Эберт Елена

Шпаргалки для начинающего верстальщика HTML/CSS

Читает вся страна

Шпаргалки для начинающего верстальщика HTML/CSS

Елена Эберт

ISBN 978-5-0053-3486-2
Создано в интеллектуальной издательской системе Ridero

Вступление

Когда я только начинала учить верстку, я записалась на бесплатные курсы по основам HTML/CSS на одной достаточно известной платформе, дополнительно я читала книгу Э. Робсон, Э. Фримена «Изучаем HTML, XHTML и CSS (Head First)».

Все было отлично до того момента, пока я не сделала попытку сверстать небольшую простенькую веб-страницу самостоятельно. Оказывается пройденные курсы и прочитанная книга – это было зря потраченное время, так как информация, которую, например, мне дали курсы, была дана кусочками, дозировано, вследствие чего у меня совершенно не сложилось целостное понимание как и что я должна делать.

Да я уже знала основные теги в HTML, у меня были базовые знания CSS, но только теоретические, применять их в реальности на примере реальной веб-страницы я не умела.

Так пришла идея проанализировать основные типовые решения, типовой код, который используется при верстке веб-страницы и сформировать шпаргалку, с помощью которой я могу действительно верстать сайты, а не выполнять теоретические задачи из заданий курса, на которые в итоге я лишь трачу свое драгоценное время.

Надеюсь мои шпаргалки помогут таким же начинающим верстальщикам сформировать понимание логики и структуры работы над версткой и продвигаться далее в этот интересный захватывающий мир веб-разработки.

Также прошу учесть, что предложенные пути решения той или иной проблемы, имеют и иные варианты, здесь указаны самые простые для понимания, чтобы новичок мог сам понять и усвоить нужные действия и решения, а далее, с опытом смог их самостоятельно усложнять/ упрощать.

Итак, начинаем!

Как работать с данной книгой

Практически любой дизайн-макет, которому требуется верстка состоит из стандартных элементов: кнопки, карточки, текст и пр. Поэтому если начинающий верстальщик хочет как можно быстрее перейти к практике, научиться верстать, писать код для реальных проектов, ему нужно в первую очередь освоить навык работы с указанными стандартными / универсальными элементами.

Всем известно, что чтобы лучше и быстро разобраться с тонкостями любой работы, любой профессии, следует как можно раньше начинать практиковаться. Поэтому важно сразу теорию совместить с практикой, для начала советую в поисковике найти бесплатные макеты для верстки для начинающих, взять макет и начинать его верстать, параллельно изучая теоретические моменты. Только так вы научитесь верстать за достаточно короткое время.

В данной книге вы найдете код для стандартных элементов, о которых я писала выше, вы можете изучить код, прочитать про теги, которые в нем использованы, далее повторить данный код, учитывая особенности своего макета. Получается данные шпаргалки – ориентир вам на начальном этапе верстки.

Далее с практикой вы будете верстать все более сложные макеты и лучше ориентироваться какие свойства и параметры вам следует в том или ином случае написать.

Когда вы изучите и на практике закрепите основные принципы и способы работы с HTML и CSS, вы можете скачивать уже сверстанные шаблоны с psd исходниками. Сначала верстать самостоятельно по макету, используя знания и навыки полученные с данной шпаргалкой и после каждого блока можно сверять с исходной версткой, что бы приблизительно понимать какие ошибки сделаны, как более оптимально можно было решить ту или иную проблему, так ваше обучение верстке будет намного эффективней.

Создаем каркас сайта

Общая универсальная структура нашей страницы

Страница сайта является обычным текстовым файлом, расширение которого. html.

Как его создать? Можно открыть Блокнот через панель Пуск на компьютере и нажать во вкладке Файл – Сохранить как…

Сохранить мы должны данный файл как index.html.

Аналогично в нашу папку с названием проекта мы должны положить файл для CSS, его мы должны назвать style. css.

Внутри данного файла и хранится текст HTML-страницы. Он должен обязательно иметь такие теги как:

– тег <html>, который содержит в себе текст всего сайта (все, что написано вне его, браузер будет проигнорировать)

– внутри <html> должен обязательно быть тег <head>, в нем отражается служебное содержимое страницы

– также внутри <html> должен обязательно быть тег <body>, в нем отражен основной текст, который и виден на экране браузера.

В теге <head> должны быть тег <title>, он задает название страницы, в браузере, тег <meta>, задающий кодировку страницы (в атрибуте charset, как правило, это значение utf-8).

Также перед тегом <html> пишется конструкция doctype, она указывает на версию языка HTML, на которой сделан сайт. В настоящее время актуальна версия, которую выражает <!DOCTYPE html>.

Структура любой страницы имеет общую структуру, которая в целом выглядит так:

```
<!DOCTYPE html>
<html lang=«en»>
<head>
<meta charset=«UTF-8»>
<meta name=«viewport» content=«width=device-width, initial-scale=1.0»>
<title> Заголовок страницы </title>
```

<link rel=«stylesheet» href=«style. css»> // подключаем CSS
</head>
<body>
<header> шапка сайта </header>
<main class=«main»> основная часть </main>
<footer class=«footer»> подвал сайта </footer>
</body>
</html>

Пишем шапку сайта

Что такое шапка сайта? Это верхняя часть сайта, где размещены логотип компании, контактные данные компании и панель навигации. Рассмотрим ниже два самых распространенных типа шапки сайта.

В HTML шапку сайта оформляют с помощью тега header, это парный тег.

> *Тег – это специальное зарезервированное слово в языке HTML, его ключевая составляющая. Именно тегом начинается код и им же заканчивается. Внутри тега располагается информация, которая отображается на веб-странице.*
>
> *Теги могут быть парными и одиночными. Парный тег состоит из открывающего и закрывающего тегов. Открывающий тег изображается с помощью знаков “ <” и ">». У закрывающего тега перед именем стоит слэш “/».*
>
> *Например, <header> – открывающий тег, </header> – закрывающий.*

Как правило, в 99% случаев шапка сайта сделана стандартно, с позиции языка HTML, поэтому нам, новичкам, следует только несколько раз повторить код указанный ниже на реальных примерах, на практике, чтобы понять как данная шапка с позиции HTML устроена и идти по пути обучения верстке далее.

> *Начинающему верстальщику для того, чтобы лучше запомнить теги и свойства, а также чтобы лучше ориентироваться в своем коде, необходимо писать комментарии, что и как мы делаем. Комментарии в оформляются следующим образом:*
>
> *<! – Здесь мы пишем комментарий – >*

Так как контент сайта расположен по середине, а по бокам у него имеется пространство, можно предположить, что все содержимое сайта помещено в контейнер – это некий блок, в которым расположена вся информация веб-страницы.

<header>
 // весь контент, вся информация располагается в контейнере

</header>

Далее верстаем шапку.

Оформляем шапку сайта в HTML (тип 1)

Первый тип состоит из логотипа, формы поиска, аватар пользователя.

Стандартный HTML-код для первого типа шапки сайта будет выглядеть так:

```
<header class=«header»>
<div class=«logo»> Логотип </div>
<! – Здесь форма поиска – >
<form action=" " class=«search»>
<input type=«search» class=«search-input» placeholder=«Поиск»>
<button type=«submet» class=«search-button»>
<img src="img/search-icon.svg» alt=«search-icon»>
</button>
</form>
<! – Здесь аватар пользователя – >
<div class=«user»>
<img src=«img/Изображение – Иконка.svg» alt=«bookmark» class=«bookmark»>
<img src=«img/Аватарка пользователя.png» alt=«avatar» class=«avatar»>
</div>
</header>
```

Кратко опишу теги, примененные выше в коде

button – данный тег создает кликабельную кнопку.

img – добавляет на веб-страницу HTML-изображения. Это одиночный тег. У него есть обязательный атрибут src, который указывает путь к изображению, и необязательный атрибут alt, который содержит в себе резервный контент. То есть если на веб-странице по какой-либо причине не откроется изображение, на ней будет указана информация, которую содержит атрибут alt.

