.

Поэтому если вы хотите, чтобы ваш сайт отображался одинаково красиво и ровно во всех популярных браузерах, то обязательно указывайте эту директиву с версией языка html, которую вы используете. Причем doctype следует указывать с общепринятыми правилами.

Если вы будете пользоваться неполным тегом DOCTYPE, устаревшим его видом, или вообще забудете про него, то браузер перейдет в хитрый режим уловок - quirks mode. И возможно странички вашего сайта отобразятся правильно в каких-то браузерах, но не факт, что они будут выглядеть также аккуратно в других популярных браузерах.

1. **Что такое тэг? На какие группы их можно разделить? Назовите несколько тэгов каждой группы**

Тег (дескриптор) — это специальное служебное слово, заключенное в угловые скобки. Его ещё называют «элемент HTML». Тегов в языке HTML много и каждый что-то делает с контентом, который обычно находится внутри скобок или между тегами.  
Если тег парный, то тегу <ТЕГ> соответствует </ТЕГ>.

Условно теги делятся на следующие типы:

* **теги верхнего уровня;**

Эти теги предназначены для формирования структуры веб-страницы и определяют раздел заголовка и тела документа.

[Тег <body>](https://htmlbook.ru/html/body)

[Тег <head>](https://htmlbook.ru/html/head)

[Тег <html>](https://htmlbook.ru/html/html)

* **теги заголовка документа;**

К этим тегам относятся элементы, которые располагаются в контейнере <head>. Все эти теги напрямую не отображаются в окне браузера, за исключением тега <title>, который определяет название веб-страницы.

* + [Тег <meta>](https://htmlbook.ru/html/meta) - Метатеги используются для хранения информации, предназначенной для браузеров и поисковых систем. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных. Хотя тег <meta> всего один, он имеет несколько атрибутов, поэтому к нему и применяется множественное число.
  + [Тег <title>](https://htmlbook.ru/html/title) -Используется для отображения строки текста в левом верхнем углу окна браузера, а также на вкладке. Такая строка сообщает пользователю название сайта и другую информацию, которую добавляет разработчик.
* **блочные элементы;**

Блочные элементы характеризуются тем, что занимают всю доступную ширину, высота элемента определяется его содержимым, и он всегда начинается с новой строки.

<address>, <article>, <aside>,  
<blockquote>,  
<dd>, <div>, <dl>, <dt>, <details>,  
<fieldset>, <figcaption>, <figure>, <footer>, <form>,  
<h1>-<h6>, <header>, <hr>,  
<li>, <legend>,  
<nav>, <noscript>,  
<ol>, <output>, <optgroup>, <option>,  
<p>, <pre>,  
<section>, <summary>,  
<table>,  
<ul>

* **строчные элементы**;

Строчными называются такие элементы веб-страницы, которые являются непосредственной частью другого элемента, например, текстового абзаца. В основном они используются для изменения вида текста или его логического выделения.

<a>, <area>,  
<b>, <bdo>, <bdi>,  
<cite>, <code>,  
<dfn>, <del>,  
<em>,  
<i>, <iframe>, <img>, <ins>,  
<kbd>,  
<label>,  
<map>, <mark>,  
<s>, <samp>, <small>, <span>, <strong>, <sub>, <sup>,  
<time>,  
<q>,  
<ruby>,  
<u>,  
<var>

* **универсальные элементы**;

Особенность этих тегов состоит в том, что они в зависимости от контекста могут быть как блочными, так и встроенными элементами. Если тег используется как встроенный, он не должен содержать любые блочные элементы.

## [Тег <object>](https://htmlbook.ru/html/object)

Сообщает браузеру, как загружать и отображать объекты, которые исходно браузер не понимает.

## [Тег <iframe>](https://htmlbook.ru/html/iframe)

Создает плавающий фрейм.

## [Тег <button>](https://htmlbook.ru/html/button)

Создает на веб-странице кнопку.

## [Тег <applet>](https://htmlbook.ru/html/applet)

Предназначен для вставки на страницу апплетов — небольших программ, написанных на языке Java.

## [Тег <ins>](https://htmlbook.ru/html/ins)

Предназначен для выделения текста, который был добавлен в новую версию документа.

## [Тег <del>](https://htmlbook.ru/html/del)

Используется для выделения текста, который был удален в новой версии документа.

* **Списки**;

## [Тег <ul>](https://htmlbook.ru/html/ul)

Устанавливает маркированный список.

## [Тег <ol>](https://htmlbook.ru/html/ol)

Устанавливает нумерованный список.

## [Тег <li>](https://htmlbook.ru/html/li)

Определяет отдельный элемент списка.

## [Тег <menu>](https://htmlbook.ru/html/menu)

Отображает список меню.

## [Тег <dt>](https://htmlbook.ru/html/dt)

Создает термин в списке определений.

## [Тег <dl>](https://htmlbook.ru/html/dl)

Создаёт список определений.

## [Тег <dir>](https://htmlbook.ru/html/dir)

Создаёт список, содержащий названия директорий.

## [Тег <dd>](https://htmlbook.ru/html/dd)

Указывает термин в списке определений

* **таблицы;**

Таблица состоит из строк и столбцов ячеек, которые могут содержать текст и рисунки. Обычно таблицы используются для упорядочения и представления данных, однако возможности таблиц этим не ограничиваются. C помощью таблиц удобно верстать макеты страниц, расположив нужным образом фрагменты текста и изображений.

Для добавления таблицы на веб-страницу используется тег <table>. Этот элемент служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются соответственно с помощью тегов <tr> и <td>. Таблица должна содержать хотя бы одну ячейку (пример 12.1). Допускается вместо тега <td> использовать тег <th>. Текст в ячейке, оформленной с помощью тега <th>, отображается браузером шрифтом жирного начертания и выравнивается по центру ячейки. В остальном, разницы между ячейками, созданными через теги <td> и <th> нет.

* **Фреймы.**

Фреймы разделяют окно браузера на отдельные области, расположенные вплотную друг к другу. В каждую из таких областей загружается самостоятельная веб-страница определяемая с помощью тега <frame>. С помощью фреймов веб-страница делится на два или более документа, которые обычно содержат навигацию по сайту и его контент. Механизм фреймов позволяет открывать документ в одном фрейме, по ссылке, нажатой в совершенно другом фрейме. Допустимо также использовать вложенную структуру элементов, это позволяет дробить фреймы на мелкие области.

### <frame>

Тег <frame> определяет свойства отдельного фрейма, на которые делится окно браузера.

### <frameset>

Тег <frameset> заменяет собой элемент <body> на веб-странице и формирует структуру фреймов.

### <iframe>

Тег <iframe> создает плавающий фрейм, который находится внутри обычного документа, он позволяет загружать в область заданных размеров любые другие независимые документы.

1. **Что такое атрибут элемента? Назовите несколько стандартных атрибутов.**

Атрибут HTML-это часть языка разметки, используемая для настройки поведения или отображения элемента HTML.Например, атрибуты могут использоваться для изменения цвета, размера или функциональности элементов HTML. Атрибут value определяет значение элемента <input>. Атрибут value используется по-разному для разных типов ввода: для «кнопки», «сброса» и «отправить» — он определяет текст на кнопке.