Пример кода:

```
<img src=«img/Иконка (или фотография).svg» alt=«Иконка (Фото)»>
```

form содержит наше форму поиска, рассмотрим ее подробнее далее.

Ширина изображения

По умолчанию изображения отображаются реальным размером, однако мы можем изменить его размер через атрибут width, который задает ширину изображению

<img src="foto.png» width=«100»>

Если мы, например, зададим ширину 100, а реальные размеры изображения были 200X200 пикселей, то изображение станет 100X100, так как вследствие изменения ширины, изменилась и наша высота, чтобы пропорции изображения не исказились.

Аналогично, если мы зададим высоту изображению с помощью height

<img src=" foto.png» height=«100»>

Примечание: одновременно не следует задавать и широту и высоту в <img>.

Форма поиска по сайту

Запишем отдельно общий пример формы поиска по сайту


```
<form>
<input type=«search»>
<input type=«submit» value=«Найти»>
</form>
```

form – определяет форму в HTML документе. По сути данный элемент – это просто контейнер, внутри которого можно разместить разные надписи, элементы управления и типы входных элементов, флажки, радио-кнопки и пр.

input – ключевой элемент тега form, он определяет пользовательское поле для ввода информации. Поле ввода может принимать различный вид, который зависит от атрибута, который применен к input. К примеру, placeholder.

type – указывает браузеру, к какому типу относится элемент формы.

placeholder – это атрибут тега input, он указывает подсказку, которая описывает ожидаемое значение для ввода в элемент:

– email (поле для адреса электронной почты)

– password (поле с паролем, в котором скрываются символы)

– search (текстовое поле для ввода строки поиска)

– text (однострочное текстовое поле).
Пример кода:

```
<input type=«search» placeholder=«Поиск»>
```

Зачем нужны секция, контейнер и див-блоки

section (секция) – это полочки, разделы, в которых размещается блок какого-то контента, данный блок объединен определенной графикой или картинкой. То есть его предназначение – выделять цветовым решением или графикой определенный контент.

container (контейнер) – блок, который центрует контент по середине.

Тег div является пустым универсальным контейнером, который наполняется определенным содержанием. Его цель – логически объединить любой набор элементов внутри данного тега в единственном блоке.

При этом если мы зададим данному div-блоку определенный класс, то через данный класс далее мы можем вложенные элементы стилизовать средствами CSS, или динамически манипулировать ими с применением скриптов Javascript.

Зачем нужен class

Классы необходимо использовать, когда нужно определить стиль для индивидуального элемента веб-страницы или задать различные стили для одного тега. Отметим, что использовать русские буквы в именах классов нельзя.

Атрибут class указывает одно или несколько имен классов для элемента HTML. Если мы в HTML пишем элементу несколько классов, их мы должны указать просто через пробел.

```
<h1 class=«title-home office»>
```

Как видно из примера выше, если класс состоит из несколько слов, частей, то записывается оно через тире, например, title-home, title-1 и т. д.

В целом название класса можно описывать, используя различные направления и правила, ориентируясь на DOM, БЭМ и прочее. Советую вам, если интересно, почитать данную тему отдельно и для самого себя выстроить определенную систему названий, которые для вас и желательно для других разработчиков будут логичны и понятны.

Имя класса может использоваться CSS, как говорилось выше, чтобы задать различные стили для одного тега. Для того, чтобы записать стили для определенного класса, перед его именем ставят точку:

```
.title-home {
стили
}
```

В JavaScript имя класса используется для выполнения определенных задач для элементов.

Оформляем шапку сайта в HTML (тип 2)

Второй тип шапки состоит из логотипа, меню – навигации сайта и аватар пользователя или какой-либо иной информации

Логотип как правило оформляется в виде картинки:

```
<img src="img/Logo.png» alt=«logo» class=«logo»>
```

Данную картинку-логотип мы скачиваем в формате png с макета в Photoshop и Figma или другой программе, в которой работал дизайнер.

<nav> – это тег навигации сайта, в него входит меню сайта в виде списка ссылок:

```
<nav class=«menu-list»>
<a href=«#» class=«menu-link»> </a>
<a href=«#» class=«menu-link»> </a>
<a href=«#» class=«menu-link»> </a>
<a href=«#» class=«menu-link»> </a>
</nav>
```

Навигацию сайта можно создать различными способами, например, не менее распространен способ создания с помощью списка.

Далее идет аватарка пользователя, во многом схожая с шапкой первого типа:

```
<div class=«user»>
```

```
<img src=«img/User Картинка» alt=«user» class=«user-icon»>
<p class=«user-text»> User </p>
</div>
```

В целом код шапки второго типа получился такой:

```
<header>
<div class=«container»>
<div class=«header»>
<img src="img/Logo.png» alt=«logo» class=«logo»>
<nav class=«menu-list»>
<a href=«#» class=«menu-link»> Menu One </a>
<a href=«#» class=«menu-link»> Menu Two </a>
<a href=«#» class=«menu-link»> Menu Three </a>
<a href=«#» class=«menu-link»> Menu Four </a>
</nav>
<div class=«user»>
<img src=«img/User icon.png» alt=«user» class=«user-icon»>
<p class=«user-text»> User </p>
</div>
</div>
</div>
</header>
```

Кратко опишу теги, примененные выше в коде

 Menu – это тег ссылки. Нажимая на ссылку, мы можем перейти на другую страницу сайта или на другой сайт. В данном теге обязательно должен быть атрибут href, в него записывают адрес страницы, на которую ведет ссылка.

Лайфхак к ссылке: когда мы только пишем код, на любые ссылки необходимо ставить «затычки» #:

<nav> – используют для обозначения содержимого в форме основных навигационных ссылок. Документ может иметь несколько данных элементов, к примеру, один для навигации по сайту, а второй для навигации по странице.

<p> – определяет абзац в HTML-документе, при его отображении браузер автоматически вставляет до и после него отступы.

Оформляем шапку первого типа в CSS

Описать универсальное оформление шапки сайта в CSS на порядок сложнее, чем в HTML, постараюсь указать основное.

Во-первых, нам нужно ограничить контейнер, то есть задать ему максимальную ширину, для каждого макета она своя, а также выравниваем содержимое данного контейнера по середине – margin: auto;

```
.container {
max-width: 1140px;
margin: auto;
}
```

В классе header можно поставить универсальное выравнивание элементов по центру, чтобы выровнять логотип соответственно прочим элементам шапки, также универсально распределение элементов органично по длине шапки и пр.

```
.header {
display: flex; /*выравнивание по центру по вертикали*/
align-items: center; /*раскидываем по сторонам*/
justify-content: space-between;
}
```

justify-content: space-between; – данное свойство равномерно распределяет элементы по всей строке. Первый и последний элемент прижимаются к соответствующим краям нашего контейнера. В целом про свойство justify-content смотрите далее.

display: flex; – делает все дочерние элементы резиновыми – flex, а не блочными, как было изначально. Если родительский блок содержит, например, графические элементы, они становятся анонимными flex-элементами.

Оформим непосредственно меню, ниже указан пример, отмечу, что при работе над проектом данные вы должны смотреть в своем макете (боковая панель слева).

```
.menu-link {
font-size: 16px;
line-height: 24px;
color: #2E266F;
text-decoration: none;
margin-right: 45px;
}
```

Кратко опишу теги, примененные выше в коде

text-decoration: none; – убирает автоматическое подчеркивание в созданном нами списке

font-size: – определяет размер шрифта элемента.

line-height: – высота строки.

color: – определяет цвет текста.

Причем существует несколько цветовых моделей, через которые можно указать значение цвета.

Так зеленый цвет можно вывести в свойстве color как:

– green

– #41AB0D

– rgb (255,0,0)

– rgba (255,100,0,.5)

– hsl (0, 20%, 50%)

– hsla (221, 100%, 50%,.8) и пр.

Самые популярные это green и #41AB0D.

Через # записывается шестнадцатеричная система счисления цвета. Выше приведена сокращенная запись hex-цветов.

Есть специализированные сайты, с помощью которых можно узнать название того или иного цвета, его прочие цветовые модели. Например, сайт Colorscheme https://colorscheme.ru/color-names.html

max-width: – устанавливает максимальную ширину элемента.

margin: auto; – свойство margin в значении auto применяется для горизонтального центрирования элемента в его контейнере. Данный элемент будет занимать заданную ширину, а остальное пространство будет равномерно распределено между левым и правым полями.

Оформляем шапку второго типа в HTML с помощью Bootstrap

Оформляем с помощью библиотеки Bootstrap, как ее подключить расскажу далее.

```
<header class=«container header»>
<div class=«row»>
<div class=«col-2»>
<a href="/" class=«logo-link»>
<img width=«128» src=«img/Фото логотипа» alt=«logo» class=«logo-image»>
</div>
</div>
<div class=«col-6»>
<nav>
<ul class=«navigation»>
<li class=«navigation-icon»>
<a href=«#» class=«navigation-link»> Womens </a>
</li>
<li class=«navigation-icon»>
<a href=«#» class=«navigation-link»> Mens </a>
</li>
<li class=«navigation-icon»>
<a href=«#» class=«navigation-link»> Goods </a>
</li>
<li class=«navigation-icon»>
<a href=«#» class=«navigation-link»> Brands </a>
</li>
<li class=«navigation-icon»>
<a href=«#» class=«navigation-link»> Blog </a>
</li>
</ul>
</nav>
</div>
```

```
<div class=«col-2»>
<button class=«button»>
<img class=«button-icon» src=«img/Фото» alt=«icon Card»>
<span class=«button-text»> Cart </span>
</button>
</div>
</div>
</header>
```

Кратко опишу теги, примененные выше в коде

container – это ключевой строительный блок Bootstrap. Его применяют для содержания, заполнения, центрирования содержимого внутри него.

 класс «row"является элементом библиотеки Bootstrap. Ее сетка состоит из строк и колонок, что дает возможность позиционировать элементы на странице как это необходимо по макету. Сам класс означает ряд, который занимает всю ширину элемента, внутри которого он находится, выравнивание по горизонтали реализуется через колонки col.

Здесь через теги <ul> и <li> сформирован неупорядоченный список (список, где маркеры-точки, а не цифры 1, 2, 3, …).

Универсальный пример списка:

```
<ul>
<li> пункт списка </li>
<li> пункт списка </li>
</ul>
```

Его результат будет:

– пункт списка
– пункт списка

Через теги <ol> и <li> можно сформировать упорядоченный список. Универсальный пример списка:

```
<ol>
```

 пункт списка
 пункт списка

Его результат будет:

– пункт списка
– пункт списка

Как оформить блок из 2х частей

Если блок состоит из текста с кнопкой в первой части и картинки/ фото с другой, то

в HTML мы оформим его так

* *
* *
* *
<h1 class=«title-home-office»> Заголовок </h1>
* Текст *
<button class=«button»>
* Explore Projects *
<img src=«img/Arrow 3.png» alt=«arrow-icon» class=«button-icon»>
</button>
* *
* *
<img src="img/house.png» alt=«house» class=«house-img»>

```
</div>
</div>
</div>
```

И если в CSS укажем

```
.row {
display: flex;
}
```

наша картинка встанет справа, как нам нужно

Оформляем блок с карточками

Если нам нужен, например, блок, состоящий из 3-х карточек, мы сначала оформляем первую карточку, далее копируем ее код для двух следующих карт, потом вставляем во вторую и третью карточки соответствующую информацию и картинки.

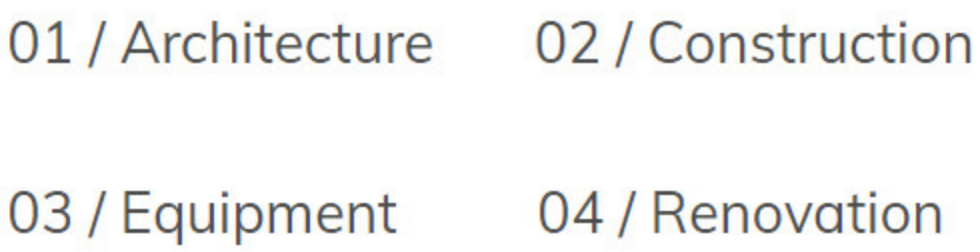
HTML мы оформим так

```
<! – оформляем для карт общий контейнер, далее карточку №1 – >
<div class=«container»>
<div class=«row»>
<! – карточка 1 – >
<div class=«col-6 mb-4»> <! – применяем Bootstrap – >
<div class=«card card-1»>
<h3 class=«card-title»> Заголовок карточки 1 </h3>
<p class=«card-text»> Текст карточки 1 </p>
<! – если в карточке есть кнопка – >
<button class=«button»>
<span class=«button-text»> Надпись на кнопке </span>
```

<img src=«img/Иконка на кнопке» alt=«icon» class=«button-icon»>
</button>

Текст списком

Если нужно сделать текст списком как указано на рисунке, есть такой вариант



в HTML

```
<ul class=«list»>
<il class=«list-card card-1»> 01 / Architecture </il>
<il class=«list-card»> 02 / Construction </il>
</ul>
<ul class=«list»>
<il class=«list-card card-2»> 03 / Equipment </il>
<il class=«list-card»> 04 / Renovation </il>
</ul>
```

в CSS

```
.list {
display: flex;
justify-content: flex-start;
padding-left: 0;
font-size: 28px;
```

```
color: #4E4E4E;
margin-top: 50px;
}
.card-1 {
padding-right: 50px;
}
.card-2 {
padding-right: 70px;
}
```

Верстаем подвал

Футером или подвалом сайта называется нижняя область веб-страницы под контентом. Ее используют как дополнительный блок с информацией. Как правило, подвал содержит контактную информацию компании, ее представительство в соцсетях и прочее.

Сверстаем следующий подвал

в HTML

```
<footer class=«footer»>
<div class=«container»>
<div class=«row»>
<div class=«col-7»> <! – применяем Bootstrap – >
<nav>
<ul class=«footer-menu»>
<li class=«footer-menu-item»> <a href=«#» class=«footer-menu-link»> Shop </a> </li>
<li class=«footer-menu-item»> <a href=«#» class=«footer-menu-link»> About Us </a> </li>
<li class=«footer-menu-item»> <a href=«#» class=«footer-menu-link»> Careers </a> </li>
<li class=«footer-menu-item»> <a href=«#» class=«footer-menu-link»> FAQ </a> </li>
<li class=«footer-menu-item»> <a href=«#» class=«footer-menu-link»> Blog </a> </li>
<li class=«footer-menu-item»> <a href=«#» class=«footer-menu-link»> Contacts </a> </li>
</ul>
```

```
</nav>
</div>
<div class=«col-3»>
<span class=«footer-text»> Follow Us </span>
<span class=«footer-social»>
<a href=«#» class=«social-link»> <img src="img/Facebook.svg» alt=«Facebook»> </a>
<a href=«#» class=«social-link»> <img src="img/Instagram.svg» alt=«Instagram»> </a>
<a href=«#» class=«social-link»> <img src="img/Twitter.svg» alt=«Twitter»> </a>
</span>
</div>
<div class=«col-2»>
<span class=«footer-text»> © 2021 Universal UI Kit </span>
</div>
</div>
<div class=«row justify-content-between»>
<div class=«col-4»>
<img class=«pay-logo» src="img/bitcoin.png» alt=«bitcoin»>
<img class="pay-logo"src="img/VISA.png» alt=«VISA»>
<img class=«pay-logo"src=«img/pay play.png» alt=«pay play»>
<img class="pay-logo"src="img/sber.png» alt=«sber»>
</div>
<div class=«col-2 justify-content-end»>
<a href=«#» class=«top-link»>
<span class=«top-link-text»> Top </span>
<img src="img/top.svg» alt=«icon»>
</a>
</div>
</div>
</div>
</footer>
```