Class

id

[checked](https://html5beginner.github.io/tags/att_checked.php)

[placeholder](https://html5beginner.github.io/tags/att_placeholder.php)

[value](https://html5beginner.github.io/tags/att_value.php)

[type](https://html5beginner.github.io/tags/att_type.php)

1. **Какие виды списков вы знаете? Как они обозначаются?**

Списки являются одной из основных форм представления контента на странице. Они позволяют группировать связанные между собой фрагменты, объединяя их по смыслу. В HTML представлены три вида списков:

* **Маркированные;**

<ul>

<li>Маркированные;</li>

<li>Нумерованные;</li>

<li>Списки определений.</li>

</ul>

* **Нумерованные;**

<ol>

<li>Позавтракать</li>

<li>Оплатить билет на метро</li>

<li>Съездить к бабушке</li>

<li>Лечь спать</li>

</ol>

* **Списки определений.**

<dl>

<dt>HTML</dt>

<dd>Язык гипертекстовой разметки</dd>

<dt>CSS</dt>

<dd>Каскадные таблицы стилей</dd>

</dl>

1. Какие виды input бывают и для чего они используются? Что такое textarea?

|  |  |
| --- | --- |
| text | Обычный текст |
| email HTML5 | e-mail адрес (требует ввода знака - '@') |
| tel HTML5 | Номер телефона (при вводе на мобильных устройствах включается цифровая клавиатура) |
| url HTML5 | URL адрес (требует ввода точки и названия протокола) |
| password | Пароль (прячет вводимые символы под звездочкой - '\*') |
| number HTML5 | Число (добавляет элементы управления - стрелочки) |
| search HTML5 | Текст для поиска (отличие от обычного text только в стилистике) |
| datetime-local HTML5 | Локальные дата и время (добавляет элементы управления: календарь и часы) |
| date HTML5 | Дата (добавляет элементы управления и календарь) |
| month HTML5 | Месяц и год (добавляет элементы управления и календарь) |
| week HTML5 | Номер недели (добавляет элементы управления и календарь) |
| time HTML5 | Время, часы и минуты (добавляет элементы управления) |
| range HTML5 | Регулятор выбора заданных значений |
| radio | Поле для выбора только одного значения из представленных |
| checkbox | Поле для выбора одного или нескольких значений из представленных |
| color HTML5 | Поле для выбора цвета |
| file | Поле для загрузки файла |
| image | Изображение для отправки формы на сервер (аналог типу submit) |
| submit | Кнопка отправки формы на сервер |
| reset | Кнопка для очистки полей. Если в поле есть значение по умолчанию то будет оно будет вставлено вместо очистки. |
| button | Обычная кнопка (по умолчанию ничего не делает) |
| hidden | Скрытое поле с текстом |

Поле <textarea> представляет собой **элемент формы для создания области, в которую можно вводить несколько строк текста**. В отличие от тега <input> в текстовом поле допустимо делать переносы строк, они сохраняются при отправке данных на сервер. Между тегами <textarea> и </textarea> можно поместить любой текст, который будет отображаться внутри поля.

1. Как создать таблицу с наполнением как у таблицы умножения до 5?

<table>  
<tr>  
 <th>\*</th>

<th>1</th>

<th>2</th>

<th>3</th>

<th>4</th>

<th>5</th>  
</tr>

<tr>  
 <th>1</th>

<td>1</td>

<td>2</td>

<td>3</td>

<td>4</td>

<td>5</td>

</tr>

<tr>

<th>2</th>

<td>2</td>  
 <td>4</td>  
 <td>6</td>  
 <td>8</td>  
 <td>10</td>  
</tr>

<tr>

<th>3</th>

<td>3</td>

<td>6</td>

<td>9</td>

<td>12</td>

<td> 15</td>

</tr>

<tr>

<th>4</th>

<td>4</td>

<td>8</td>

<td>12</td>

<td>16</td>  
 <td>20</td>

</tr>

<tr>

<th>5</th>

<td>5</td>

<td>10</td>

<td>15</td>

<td>20</td>

<td>25</td>

</tr>

</table>

1. Как создать гипертекстовую ссылку открывающуюся в новой вкладке?

<a href="www.yandex.ru" target="\_blank">ваш текст</a>

1. Что такое iframe? Для чего они используются?

IFrame — это кадр внутри кадра. Это компонент HTML-элемента, который позволяет встраивать документы, видео и интерактивные медиафайлы на страницу. Сделав это, вы сможете отобразить дополнительную страницу на главной веб-странице.

Элемент iFrame позволяет включать часть содержимого из других источников. Он может интегрировать контент в любом месте на вашей странице, без необходимости включать их в структуру веб-макета, как традиционный элемент.

[align](https://htmlbook.ru/html/iframe/align)

Определяет как фрейм будет выравниваться по краю, а также способ обтекания его текстом.

[allowtransparency](https://htmlbook.ru/html/iframe/allowtransparency)

Устанавливает прозрачный фон фрейма, через который виден фон страницы.

[frameborder](https://htmlbook.ru/html/iframe/frameborder)

Устанавливает, отображать границу вокруг фрейма или нет.

[height](https://htmlbook.ru/html/iframe/height)

Высота фрейма.

[hspace](https://htmlbook.ru/html/iframe/hspace)

Горизонтальный отступ от фрейма до окружающего контента.

[marginheight](https://htmlbook.ru/html/iframe/marginheight)

Отступ сверху и снизу от содержания до границы фрейма.

[marginwidth](https://htmlbook.ru/html/iframe/marginwidth)

Отступ слева и справа от содержания до границы фрейма.

[name](https://htmlbook.ru/html/iframe/name)

Имя фрейма.

[sandbox](https://htmlbook.ru/html/iframe/sandbox)

Позволяет задать ряд ограничений на контент загружаемый во фрейме.

[scrolling](https://htmlbook.ru/html/iframe/scrolling)

Способ отображения полосы прокрутки во фрейме.

[seamless](https://htmlbook.ru/html/iframe/seamless)

Определяет, что содержимое фрейма должно отображаться так, словно оно является частью документа.

[src](https://htmlbook.ru/html/iframe/src)

Путь к файлу, содержимое которого будет загружаться во фрейм.

[srcdoc](https://htmlbook.ru/html/iframe/srcdoc)

Хранит содержимое фрейма непосредственно в атрибуте.

[vspace](https://htmlbook.ru/html/iframe/vspace)

Вертикальный отступ от фрейма до окружающего контента.

[width](https://htmlbook.ru/html/iframe/width)

Ширина фрейма.

1. **Как подключить файл со стилями ?**

<head>

<link rel="stylesheet" href="./css/style.css">

</head>

1. **Как подключить javascript файл ?**

<script src="./js/script.js"></script>

1. **Как управлять переносом текста и разбиением слов?**

<wbr> - разбиение слов

<br> - перенос строки

word-break: break-all;

Правила переносов текста в этом случае не учитываются, поэтому слова могут переноситься весьма причудливым образом.

|  |  |
| --- | --- |
| normal | Значение по умолчанию. Между словами вставляется только по одному пробелу, дополнительные пробелы отбрасываются. Текст переносится только в случае необходимости. |
| nowrap | Запрещает переносы строк, за исключением применения <br>. |
| pre | Пробелы в тексте не игнорируются, браузер отображает дополнительные пробелы и переносы строк. |
| pre-wrap | Сохраняет пробелы в тексте, делая разрывы строк там, где это необходимо. |
|  |  |
| pre-line | Удаляет лишние пробелы, за исключением случаев <br>. |
| break-spaces | Поведение идентично pre-wrap, за исключением того, что: любая последовательность неудаляемых пробелов всегда занимает место, в том числе в конце строки; возможность переноса строки существует после каждого неудаляемого пробела, в том числе между пробелами. |
| initial | Устанавливает значение свойства в значение по умолчанию. |
| inherit | Наследует значение свойства от родительского элемента. |

1. **Что делает свойство overflow и какие значения оно может принимать?**

Свойство overflow управляет отображением содержания блочного элемента, если оно целиком не помещается и выходит за область заданных размеров.

visible

Отображается все содержание элемента, даже за пределами установленной высоты и ширины.

hidden

Отображается только область внутри элемента, остальное будет скрыто.

scroll

Всегда добавляются полосы прокрутки.

auto

Полосы прокрутки добавляются только при необходимости.

inherit

Наследует значение родителя.