в CSS

Приведу лишь несколько универсальных свойств, остальные необходимо делать по своему макету

```
.footer {
background: green;
padding-top: 45px;
padding-bottom: 40px;
color: #fff;
}
.footer-text {
margin-right: 15px;
color: #E7D9FF;
font-size: 14px;
line-height: 20px;
}
.footer-menu-link {
color: #fff;
text-decoration: none;
font-weight: 900;
font-size: 14px;
line-height: 20px;
}
.social-link {
text-decoration: none;
margin-left: 15px;
}
.footer-social {
margin-right: 15px;
}
.top-link {
text-decoration: none;
color: #fff;
}
.top-link-text {
font-weight: 900;
font-size: 14px;
line-height: 20px;
```

```
margin-right: 15px;
}
.pay-logo {
vertical-align: middle;
margin-right: 25px;
}
```

Примечание по коду

vertical-align: middle; – данное свойство управляет вертикальным выравниванием элементов внутри своих родителей. Его применяют к строчным элементам и к ячейкам таблиц. Значение middle для строчных элементов выравнивает середину текущего элемента по середине родительского элемента. Для табличных элементов данное значение выравнивает содержимое текущей ячейки по центру строки, учитывая внутренние отступы.

Часто используемые свойства в CSS

Отображение на экране

По умолчанию элементы делятся на строчные и блочные. Блочные не могут стоять в одной строке друг с другом, а строчным нельзя задать размер, их размер зависит от их содержимого. Для того, чтобы изменить поведение элементов, применяют display и его значения.

Самые основные значения display это:

– block – элемент становится блочным, даже если был строчным. Вследствие чего ему можно записывать размеры, внешние отступы и пр.

– inline- преобразуем в строчный элемент.

– none – элемент исчезает со страницы.

– inline-block – блочно-строчный тип, его блочные свойства сохраняются, однако он может стать в одну строку с другими блоками, если у них тоже прописано inline-block и им хватает места по ширине.

Примечание: по умолчанию блок занимает в родителе все свободное место по ширине, вследствие чего необходимо вручную определить ширину для него.

– table – преобразуем элемента в таблицу.

Оформляем элемент

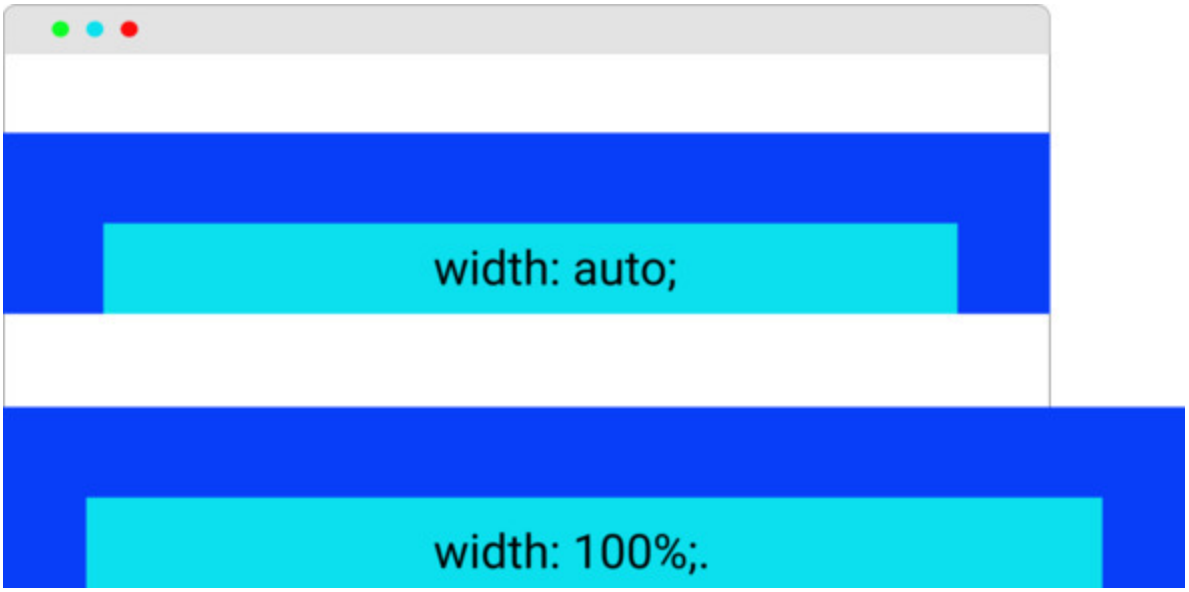
Фоновый цвет

Это свойство background-color: – фоновый цвет нашего дива/ контейнера /…

Свойство width

width – CSS-свойство задает не общую ширину блока, а ширину его содержания. Общая ширина блока состоит из 3х компонентов: ширина содержания, внутренние отступы и ширина рамок слева и справа.

Поведение элемента может зависеть от того, как ему задана его ширина:



Если свойство width стоит по умолчанию, то есть мы не задаем ширину, то оно считается auto;. То тогда блок занимает всю ширину родительского блока. Если у данного блока есть внутренние отступы/ рамки, то его ширина содержания автоматически уменьшается, а общая ширина осталась равна ширине родителя.

Если width задана 100%, то ширина содержания блока равна ширине родительского блока. Если у блока будут еще внутренние отступы и рамки, то его общая ширина станет больше ширины родителя.

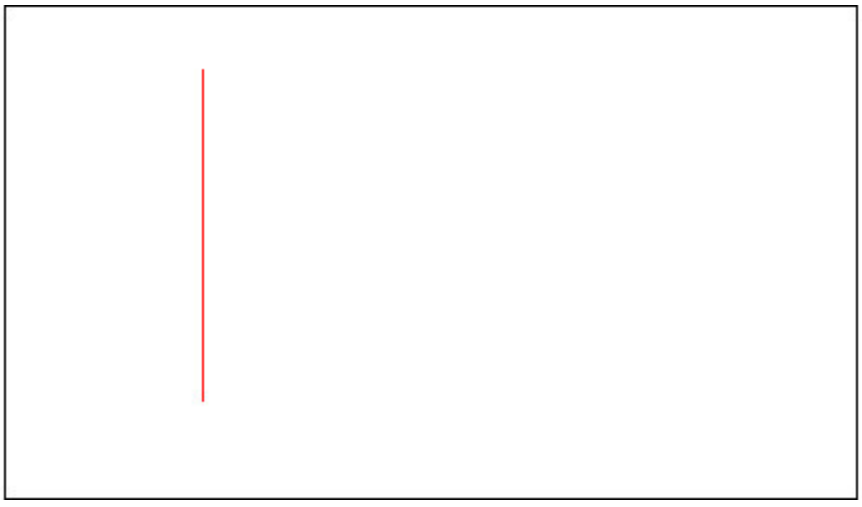
– width – ширина
– height – высота
– max-width, min-width – максимальная или минимальная ширина. Если, к примеру, у блока max-width, то данная ширина не будет жесткой, она будет меняться в зависимости от размеров окна.

Если же указать min-width, то тогда блок не сможет стать уже, чем указанный размер.

Аналогично max-height и min-height.

Оформляем границу

Границу можно оформить при помощи трех свойств:
свойство border-width задает толщину границы
border-color – цвет границы
border-style – тип границы. Его значения может быть:
solid задает сплошную линию
dotted – линия в виде точек
dashed – линия в виде тире
ridge – выпуклая линия
double – двойная линия
Можно сделать границу для отдельных сторон:
border-left (левая граница). Например, border-left: 1px solid red;
На экране мы увидим:



border-right (правая граница)
border-top (верхняя граница)
border-bottom (нижняя граница).