1. **Как сделать так, чтобы текст, который не поместился, заменялся на точки в конце? А как сделать это для многострочного текста?**

text-overflow: clip | ellipsis

Определяет параметры видимости текста в блоке, если текст целиком не помещается в заданную область. Возможны два варианта: текст обрезается; текст обрезается и к концу строки добавляется многоточие. text-overflow работает в том случае, если для блока значение свойства overflow установлено как auto, scroll или hidden.

clip

Текст обрезается по размеру области.

ellipsis

Текст обрезается и к концу строки добавляется многоточие.

line-clamp: [none | <integer>];

<integer>: устанавливает максимальное количество строк перед усечением содержимого, а затем отображает многоточие (…) в конце последней строки.

display: -webkit-box;

-webkit-line-clamp: 3;

-webkit-box-orient: vertical;

overflow: hidden;

В этом примере значение -webkit-line-clamp равно 3, что означает, что будет показано только три строки текста, а остальной текст будет скрыт и в конце последней строки будет добавлено многоточие. Значение этого свойства можно изменять в зависимости от требований. Свойство -webkit-box-orient: vertical; обеспечивает вертикальное выравнивание текста.

Тем не менее, стоит отметить, что свойство -webkit-line-clamp не поддерживается во всех браузерах, поэтому его использование может вызвать проблемы с отображением на некоторых устройствах или в некоторых версиях браузеров.

1. **Какие блочные и строчные элементы вы знаете? Для чего они нужны?**
2. **Что такое css-селектор? Напишите селектор для всех элементов одного типа(например для всех span), для элемента с id=’test’, для элемента с атрибутом data-value=’45’, для элемента с классом abc.**

Селектор - это часть CSS-правила, которая сообщает браузеру, к какому элементу (или элементам) веб-страницы будет применён стиль.

Селектор можно использовать, чтобы выбрать сразу все заголовки на странице и изменить их цвет или шрифт.

Составной селектор — это последовательность простых селекторов, которые не разделены комбинаторами, т. е. за одним селектором сразу идёт следующий.

(Он выбирает элемент, который соответствует всем простым селекторам, которые он содержит. Селектор типа или универсальный селектор, входящий в составной селектор, должен быть расположен первым в этой последовательности. В составном селекторе допустим только один селектор типа или универсальный селектор.)

Сложный селектор — это последовательность селекторов, которые разделены комбинаторами.

Список селекторов — это селекторы, перечисленные через запятую.

## [**Дочерний комбинатор**](https://developer.mozilla.org/ru/docs/Learn/CSS/Building_blocks/Selectors/Combinators#%D0%B4%D0%BE%D1%87%D0%B5%D1%80%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80)

Дочерний комбинатор ( >) располагается между двумя селекторами CSS. При этом будут выбраны только те элементы, которые будут выбраны вторым вторым селектором, которые являются элементами потомками, соответствующими первому селектору. Все элементы-потомки на более низкой иерархии будут пропущены.

## [**Соседний родственный комбинатор**](https://developer.mozilla.org/ru/docs/Learn/CSS/Building_blocks/Selectors/Combinators#%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%BD%D0%B8%D0%B9_%D1%80%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80)

Соседний родственный селектор ( +) используется для выбора элемента, который непосредственно следует за другим элементом и находится на одном с ним уровне иерархии.

## [**Общий родственный комбинатор**](https://developer.mozilla.org/ru/docs/Learn/CSS/Building_blocks/Selectors/Combinators#%D0%BE%D0%B1%D1%89%D0%B8%D0%B9_%D1%80%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80)

Если вы хотите выбрать родственные элементы, даже если они не являются непосредственными соседями, вы можете использовать общий родственный комбинатор ( ~).

span {}  
#test {}  
[data-value=“45“] {}  
.abc {}

1. **Напишите селектор для элемента с классом child, который вложен в элемент с классом parent. Напишите селектор для элемента с двумя классами text и active. Напишите селектор для второго элемента списка внутри маркированного списка с id=’my-list’.**

.parent .child {}

.text.active {}

#my-list > \*:nth-child(2) {}

1. **Что такое псевдокласс? Какие псевдоклассы вы знаете?**

Псевдоклассы определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на неё курсора мыши. При использовании псевдоклассов браузер не перегружает текущий документ, поэтому с помощью псевдоклассов можно получить разные динамические эффекты на странице.

Псевдокласс — это селектор, который выбирает элементы, находящиеся в конкретном состоянии, например, они сначала являются элементом своего типа, или на них наводится указатель у мыши. Обычно они думают так, как если бы вы применили класс к какой-то части вашего документа, что часто помогает сократить избыточные классы в разметке и обеспечивает более гибкий и удобный в вспомогательном коде.

Условно все псевдоклассы делятся на три группы:

* определяющие состояние элементов;
* имеющие отношение к дереву элементов;
* указывающие язык текста.

### :focus

Применяется к элементу при получении им фокуса. Например, для текстового поля формы получение фокуса означает, что курсор установлен в поле, и с помощью клавиатуры можно вводить в него текст

### :hover

Псевдокласс :hover активизируется, когда курсор мыши находится в пределах элемента, но щелчка по нему не происходит.

### :active

Происходит при активации пользователем элемента. Например, ссылка становится активной, если навести на неё курсор и щёлкнуть мышкой. Несмотря на то, что активным может стать практически любой элемент веб-страницы, псевдокласс :active используется преимущественно для ссылок.

### :first-child

Применяется к первому дочернему элементу селектора, который расположен в дереве элементов документа.

### :lang

Определяет язык, который используется в документе или его фрагменте. В коде HTML язык устанавливается через атрибут lang, он обычно добавляется к тегу <html>. С помощью псевдокласса :lang можно задавать определённые настройки, характерные для разных языков, например, вид кавычек в цитатах. Синтаксис следующий.

:nth-child(num) {}

Псевдо-элемент – это виртуальный элемент, который не существует в явном виде в дереве элементов документа. Псевдо-элементы используются для выбора тех частей элемента, которые не могут быть выбраны с помощью других селекторов, а также для стилизации не всего элемента, а отдельных его частей.

1. **Как рассчитывается весомость css-селекторов?**

Формат 0 0 0 0

line стили — 1

id — 2

class или атрибут — 3

tag — 4

1. **Расскажите о блочной модели CSS. Что делает свойство box-sizing, какие значения оно может принимать, что они означают?**

При написании CSS и работе в Интернете в целом очень важно помнить, что все, что отображается с помощью CSS, является блоком. Будь то блок, использующий border-radius, чтобы выглядеть как круг, или даже просто текст: главное помнить, что все это блоки.

Блоки ведут себя по-разному в зависимости от значения display, заданных размеров и содержимого, которое в них находится. Это содержимое может быть еще большим количеством блоков, созданных дочерними элементами, или обычным текстовым содержимым. В любом случае это содержимое будет влиять на размер блока по умолчанию.

Вы можете управлять этим, используя внешний размер, или позволить браузеру принимать решения за вас, основываясь на размере содержимого, используя внутренний размер.

Блоки состоят из отдельных областей модели блока, которые выполняют определенную работу.

Любой блочный элемент состоит из набора свойств, подобно капустным листам накладываемых друг на друга. Основой блока выступает его контент (это может быть текст, изображение и др.), ширина которого задается свойством width, а высота через height; вокруг контента идут поля (padding), они создают пустое пространство от контента до внутреннего края границ; затем идут собственно сами границы (border) и завершают блок отступы (margin), невидимое пустое пространство от внешнего края границ. Порядок влияния этих свойств на блок четко определён и не может быть нарушен.

Когда мы используем box-sizing: content-box, который является значением по умолчанию, он добавит поля, отступы и границы за пределами поля

Но когда мы используем свойство box-sizing: border-box, оно добавит поля, отступы и границы вовнутрь блока

Свойства width и height включают в себя значения полей, но не отступов (margin) и границ (border).

1. **Для чего нужно свойство float ? Какие проблемы могут возникнуть при его использовании и как их решать?**

## Последовательность отображения элементов HTML

По умолчанию HTML-документ формируется сверху вниз, от верхнего края окна браузера по направлению к нижнему краю. Такова и очередность появления элементов веб-страницы. Порядок вывода называется потоком документа.

Элемент, к которому применено свойство float, прижимается к левому или к правому краю родительского элемента (в зависимости от заданного значения), а все элементы, которые расположены ниже, поднимаются и словно обтекают данный элемент. Для лучшего понимания представьте себе камень, который обтекает вода. Собственно, именно поэтому такие элементы еще называют «плавающими».

Данное свойство широко применяется в верстке, поэтому очень важно понять его работу. Float используется для создания макетов с несколькими колонками, навигационных меню, галерей и многого другого.

В CSS существует несколько инструментов для управления этим потоком. Например, вы можете задавать обтекание элементов, а также вытаскивать их из общего потока. Об одном из этих инструментов — свойстве float — пойдет речь далее.

## CSS float

Элемент, к которому применено свойство float, прижимается к левому или к правому краю родительского элемента (в зависимости от заданного значения), а все элементы, которые расположены ниже, поднимаются и словно обтекают данный элемент. Для лучшего понимания представьте себе камень, который обтекает вода. Собственно, именно поэтому такие элементы еще называют «плавающими».

Данное свойство широко применяется в верстке, поэтому очень важно понять его работу. Float используется для создания макетов с несколькими колонками, навигационных меню, галерей и многого другого.

Давайте рассмотрим значения, которые может принимать свойство float. Их всего три:

* left — элемент выравнивается по левой стороне. Элементы, находящиеся ниже в потоке, обтекают его с правой стороны.
* right — значение, противоположное предыдущему. Элемент выравнивается по правой стороне, а элементы, находящиеся ниже в потоке, обтекают его слева по левому краю.
* none — элемент не обтекается и находится в своей обычной позиции.

Как уже упоминалось, элемент со свойством float прижимается к левой либо правой стороне родительского элемента. Это может быть как основной контейнер веб-страницы, так и небольшой блок, скажем, сайдбар.

Применять данное свойство можно как к строчным, так и к блочным элементам. Но поскольку блочный элемент по умолчанию занимает всю ширину контейнера, то при использовании float с блоком рекомендуется задать ему ширину width и таким образом оставить место для содержимого, которое должно обходить элемент. Иными словами, необходимо дать воде возможность обтекать камень, и не перекрывать поток.

## Отмена обтекания: свойство clear

В верстке практически всегда существует необходимость сбросить обтекание после определенного элемента. Например, в случае, когда основной блок веб-страницы обтекается боковой колонкой (сайдбаром), скорее всего, нам понадобится отменить обтекание после сайдбара. Ведь это определенно не последний элемент на странице, и, как минимум, есть еще футер — нижняя часть страницы, «подвал», который должен располагаться внизу, никуда не съезжать и ничего не обтекать.

Сейчас на скриншоте обтекание после сайдбара не отменено, и блок-футер ведет себя странным образом, пытаясь обтекать элемент выше (но ничего не выходит, поскольку ширина как минимум одного элемента этого сделать не позволяет — элемент «застряет»):

Как сделать сброс обтекания? В этом нам поможет свойство clear и его значения:

* left — элемент, к которому применяется свойство clear:left, перестает обтекать элемент со свойством float:left, однако правое обтекание сохраняется.
* right — элемент, к которому применяется свойство clear:right, перестает обтекать элемент со свойством float:right, однако левое обтекание сохраняется.
* both — полностью отменяет обтекание со всех сторон. Элемент смещается вниз и образовывает стандартный поток. На следующие за ним элементы обтекание также перестает влиять.
* none — отменяет очистку clear. Элемент будет вести себя, как обычно, и на него будут влиять настройки float.

Воспользуемся полученными знаниями и зададим нашему футеру свойство clear:both:

## **Проблема 1**. Обтекание влияет на другие элементы

На первом [уроке по свойству float](https://idg.net.ua/blog/uchebnik-css/ispolzovanie-css/float-i-clear) мы рассмотрели пример, где футер, расположенный сразу за плавающими блоками, ведет себя странно, прижимаясь верхним краем к низу float-элемента. Это происходит из-за того, что, по сути, футер обтекает боковую панель, но поскольку он занимает 100% ширины, то обтекание выражается лишь прижиманием к нижнему краю, без перемещения по горизонтали.

Как сделать, чтобы элемент .b4 перестал обтекать соседний? Конечно же, с помощью свойства clear, которое мы уже рассматривали ранее. Повторим еще раз:

.b4 {

clear: both;

}

## **Проблема 2**. Нулевая высота контейнера с float-элементами

Еще одна проблема при работе с float-элементами заключается в том, что блок, содержащий плавающие элементы, теряет свою высоту, поскольку свойство float исключает элементы из нормального потока. Высота родительского контейнера изменяется следующим образом:

* Если в контейнере содержатся только плавающие элементы, то его высота станет нулевой. Либо, если родителю задано, к примеру, свойство padding-top и/или padding-bottom, то его высота будет равна сумме этих padding-ов.
* Если в родительском контейнере помимо float-элементов имеются также обычные элементы, то высота родителя будет равна высоте этих обычных элементов плюс отступы.

Иными словами, из высоты блока-родителя просто вычитается высота float-элементов.

Как заставить родительский контейнер учитывать высоту содержащихся в нем плавающих элементов? Есть несколько способов решения этой проблемы. Мы расскажем о двух наиболее распространенных и эффективных хаках:

Micro Clearfix. Этот хак был придуман разработчиком по имени Nicolas Gallagher, и заключается он в добавлении нескольких строк CSS к псевдоэлементу :after родительского контейнера, потерявшего высоту:

.wrapper:after {

content: " ";

display: table;

clear: both;

}

Этот код работает во всех современных браузерах. Преимущество данного способа состоит в отсутствии необходимости писать лишний HTML-код.

Overflow: hidden. Второй вариант решения проблемы с пропавшей высотой контейнера — добавить к этому блоку свойство overflow: hidden. Вообще, данное свойство определяет, будет ли видно содержание блочного элемента, если оно не помещается в него целиком и выходит за пределы области. Однако в данном случае оно еще и заставляет учитывать высоту float-элементов.

.wrapper {

overflow: hidden;

}

У этого метода есть недостаток: если в контейнере находятся абсолютно позиционированные элементы, размещенные за пределами его границ, то они могут исчезнуть из поля зрения. Впрочем, то же самое произойдет с любым содержимым, которое будет вылезать за область контейнера — такова работа значения hidden.

## **Проблема 3**. Перенос блоков при добавлении отступов

Это случилось из-за того, что ширина каждой колонки стала равна не 33.333333%, а 33.333333% + 10px + 10px. Как вы помните из предыдущего раздела учебника, полная ширина блока — это сумма чисел, в которую включена ширина содержимого, а также размеры боковых отступов, границ и полей.