Скругление углов

Скругляет углы свойство border-radius, причем если задать border-radius: 50px; получиться круг.

Если нужно для разных углов задать разное скругление, пишем: border-radius: 10px 20px 10px 40px;

Здесь первое значение задает скругление для верхнего левого угла, второе – для верхнего правого, третье – для нижнего правого угла, а четвертое – для нижнего левого угла.

Причем скругление можно задавать в процентах. Так border-radius: 50%; также образует круг.

Фоновое изображение

Свойство background-image задает фоновую картинку / изображение элементу. При этом следует не забывать указывать свойство background-repeat: no-repeat; чтобы картинка не отразилась много раз, а была в одном экземпляре.

background-image: url (путь к изображению);

Например:

background-image: url("foto.png»); background-repeat: no-repeat;

url – это путь к файлу с изображением, название изображение может быть в двойных кавычках, одинарных и вообще без кавычек.

Если фоновое изображение нужно прокручивать вместе с текстом

Свойство background-attachment указывает каким образом прокручивается фоновое изображение элемента: вместе с текстом или текст будет скользить по картинке. Например:

background-attachment: scroll;

fixed – изображение фона будет неподвижным а текст будет скользить по нему

scroll – изображение фона будет прокручиваться вместе с текстом

local – фон фиксируется, учитывая поведение элемента. Если элемент имеет прокрутку, то фон будет прокручиваться вместе с текстом, однако фон, который выходит за рамки элемента остается на месте.

Что означает padding и margin

padding

устанавливает внутренний отступ между внутренним краем рамки элемента и его содержимым. При этом важно помнить, что добавление внутренних полей будет воздействовать на общий размер элемента. Данное свойство может содержать от одного до четырех значений, которые разделяют между собой пробелами.

Так padding-top устанавливает отступ между верхним внутренним краем рамки элемента и его содержимым.

При указании четырех значений (5px 8px 10px 12px) – порядок расстановки внутренних отступов следующий: top (5px) – right (8px) – bottom (10px) – left (12px).

top – верхнее поле

right – правое поле

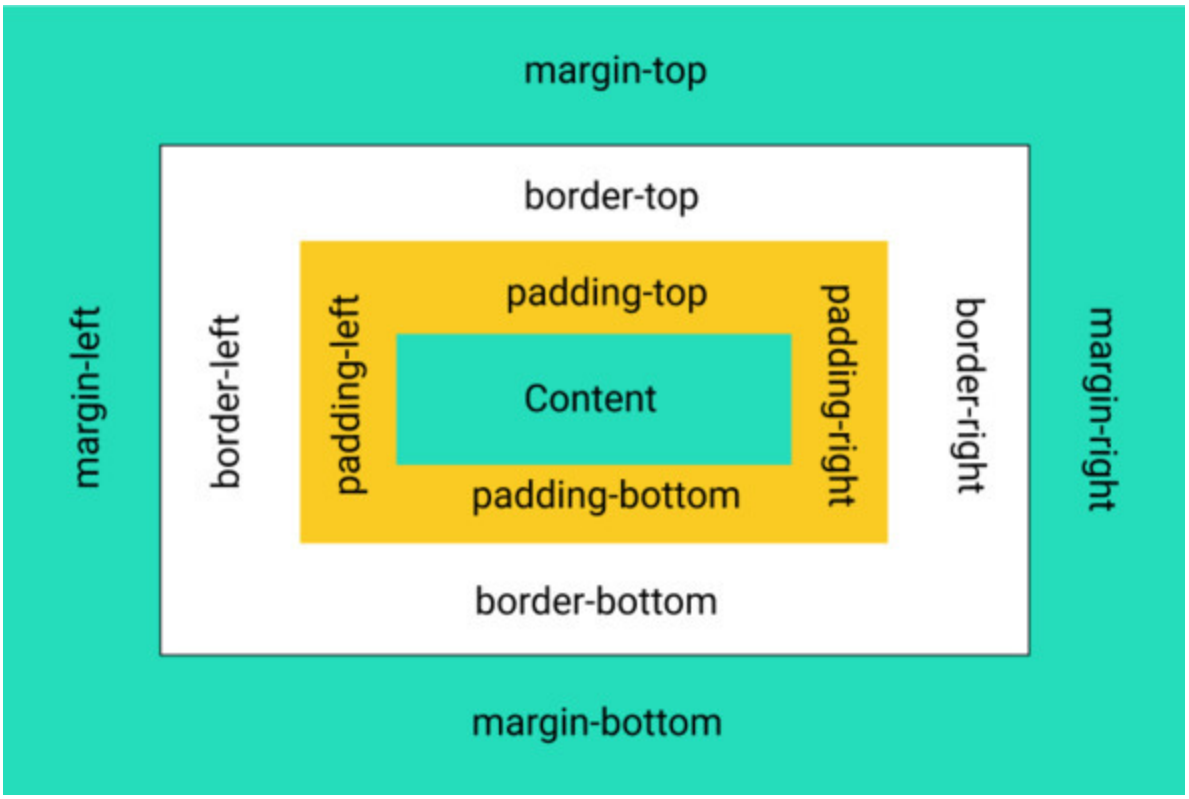
bottom – нижнее поле

left -левое поле

Примечание: данные свойства top, right, bottom и left могут принимать не только положительные, но и отрицательные значения.

margin

устанавливает значения внешнего отступа, может содержать от одного до четырех значений, которые разделяются между собой пробелами. Например, margin:8px 5px 10px 20px;



Размеры в CSS

В целом размеры в верстке могут быть в пикселах (px), процентах (%) или в других единицах измерения. Значение размера может быть как положительным, так и отрицательным.

Так в целом процент берется от размера родителя. Однако здесь могут быть исключения.

К примеру, пишем в CSS для img {width: 30%;} здесь ширина width: 30%; означает, что наша картинка будет занимать место в размере 30 процентов, от ширины окна нашего браузера, иными словами, чем больше разрешение пользователя, тем больше будет картинка, чем меньше разрешение, тем меньше.

Для width или height, как правило, процент берется от ширины или высоты родителя, но если элемент position: fixed, то процент будет браться от ширины или высоты окна, а не родителя.

При установке свойства margin-left в %, процент будет браться от ширины родительского блока, а не от его margin-left.

Есть также исключение в line-height. В нем процент будет браться от текущего размера шрифта, а вовсе не от непосредственно свойства line-height родителя.

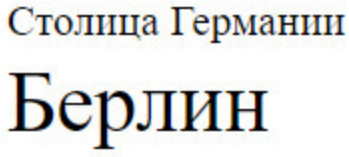
Исключения по % можно посмотреть в стандарте Visual formatting model details.

1em – текущий размер шрифта. Можно брать любые его пропорции: 2em, 0.5em и т. д. Размеры в em – это относительные размеры, они определяются по текущему контексту. Например

```
<div style=«font-size:1.5em»>
Столица Германии
<div style=«font-size:1.5em»>
Берлин
</div>
</div>
```

Хотя в данном кода указан размер у текущего шрифта (Столица Германии) и вложенного (Берлин) одинаковый. Вложенная строка

будет в два раза больше, чем с текущим шрифтом.



Получается, что размеры, заданные в em, будут уменьшаться или увеличиваться вместе со шрифтом. Учитывая что размер шрифта, как правило, определяется в родителе, его, если нужно, можно изменить лишь в одном месте, что очень удобно.

Есть размеры, которые работают относительно экрана, это такие как vw, vh, vmin, vmax:

vw – 1% ширины окна

vh – 1% высоты окна

vmin – наименьшее из vw, vh

vmax – наибольшее из vw, vh

Их главное достоинство в том, что любые размеры, которые в них заданы, автоматически масштабируются при изменении размеров окна. Иными словами, уменьшается, например, высота окна браузера, текст будет уменьшаться. Текст выше если высота окна браузера будет расти.