Чтобы исправить эту проблему, используйте свойство [box-sizing](https://idg.net.ua/blog/uchebnik-css/ispolzovanie-css/box-sizing) со значением border-box. Тогда браузер будет автоматически включать отступы, поля и рамку в заданную вами ширину блока

1. **За что отвечает свойство display, какие значения оно может принимать, и что они означают?**

display многоцелевое свойство, которое определяет, как элемент должен быть показан в документе.

* **none** — полностью скрывает элемент со страницы, не удаляя его при этом из HTML-разметки.
* **block** — элемент ведёт себя как блочный.
* **inline** — элемент ведёт себя снаружи как строчный, а внутри как блочный.
* **flex** — элемент становится флекс-контейнером, ведёт себя как блочный, а вложенные элементы становятся флекс-элементами.
* **grid** — элемент становится грид-контейнером. Снаружи грид-контейнер ведёт себя как блок. Дочерние элементы такого контейнера начинают подчиняться правилам грид-раскладки.

**inline-block** – означает элемент, который продолжает находиться в строке (inline), но при этом может иметь важные свойства блока.

Как и инлайн-элемент:

Располагается в строке.

Размер устанавливается по содержимому.

Во всём остальном – это блок, то есть:

Элемент всегда прямоугольный.

Работают свойства width/height.

1. **За что отвечает свойство position, какие значения оно может принимать, и что они означают?**

Свойство [CSS](https://developer.mozilla.org/ru/docs/Web/CSS) position указывает, как элемент позиционируется в документе. [top](https://developer.mozilla.org/ru/docs/Web/CSS/top), [right](https://developer.mozilla.org/ru/docs/Web/CSS/right), [bottom](https://developer.mozilla.org/ru/docs/Web/CSS/bottom) и [left (en-US)](https://developer.mozilla.org/en-US/docs/Web/CSS/left) определяют конечное местоположение позиционированных элементов.

### **Static**

Static является значением по умолчанию для всех элементов и будет отображаться на странице в соответствии с порядком элемента в документе. Это значение используется редко и, как правило, не требует дополнительных параметров.

### **Relative**

Relative определяет позиционирование элемента относительно его стандартного местоположения на странице. При использовании этого значения можно задать дополнительные параметры, такие как top, bottom, left и right, которые указывают на смещение элемента относительно его стандартного местоположения.

### **Absolute**

Absolute задаёт точное местоположение элемента на странице, независимо от других элементов на странице. При использовании значения можно задать параметры top, bottom, left и right, чтобы определить точное местоположение элемента.

### **Fixed**

Fixed задаёт элемент на фиксированном месте на странице, независимо от прокрутки страницы. При использовании этого значения также можно задать параметры top, bottom, left и right, чтобы определить точное местоположение элемента.

### **Sticky**

Sticky начинает себя вести как обычный элемент до тех пор, пока не достигнет определенной позиции на странице, которую вы зададите с помощью свойства \*\*top, right, bottom или left. Как только элемент достигнет этой позиции при прокрутке страницы, он зафиксируется на месте и будет оставаться там до тех пор, пока пользователь не доскроллит страницу обратно.

## **Нюансы использования**

* Если у элемента значение свойства position установлено в absolute или fixed, то такой элемент будет выведен из обычного потока документа и может перекрыть другие элементы. Если это не предполагается, то нужно следить за порядком элементов на странице и настройками z-index.
* Если у элемента значение свойства position установлено в relative, то это не влияет на позиции других элементов на странице.
* Значение свойства position может также влиять на поведение других свойств, например, на позиционирование через top, left, bottom и right.
* Использование свойства position вместе с другими свойствами, например, transform или display, может дать неожиданные результаты и требует более тщательного тестирования на разных браузерах и устройствах.
* Значение sticky может привести к проблемам с доступностью. Если элемент со значением sticky перекрывает другие элементы на странице, это может затруднить доступ к этим элементам для людей, использующих скринридеры.

1. **Задайте элементу .block следующие отступы одним css свойством: слева 0px, справа 10px, сверху и снизу по 5px.**

Как и с другими универсальными свойствами, которым можно передавать сразу 4 значения для 4 сторон (или углов), свойству margin можно передать 2 значения. К примеру так: "margin: 10px 20px". Тогда верхний и нижний отступ будут равны 10px, а правый и левый будут равны 20px. А если передать 3 значения "margin: 10px 20px 30px", то соответственно верхний отступ получится 10px, правый и левый - 20px, а нижний 20px.

1. **Как работает функция calc()? Для чего она нужна?**

calc() - это функция CSS, которая даёт возможность рассчитать значения свойств CSS во время их определения.

Функция calc() принимает в качестве параметра математическое выражение, результат вычисления которого можно использовать как значение CSS-свойства. Выражение может включать операторы +, -, \*, / с использованием стандартных правил приоритета операторов:

+

Сложение

-

Вычитание.

\*

Умножение. По крайней мере хоть один из сомножителей должен быть [<number>](https://developer.mozilla.org/ru/docs/Web/CSS/number).

/

Деление. Делитель должен быть [<number>](https://developer.mozilla.org/ru/docs/Web/CSS/number).

width: calc(100% - 80px);

1. **В каких единицах можно задавать размеры шрифта? Как они работают?**

Пиксель px – это самая базовая, абсолютная и окончательная единица измерения.

Количество пикселей задаётся в настройках [разрешения экрана](https://ru.wikipedia.org/wiki/Разрешение_(компьютерная_графика)), один px – это как раз один такой пиксель на экране. Все значения браузер в итоге пересчитает в пиксели.

1em – текущий размер шрифта.

Можно брать любые пропорции от текущего шрифта: 2em, 0.5em и т.п.

em – относительные, они определяются по текущему контексту от родителя.

Так как значение в em высчитывается относительно текущего шрифта, то вложенная строка в 1.5 раза больше, чем первая.

Выходит, размеры, заданные в em, будут уменьшаться или увеличиваться вместе со шрифтом. С учётом того, что размер шрифта обычно определяется в родителе, и может быть изменён ровно в одном месте, это бывает очень удобно.

% - процент берётся от размера шрифта родителя.

Rem - Единица rem задаёт размер относительно размера шрифта элемента <html>.

1. **Есть блочный элемент с классом parent, в который вложен элемент с классом child. Оба имеют динамическую ширину и высоту. Напишите стили для этих элементов, такие, чтобы вложенный блок располагался по центру (и по горизонтали и по вертикали). Необходимо 2 варианта один с flexbox, один — без.**

.parent {

display: flex;

justify-content: center;

align-items:center;

}

.parent {

position: relative;

}

.child {

position: absolute;

top: 50%;

left: 50%;

translate: -50% -50%;

1. **Свойства для выравнивания (align-items, justify-items, place-items, align-content, justify-content, place-content, align-self, justify-self, place-self). Как они ведут себя в flex и grid?**

**Flex:**

Основная идея флексов — гибкое распределение места между элементами, гибкая расстановка, выравнивание, гибкое управление.

Флекс-контейнер: элемент, к которому применяется свойство display: flex. Вложенные в него элементы подчиняются правилам раскладки флексов.

Флекс-элемент: элемент, вложенный во флекс-контейнер.

**justify-content**

Свойство позволяет выравнивать флекс-элементы внутри флекс-контейнера по основной оси.

Возможные значения:

start — элементы прижимаются к тому краю, откуда начинается чтение на том языке, на котором отображается сайт.

end — элементы прижимаются к краю, противоположному началу направления чтения на языке сайта.

flex — элементы прижимаются к краю, от которого начинается основная ось.

flex — элементы прижимаются к краю, у которого основная ось заканчивается.

left — элементы прижмутся к левому краю родителя.

right — элементы прижмутся к правому краю родителя.

center — элементы выстраиваются по центру родителя.

space — крайние элементы прижимаются к краям родителя, оставшиеся выстраиваются внутри контейнера равномерно, так, чтобы между ними были одинаковые отступы.

space — свободное пространство делится поровну между элементами и по половине от этой доли размещается по бокам от каждого элемента. Таким образом, между соседними элементами будет равное расстояние, а снаружи крайних элементов — по половине этого расстояния.

space — свободное место будет распределено так, чтобы расстояние между любыми двумя элементами было одинаковым и расстояние от крайних элементов до края было таким же.

**Align-items**

Свойство выравнивания элементов внутри контейнера по поперечной оси.

Возможные значения:

stretch (значение по умолчанию) — элементы растягиваются вдоль поперечной оси так, чтобы заполнить всего родителя. Это очень удобно, если вы делаете двухколоночный макет. Раньше приходилось при помощи разных костылей добиваться одинаковой высоты, а теперь достаточно сделать контейнер флексом, и колонки по умолчанию будут одной высоты.

flex или start — элементы выстраиваются у начала поперечной оси. Разница между ними лишь в том, что второе значение «уважает» направление чтения выбранного языка.

flex или end — элементы выстраиваются у конца поперечной оси. Разница между первым и вторым значениями аналогична предыдущему пункту.

center — элементы выстраиваются по центру поперечной оси.

baseline — элементы выравниваются по базовой линии текста. «Базовая линия» — baseline — воображаемая линия, проходящая по нижнему краю знаков шрифта (без учёта выносных элементов).

**Align-content**

Свойство распределяет свободное пространство по поперечной оси между рядами флекс-элементов. Предположим, у вас 11 элементов в 3 рядах. Если размер родителя по поперечной оси позволяет, то при помощи align можно распределять строчки элементов: по верхнему краю, по нижнему, по центру или равномерно.

Не имеет видимого значения, если элементы располагаются в один ряд.

Возможные значения:

stretch (значение по умолчанию) — ряды растягиваются одинаково, так, чтобы занять всё доступное пространство родителя.

flex / start — все ряды располагаются у начала поперечной оси. Первое значение не зависит от направления чтения текущего языка, в отличие от второго.

flex / end — все ряды располагаются у конца поперечной оси. end «уважает» направление чтения текущего языка.

center — ряды выравниваются по центру родителя.

space — первый ряд прижимается к началу поперечной оси, последний — к концу поперечной оси, а остальные располагаются так, чтобы свободное пространство было поделено на отступы между ними равномерно.

space — отступы у каждого ряда равнозначны отступам у любого другого ряда.

space — отступы между рядами и от краёв родителя одинаковые.

**align-self**

При помощи этого свойства можно выровнять один или несколько элементов иначе, чем задано у родительского элемента. Например, в коде выше у родителя задано выравнивание вложенных элементов по верхнему краю родителя. А для элемента с классом .item мы задаём выравнивание по нижнему краю.

**Grid:**

**justify-content**

Свойство, с помощью которого можно выровнять элементы вдоль оси строки. Данное свойство работает, только если общая ширина столбцов меньше ширины контейнера сетки. Другими словами, вам нужно свободное пространство вдоль оси строки контейнера, чтобы выровнять его столбцы слева или справа.

start — выравнивает сетку по левой стороне грид-контейнера.

end — выравнивает сетку по правой стороне грид-контейнера.

center — выравнивает сетку по центру грид-контейнера.

stretch — масштабирует элементы, чтобы сетка могла заполнить всю ширину грид-контейнера.

space — одинаковое пространство между элементами и полуразмерные отступы по краям.

space — одинаковое пространство между элементами и полноразмерные отступы по краям.

**Justify-items**

Свойство, с помощью которого задаётся выравнивание грид-элементов по горизонтальной оси. Применяется ко всем элементам внутри грид-родителя.

start — выравнивает элемент по начальной (левой для LTR) линии.

end — выравнивает элемент по конечной (правой для LTR) линии.

center — выравнивает элемент по центру грид-ячейки.

stretch — растягивает элемент на всю ширину грид-ячейки.

**Align-items**

Свойство, с помощью которого можно выровнять элементы по вертикальной оси внутри грид-контейнера.

start — выравнивает элемент по начальной (верхней) линии.

end — выравнивает элемент по конечной (нижней) линии.

center — выравнивает элемент по центру грид-ячейки.

stretch — растягивает элемент на всю высоту грид-ячейки.

**Place-items**

Шорткат для указания значений сразу и для align и для justify. Указывать нужно именно в таком порядке.

**Align-self, justify-self, place-self** — Для отдельного элемента.

1. **Flexbox-свойства и их значения для контейнера и потомков (display, flex-direction, flex-wrap, flex-flow, gap, row-gap, column-gap, order, flex-grow, flex-shrink, flex-basis, flex).**

**Flex-direction** - свойство управляет направлением основной оси внутри флекс-контейнера. Поскольку поперечная ось всегда идёт перпендикулярно основной оси, то при помощи этого свойства мы управляем и её поведением тоже.

Если совсем просто, то это свойство расставляет флекс-элементы в ряд или в колонку.

row (значение по умолчанию) — основная ось идёт горизонтально слева направо, поперечная ось идёт вертикально сверху вниз.

row-reverse— основная ось идёт горизонтально справа налево, поперечная ось идёт вертикально сверху вниз.

column — основная ось идёт вертикально сверху вниз, поперечная ось идёт горизонтально слева направо.

column-reverse — основная ось идёт вертикально снизу вверх, поперечная ось идёт горизонтально слева направо.

**flex-wrap**

Флекс-элементы по умолчанию стараются уместиться в один ряд, даже если размер им не позволяет.

Чтобы изменить это поведение, свойству flex нужно задать значение, отличное от nowrap.

По умолчанию значение у свойства flex — nowrap. При этом флекс-элементы помещаются (или пытаются уместиться) в один ряд и не переносятся на новую строку, даже если не влезают в размеры родителя.

Установив значение wrap, мы можем изменить это поведение, и флекс-элементы будут иметь возможность перенестись в новый ряд, если не влезают в одну линию в рамках родителя.

Ещё одно возможное значение — wrap. В этом случае элементы будут располагаться снизу вверх, заполнив собой сперва нижний ряд, а те, что не влезли, перепрыгнут в ряд выше.

**flex-flow**

Шорткат, позволяющий одновременно задать значения свойствам flex и flex.

**Row-gap**

отступы между элементами по строкам

**column-gap**

отступы между элементами по столбцам

**gap**

Шорткат для записи значений свойств row и column. Значения разделяются пробелом.

**order**

При помощи свойства order можно менять порядок отображения флекс-элементов внутри флекс-контейнера.

[**flex-grow**](https://doka.guide/css/flex-grow/)**и [flex-shrink](https://doka.guide/css/flex-shrink/)**

[flex-grow](https://doka.guide/css/flex-grow/) и [flex-shrink](https://doka.guide/css/flex-shrink/) — это CSS-свойства, определяющие, какое количество свободного пространства флекс-контейнера может занимать текущий элемент.

Рассмотрим каждое свойство в отдельности.

[CSS](https://developer.mozilla.org/ru/docs/Web/CSS)-свойство flex-grow определяет как много свободного пространства во flex-контейнере должно быть назначено текущему элементу (flex grow factor — «коэффициент увеличения flex»). Свободное пространство — разница между размером flex-контейнера и размером всех его flex-элементов вместе. Если все sibling-элементы (sibling items — элементы одного уровня вложенности, состоящие по отношению друг к другу в родственной связи как брат или сестра), имеют одинаковый коэффициент flex-grow, то все они получат одинаковую долю свободного пространства, в противном случае оно распределяется в соответствии с соотношением, определённым различными коэффициентами flex-grow.

flex-shrink— свойство [CSS](https://developer.mozilla.org/ru/docs/Web/CSS) , которое определяет фактор сжатия flex-элемента. Flex-элементы будут заполнять контейнер в зависимости от значения flex-shrink, когда стандартная ширина flex-элементов больше, чем flex-контейнер.