Размеров в CSS больше, чем описано здесь, однако для начала новичку хватит тех, что описаны выше.

Выравниваем текст

Для выравнивания текста используется свойство text-align. Текст можно выровнять:

– по левому краю (left)

– по правому (right)

– по центру (center)

– одновременно выровнять по правому и по левому краю (justify).

Примечание: для абзацев значение left в text-align можно и не ставить, так как абзацы по умолчанию выровнены по левому краю.

Однако есть элементы, например, тег th (тег, который делает ячейку-заголовок таблицы), где по умолчанию стоит выравнивание по центру. Вследствие чего, если мы хотим в заголовке таблицы выровнять текст по левому краю, мы должны указать значение left.

Свойства CSS для шрифтов

Толщина, размер, красная строка и курсив текста

За толщину / жирность текста отвечает свойство font-weight, оно дает возможность сделать текст жирным или отменить жирность, так, например, заголовки уже жирные, по умолчанию.

Жирный текст делает значение bold

```
font-weight: bold;
```

Отменяет жирность значение normal

```
font-weight: normal;
```

Свойство font-style: italic; сделает текст курсивным, значение normal отменит курсив.

Размер шрифта указывает свойство font-size, например:

```
font-size: 30px;
```

Свойство text-indent делает красную строку для абзацев:

```
text-indent: 20px;
```

Декорируем шрифт

– text-transform: uppercase; – все буквы будут в тексте заглавные

– text-transform: capitalize; – первый символ каждого слова в предложении будет заглавным, прочие символы свой вид не поменяют

– text-transform: lowercase; – все символы текста станут строчными

– text-decoration: none; – отменить подчеркивание текста, в ссылках, например.

Межстрочный интервал

Межстрочный интервал – это расстояние между линиями текста, то есть белый промежуток между ними. Свойство line-height задает высоту линии текста.

Однако данное свойство не задает промежуток между строками текста, а задает высоту линии текста. Иными словами, реальный видимый промежуток между строками будет вычисляться так:

line-height – font-size = видимое расстояние между строками.

Например: font-size = 10px, line-height = 5 px.

Вычислим реальное значение межстрочного интервала:

line-height – font-size = 10px – 5px = 5px.

Данный интервал не обязательно должен быть целое число в пикселах, он может быть в виде дроби. Например:

font-size = 10px, line-height = 1.5.

Вычислим реальное значение межстрочного интервала:

line-height * font-size = 10px * 1.5 = 15px.

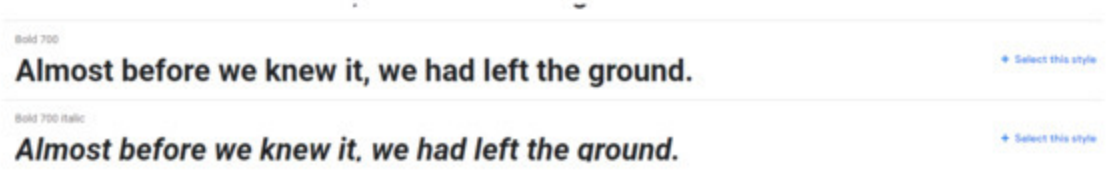
Вычислим видимый белый промежуток между линиями текста:

line-height – font-size = 15px – 10px = 5px.

Преимущество такого способа задания line-height в том, что при изменении размера шрифта автоматически будет меняться и межстрочный интервал.

Как подключать шрифты в CSS

– Заходим на сайт Google Fonts https://fonts.google.com/
– В поиске находим нужный нам шрифт (который указан в макете).
– Выбираем его толщину. например, Regular 400 и Bold 700, выбираем через +Select this style



– Слева появилась панель с link, который мы копируем и вставляем в наш HTML- файл, к примеру:

```
<link rel=«preconnect» href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300&display=swap" rel=«stylesheet»>
```

– Далее копируем строку для CSS и вставляем в тег body, например:

```
font-family: «Roboto’,sans-serif;
```

Специфика блочных и строчных элементов в CSS

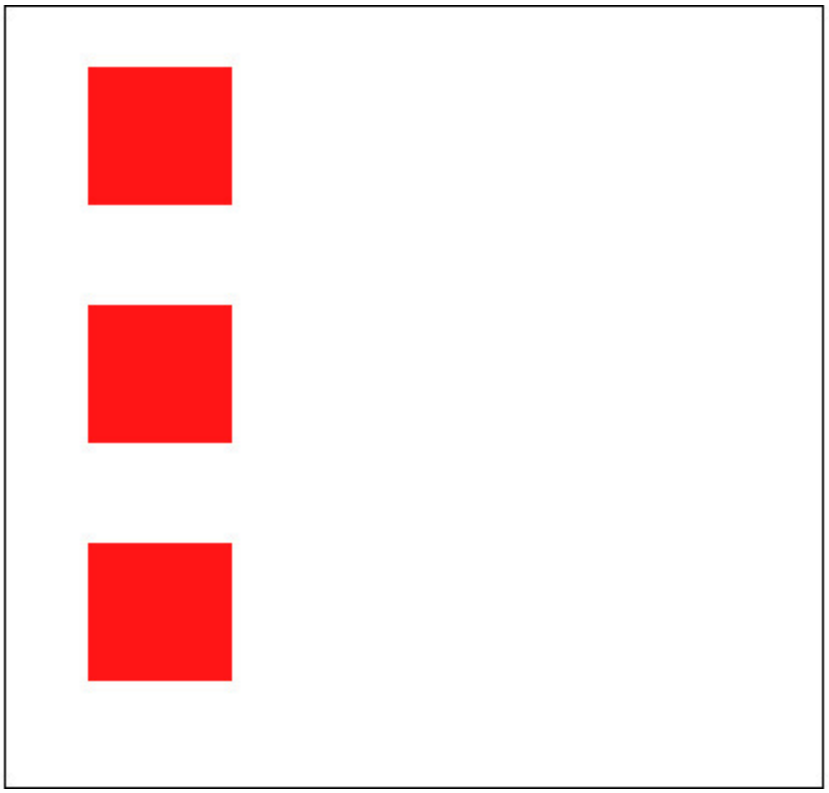
К примеру, text является блочным элементом, ему можно задать ширину и высоту.

```
.list {
width: 200px;
height: 300px;
}
```

Если у блочного элемента не задать ширину то он автоматически займет по ширине все доступное пространство, если не задать высоту, то она формируется содержимым.

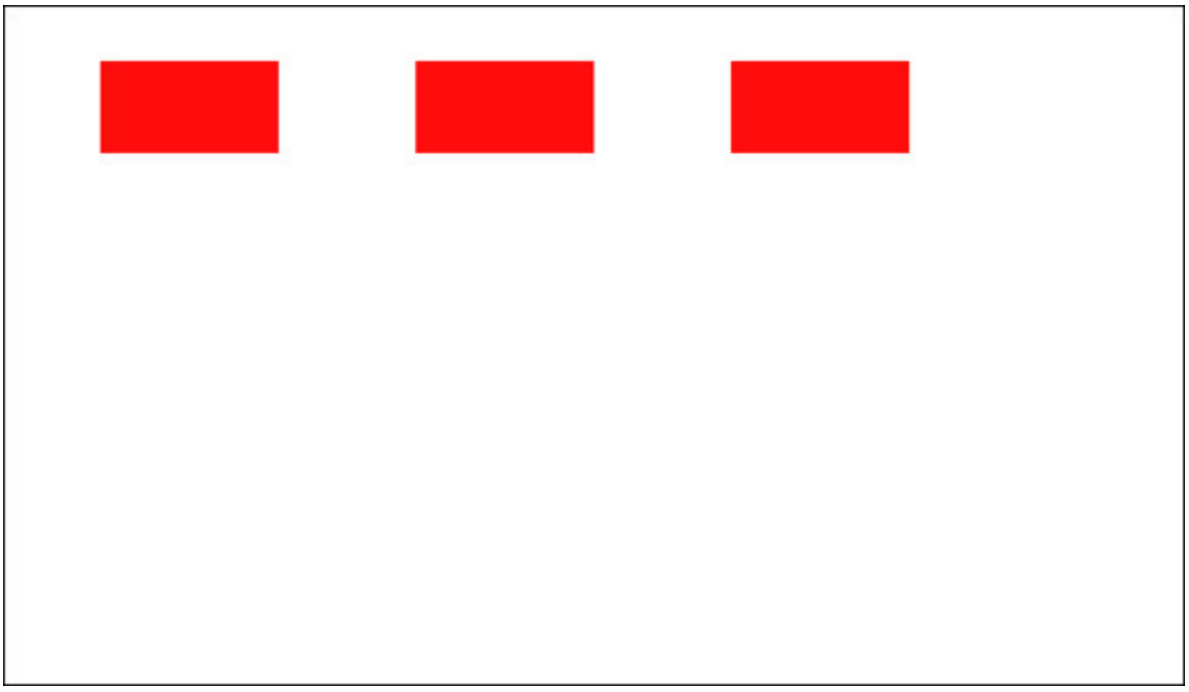
Если создать несколько элементов рядом, то каждый из них будет начинаться с новой строки.

```
<div class=«list»> text </div>
<div class=«list»> text </div>
<div class=«list»> text </div>
```



Несколько строчных элементов рядом наоборот встанут в ряд. Еще их особенность в том, что их ширина и высота всегда ограничены текстом элемента.

 text
 text
 text



Отступы между строчными элементами – символы пробела между словами текста.

Специфика строчно-блочных элементов

Свойство display

С помощью данного свойства можно сменить модель элемента. Через значение *block* данного свойства элемент можно сделать блочным, с помощью значения *inline* – строчным.

```
display: inline-block;
```

Строчно-блочные элементы сочетают особенности блочных и строчных элементов.

Если у данного элемента не задана ширина, то он его ширина сформируется содержимым, как у строчных элементов. Если у данного элемента не задана высота, то она также сформируется содержимым.

Если несколько данных элементов расположить рядом, то они выстроятся в ряд, как строчные элементы.

margin: auto; – как указывалось выше, центрирует элементы. К примеру:

margin: 20px auto; означает, что у элемента вверху и внизу будет отступ 20px, и он будет центрирован по горизонтали.

Примечание: так можно центрировать только блочные элементы, только по горизонтали, при этом обязательно должна быть им задана ширина.

Значение auto для margin не только центрирует элемент, но и дает возможность рассчитать отступ автоматически и сделать его максимально возможным. Например, если мы укажем, что левый отступ автоматический

```
margin-left: auto;
```

То данный элемент получит максимально возможный отступ слева и займет крайнее правое положение.

Однако, если одновременно задать правый отступ, элемент не будет прижат к краю

```
margin-left: auto;
margin-right: 30px;
```

flex-элементы CSS

Для того, чтобы получить flex-элементы, мы должны их родителю задать свойство

```
display: flex;
```

В итоге родитель останется блочным элементом, а вот его потомки будут flex-элементами.

flex-элементы, как блочные могут иметь ширину и высоту, margin и padding. Но в отличие от блочных, они по умолчанию выстраиваются в ряд внутри родителя.

Если родителю flex-элементов не задана высота, то по высоте он растянется содержимым, то есть его высотой будет высота потомков.

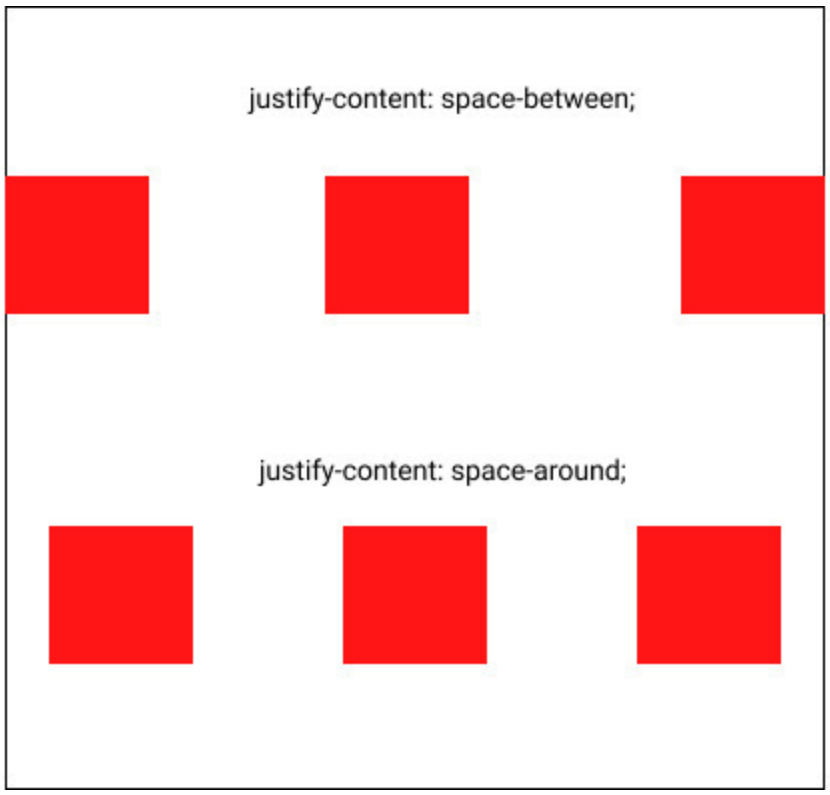
Если родителю не задать ширину, то по ширине он займет все доступное место.

Если родителю задать display со значением inline-flex, то он будет вести себя, как строчно-блочный элемент.

Свойством justify-content мы можем выровнять flex-элементы, так по для выравнивания их по центру пишем родителю

```
.parent {
display: flex;
justify-content: center;
}
```

justify-content: space-between; – равномерно распределит элементы по родителю, а расстояние между ними рассчитается автоматически.



justify-content: space-around; – аналогично space-between; распределит элементы, только первый и последний элементы будут отступать от края родителя. НО: расстояние между элементами будет в 2 раза больше, чем между родителем и крайним элементом.

justify-content: space-evenly; – расставляет элементы через равные промежутки.

Абсолютное и относительное позиционирование элементов CSS

Абсолютное позиционирование элементов дает возможность располагать элементы по указанным координатам страницы. К примеру, можно расположить элемент в позицию 300px сверху страницы и 500px слева. Элемент переместится туда, даже если там будут располагаться другие элементы, он просто встанет поверх их.

position: absolute;

position: absolute; задает абсолютное позиционирование. Помимо данного свойства, необходимо написать координату по вертикали и координату по горизонтали.

position: absolute;
top: 100px;
left: 50px;

Получается, что по вертикали задается отступ сверху или снизу, то есть top или bottom. Для горизонтали – это отступ слева или справа, иными словами left или right.

Относительное позиционирование задает position: relative;

Относительное позиционирование дает возможность сдвигать элементы относительно своего текущего положения на заданную величину. Причем все прочие элементы веб-страницы будут его воспринимать так, как будто он стоял там изначально. То есть при таком типе позиционирования элемент не выпадает из нормального потока.

С помощью свойств top, bottom, left, right задаем элементу смещение от текущего.

Позиционирование относительно родителя

Если элементу задать relative, а его потомку задать absolute, то данный потомок будет позиционироваться относительно родителя, а не относительно окна браузера. Обычно родителю не указывают координаты смещения, поэтому родитель остается на месте, а его потомки позиционируются относительно него.