Свойство [CSS](https://developer.mozilla.org/ru/docs/Web/CSS) flex - это [сокращённое свойство](https://developer.mozilla.org/ru/docs/Web/CSS/Shorthand_properties), определяющее способность гибкого элемента растягиваться или сжиматься для заполнения собой доступного свободного пространства. Это свойство устанавливает [flex-grow](https://developer.mozilla.org/ru/docs/Web/CSS/flex-grow), [flex-shrink](https://developer.mozilla.org/ru/docs/Web/CSS/flex-shrink) и [flex-basis](https://developer.mozilla.org/ru/docs/Web/CSS/flex-basis).

как и у каждого из подсвойств этого свойства:

* [flex-grow](https://developer.mozilla.org/ru/docs/Web/CSS/flex-grow): как указано
* [flex-shrink](https://developer.mozilla.org/ru/docs/Web/CSS/flex-shrink): как указано
* [flex-basis](https://developer.mozilla.org/ru/docs/Web/CSS/flex-basis): как указано, но с относительной длиной, конвертируемой в абсолютные длины

1. **Расскажите что такое блок, элемент, модификатор в БЭМ? Как они записываются?**

**БЭМ (Блок, Элемент, Модификатор)** — компонентный подход к веб-разработке. В его основе лежит принцип разделения интерфейса на независимые блоки. Он позволяет легко и быстро разрабатывать интерфейсы любой сложности и повторно использовать существующий код, избегая «Copy-Paste».

**Блок** - функционально независимый компонент страницы, который может быть повторно использован. В HTML блоки представлены атрибутом class.

#### Вложенность

Блоки можно вкладывать друг в друга.

Допустима любая вложенность блоков.

**Элемент** - составная часть блока, которая не может использоваться в отрыве от него.

#### Вложенность

Элементы можно вкладывать друг в друга.

Допустима любая вложенность элементов.

Элемент — всегда часть блока, а не другого элемента. Это означает, что в названии элементов нельзя прописывать иерархию вида block\_\_elem1\_\_elem2.

**Модификатор** - сущность, определяющая внешний вид, состояние или поведение блока либо элемента.

Особенности:

[Название модификатора](https://ru.bem.info/methodology/naming-convention/" \l "%D0%98%D0%BC%D1%8F-%D0%BC%D0%BE%D0%B4%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80%D0%B0-%D0%B1%D0%BB%D0%BE%D0%BA%D0%B0) характеризует внешний вид («какой размер?», «какая тема?» и т. п. — «размер»: size\_s, «тема»: theme\_islands), состояние («чем отличается от прочих?» — «отключен»: disabled, «фокусированный»: focused) и поведение («как ведет себя?», «как взаимодействует с пользователем?» — «направление»: directions\_left-top).

Имя модификатора отделяется от имени блока или элемента одним подчеркиванием (\_).

1. **Какие html5 тэги вы знаете?**

|  |  |
| --- | --- |
| **<article>** | Раздел контента, который образует независимую часть документа или сайта, например, статья в журнале, запись в блоге, комментарий. |
| **<aside>** | Представляет контент страницы, который имеет косвенное отношение к основному контенту страницы/сайта. |
| **<audio>** | Загружает звуковой контент на веб-страницу. |
| **<bdi>** | Изолирует отрывок текста, написанный на языке, в котором чтение текста происходит справа налево, от остального текста. |
| **<canvas>** | Холст-контейнер для динамического отображения изображений, таких как простые изображения, диаграммы, графики и т.п. Для рисования используется скриптовый язык JavaScript. |
| **<data>** | Элемент используется для связывания значения атрибута value, которое представлено в машиночитаемом формате и может быть обработано компьютером, с содержимым элемента. |
| **<datalist>** | Элемент-контейнер для выпадающего списка элемента <input>. Варианты значений помещаются в элементы <option>. |
| **<details>** | Создаёт интерактивный виджет, который пользователь может открыть или закрыть. Представляет собой контейнер для контента, видимый заголовок виджета помещается в элемент <summary>. |
| **<dialog>** | Интерактивный элемент, с которым взаимодействует пользователь для выполнения задачи, например, диалоговое окно, инспектор или окно. Без атрибута open не виден для пользователя. |
| **<embed>** | Элемент-контейнер для встраивания внешнего интерактивного контента или плагина. |
| **<figcaption>** | Заголовок/подпись для элемента <figure>. |
| **<figure>** | Самодостаточный элемент-контейнер для такого контента как иллюстрации, диаграммы, фотографии, примеры кода, обычно с подписью. |
| **<footer>** | Определяет завершающую область (нижний колонтитул) документа или раздела. |
| **<header>** | Секция для вводной информации сайта или группы навигационных ссылок. Может содержать один или несколько заголовков, логотип, информацию об авторе. |
| **<main>** | Контейнер для основного уникального содержимого документа. На одной странице должно быть не более одного элемента <main>. |
| **<mark>** | Выделяет фрагменты текста, помечая их желтым фоном. |
| **<meter>** | Индикатор измерения в заданном диапазоне. |
| **<nav>** | Раздел документа, содержащий навигационные ссылки по сайту. |
| **<output>** | Поле для вывода результата вычисления, рассчитанного с помощью скрипта. |
| **<picture>** | Элемент-контейнер, содержащий один элемент <img> и ноль или несколько элементов <source>. Сам по себе ничего не отображает. Дает возможность браузеру выбирать наиболее подходящее изображение. |
| **<progress>** | Индикатор выполнения задачи любого рода. |

|  |  |
| --- | --- |
| **<ruby>** | Контейнер для Восточно-Азиатских символов и их расшифровки. |
| **<rb>** | Определяет вложенный в него текст как базовый компонент аннотации. |
| **<rt>** | Добавляет краткую характеристику сверху или снизу от символов, заключенных в элементе <ruby>, выводится уменьшенным шрифтом. |
| **<rtc>** | Отмечает вложенный в него текст как дополнительную аннотацию. |
| **<rp>** | Выводит альтернативный текст в случае если браузер не поддерживает элемент <ruby>. |
| **<section>** | Определяет логическую область (раздел) страницы, обычно с заголовком. |
| **<source>** | Указывает местоположение и тип альтернативных медиаресурсов для элементов <picture>, <video>, <audio>. |
| **<summary>** | Создаёт видимый заголовок для элемента <details>. Отображается с закрашенным треугольником, кликнув по которому можно просмотреть подробности заголовка. |
| **<template>** | Используется для объявления фрагментов HTML-кода, которые могут быть клонированы и вставлены в документ скриптом. Содержимое элемента не является его дочерним элементом. |
| **<time>** | Определяет дату/время. |
| **<track>** | Добавляет субтитры для элементов <audio> и <video>. |
| **<video>** | Добавляет на страницу видео-файлы. Поддерживает 3 видео формата: MP4, WebM, Ogg. |
| **<wbr>** | Указывает браузеру возможное место разрыва длинной строки. |

1. **Как писать кроссбраузерную верстку?**

Важная информация: нет необходимости загружать и устанавливать все имеющиеся приложения во всех их релизных версиях. Чтобы проверить, как ваш веб-сайт будет отображаться, достаточно провести тестирование на трех основных обозревателях, которые работают на наиболее популярных и распространенных движках:

Google Chrome (Blink).

Safari (WebKit).

Mozilla Firefox (Gecko).

Если ваш сайт корректно отображается на этих браузерах, вероятность возникновения проблем при использовании других обозревателей снижается, так как именно эти программы отвечают за преобразование и форматирование и используются в большинстве известных веб-браузеров, таких как Яндекс, Opera, Gnome Web, MS Edge, Tor и другие.

Для того чтобы сделать оценку о том, насколько эффективно работает веб-ресурс, обычному пользователю требуется всего лишь несколько секунд. Среди ключевых факторов оценки можно выделить:

Адаптивность. В настоящее время более 70% всего интернет-трафика генерируется на мобильных устройствах. Это делает адаптивность важным аспектом, особенно учитывая разнообразие смартфонов и планшетов. Google также уделяет этому аспекту большое внимание при индексации новых ресурсов.

Время загрузки. Чем дольше нужно ждать, пока страница откроется, тем больше вероятность, что посетитель уйдет на другой сайт.

Положение элементов. Проблемы с расположением, наложениями или выходом за границы экрана сразу бросаются в глаза.

Соответствие функционала. Если сценарии JavaScript не срабатывают, кнопки не реагируют, а результат работы программы не соответствует ожиданиям, использование ресурса может стать проблематичным.

Читаемость текстового контента. Часто возникают трудности с кодировкой, из-за чего статьи могут превращаться в набор непонятных символов.

Использование CSS-хаков

CSS-хаки — это участки кода, нацеленные на конкретные браузеры и игнорируемые другими. Этот метод позволяет частично устранить отдельные проблемы, но может усложнить общую структуру кода и сделать его менее читаемым.

Интеграция универсальных элементов

Это более простой способ, который позволяет использовать общие элементы, понятные для большинства браузеров. Для проверки совместимости элементов с разными движками можно воспользоваться ресурсом caniuse.com.

Использование вендорных префиксов

Этот метод подразумевает добавление префиксов к названиям свойств, что помогает браузерам понять, как применять эти свойства. В отличие от хаков, это считается более правильным подходом с точки зрения получения последовательных результатов.

1. **Что такое адаптивная верстка? Как ее писать?**

Адаптивная вёрстка меняет дизайн страницы в зависимости от поведения пользователя, платформы, размера экрана и ориентации девайса и является неотъемлемой частью современной веб-разработки. Она позволяет существенно экономить и не отрисовывать новый дизайн для каждого разрешения, а менять размеры и расположение отдельных элементов.

Медиазапросы:  
@media screen and (min-width: 800px) and (max-width: 1200px)

Резиновая верстка:  
Строить верстку на процентах, vm, vh.

1. **Расскажите про любую grid-систему, например про grid-систему bootstrap 4 или css-grid.**

CSS Grid Layout ([спецификация](https://www.w3.org/TR/css-grid-1/)) или просто гриды — это удобная технология для раскладки элементов на веб-страницах. В отличие от [флексбоксов](https://doka.guide/css/flexbox-guide/), одновременно работающих только с одним измерением, гриды дают возможность работать одновременно с двумя: горизонталью и вертикалью.

Принцип работы гридов чем-то похож на таблицы. Вместо работы только с рядами или только с колонками с помощью гридов можно работать с так называемыми грид-ячейками, позиционируя элементы по вертикали и горизонтали одновременно.

Грид-контейнер: родительский элемент, к которому применяется свойство display grid.

Грид-элемент: дочерний элемент, прямой потомок грид-контейнера. Подчиняется правилам раскладки гридов.

Грид-линия: разделительная линия, формирующая структуру грида. Может быть как вертикальной (грид-линия колонки), так и горизонтальной (грид-линия ряда). Располагается по обе стороны от колонки или ряда. Используется для привязки грид-элементов.

Грид-ячейка: пространство между соседними грид-линиями. Единица грид-сетки.

Грид-полоса: пространство между двумя соседними грид-линиями. Может быть проще думать о грид-полосе как о ряде или колонке.

Грид-область: область, ограниченная четырьмя грид-линиями. Может состоять из любого количества ячеек как по горизонтали, так и по вертикали.

Свойства, задающие размеры и количество колонок или рядов грид-раскладки соответственно.

.container {

display: grid;

/\* Будет создано 3 колонки \*/

grid-template-columns: 150px auto 40%;

/\* Будет создано 3 ряда \*/

grid-template-rows: 250px 10vw 15rem;

}

Можно явно именовать грид-линии, используя для этого квадратные скобки:

.container {

display: grid;

grid-template-columns: [start] 250px [line2] 400px [line3] 600px [end];

grid-template-rows: [row1-start] 15rem [row1-end] 30vh [last];

Если нужны одинаковые колонки или ряды, то можно воспользоваться функцией repeat.

Будет создано 3 колонки по 250 пикселей:

.container {

display: grid;

grid-template-columns: repeat(3, 250px);

}

С появлением гридов у нас появилась и новая единица измерения: fr

fr (от fraction — доля, часть) отвечает за свободное пространство внутри грид-контейнера.

Например, этот код создаст три колонки, каждая из которых будет занимать 1/3 ширины родителя:

.container {

display: grid;

grid-template-columns: repeat(3, 1fr);

}

Что аналогично записи:

.container {

display: grid;

grid-template-columns: 1fr 1fr 1fr;

}

Свободное пространство рассчитывается после того, как место отдано всем фиксированным размерам. К примеру, в коде ниже сначала будет создана колонка шириной 200 пикселей, а затем свободное пространство — ширина родителя минус 200 пикселей — будет поделено между остальными колонками. Каждая будет занимать ширину (100% - 200px) / 2:

.container {

display: grid;

grid-template-columns: 1fr 200px 1fr;

}

### grid

Свойство-шорткат для свойств grid, grid. Позволяет записать все значения в одну строку. Главное после этого не запутаться при чтении

Можно прописать все колонки и ряды сразу, разделяя их слэшем /. Сперва идут ряды, а затем колонки, не перепутайте!

.container {

display: grid;

grid-template: repeat(4, 1fr) / repeat(3, 500px);

}

В этом же свойстве можно задавать значение и для свойства grid, но тогда код превращается в кашу и становится совершенно нечитабельным. Лучше всё же использовать это свойство отдельно:

### row, column

Задают отступы между рядами или колонками.

.container {

display: grid;

grid-template-columns: 1fr 350px 1fr;

grid-template-rows: repeat(3, 150px);

/\* Отступы между рядами \*/

row-gap: 50px;

/\* Отступы между колонками \*/

column-gap: 20px;

}

### grid

Двуличное свойство, которое может как указывать элементу, какую из именованных областей ему нужно занять, так и служить шорткатом для одновременного указания значений для четырёх свойств: grid, grid, grid и grid

1. **Что пишут в теге head?**

Тег <head> используется для определения заголовочной части HTML-документа.

<head> не отображается на веб-странице, но содержит информацию, которая может быть использована браузером и другими приложениями. Этот тег содержит мета-информацию о документе, такую как заголовок, ключевые слова, ссылки на стили, скрипты и другие данные, которые необходимы для правильного отображения и интерпретации веб-страницы.

Определение заголовка страницы с помощью тега <title>:

<title>Заголовок страницы</title>

Определение описания страницы с помощью метатега description:

<meta name="description" content="Описание страницы">

Определение ключевых слов для страницы с помощью метатега keywords:

<meta name="keywords" content="ключевые слова">

Ссылка на файлы стилей с помощью тега <link>:

<link rel="stylesheet" href="styles.css">

Ссылка на файлы скриптов с помощью тега <script>:

<script src="script.js"></script>

Определение кодировки страницы с помощью метатега charset:

<meta charset="UTF-8">

Определение масштаба страницы для мобильных устройств с помощью метатега viewport:

<meta name="viewport" content="width=device-width, initial-scale=1.0">