Однако заметим, что если родительский элемент уже имеет позицию absolute, то потомки с absolute тоже будут позиционироваться относительно родителя, а не относительно окна браузера:

```
.parent {
position: absolute;
}
.child {
position: absolute;
}
```

При абсолютном позиционировании не обязательно задавать координаты. Если элементу просто написать position: absolute; то он станет абсолютно спозиционированным, однако останется там, где и стоял. Все прочие элементы будут вести себя так, как-будто данного элемента нет и могут налезть на него.

Если нужно создать блок, который расположен по центру экрана и отступает со всех сторон по 200px нужно:

```
.element {
position: absolute;
top: 200px;
right: 200px;
bottom: 200px;
left: 200px;
}
```

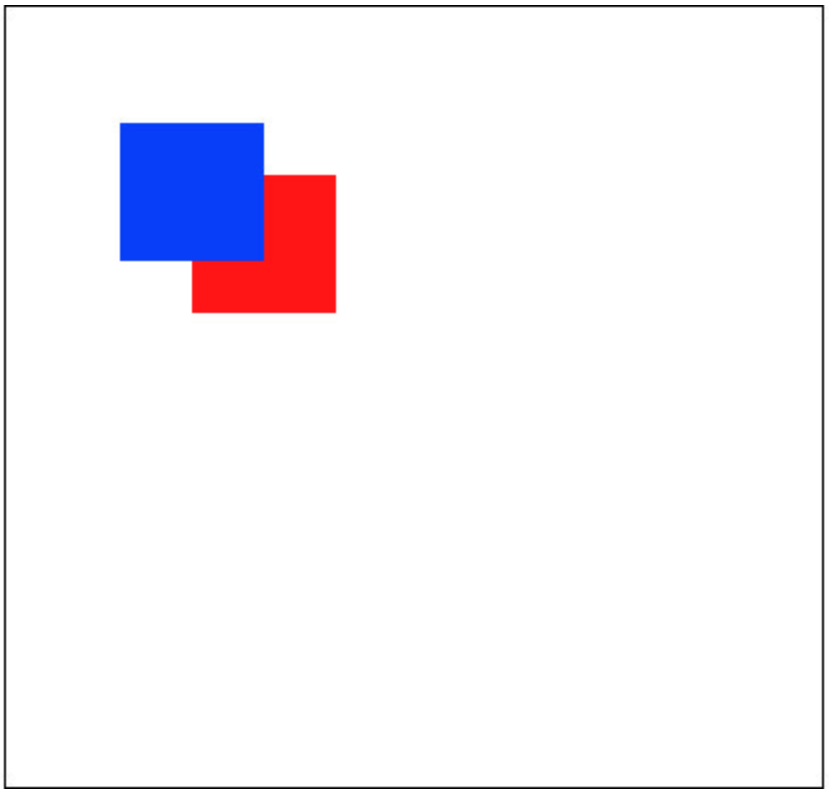
Наложение элементов

Порядок наложения элементов регулируется свойством z-index, значение которого может быть целым положительным или отрицательным числом, или нулем. Свойство работает лишь для элементов, у которых значение position: absolute или fixed или relative.

Правило наложения следующее: элемент, у которого больше значение z-index будет находится выше.

```
.element-1 {
z-index: 4;
}
.element-2 {
z-index: 2;
}
```

Если у элемента нет z-index, он не задан, это значит z-index равен 0. Если второй элемент имеет z-index со значением -1, то он будет ниже, чем элемент без z-index.

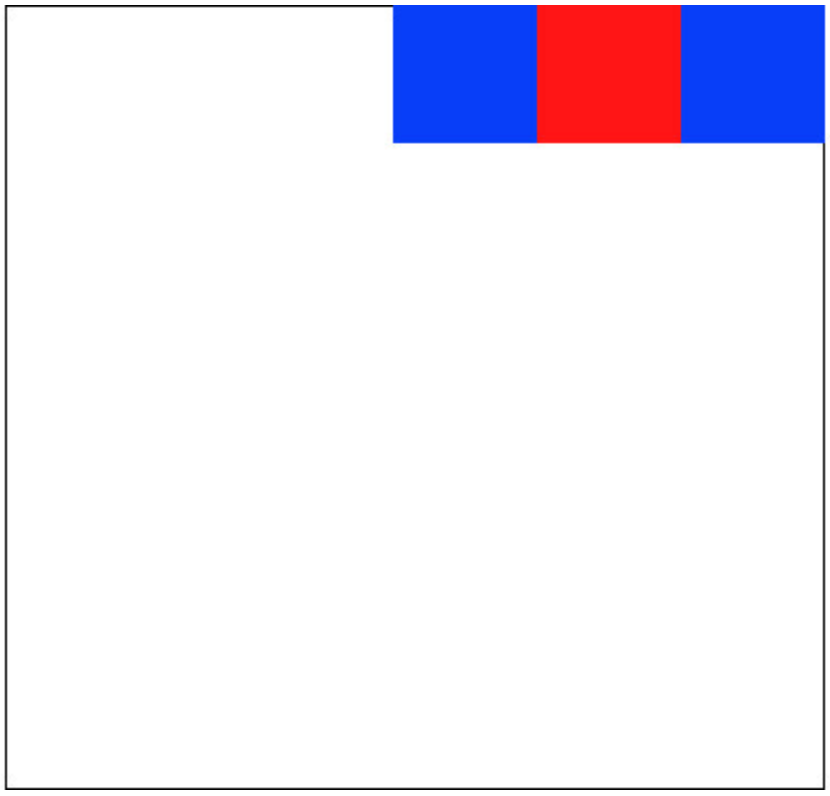


Размещение flex- элементов

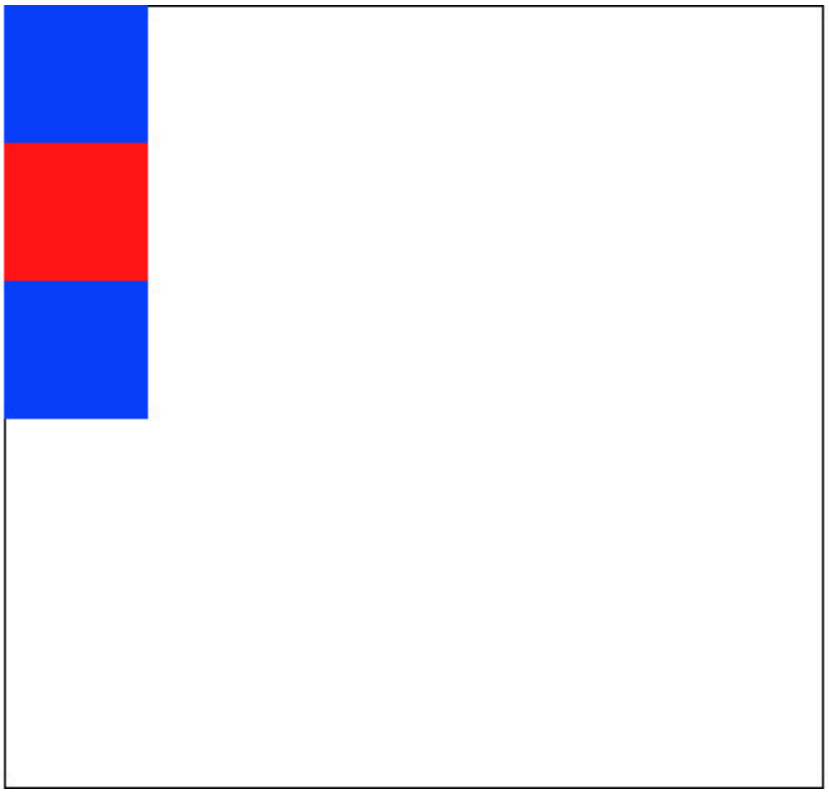
Свойство flex-direction регулирует выстраивание блоков, flex-элементов. По умолчанию данное свойство имеет значение row, то есть элементы выстраиваются в ряд.

```
display: flex;
flex-direction: row-reverse;
```

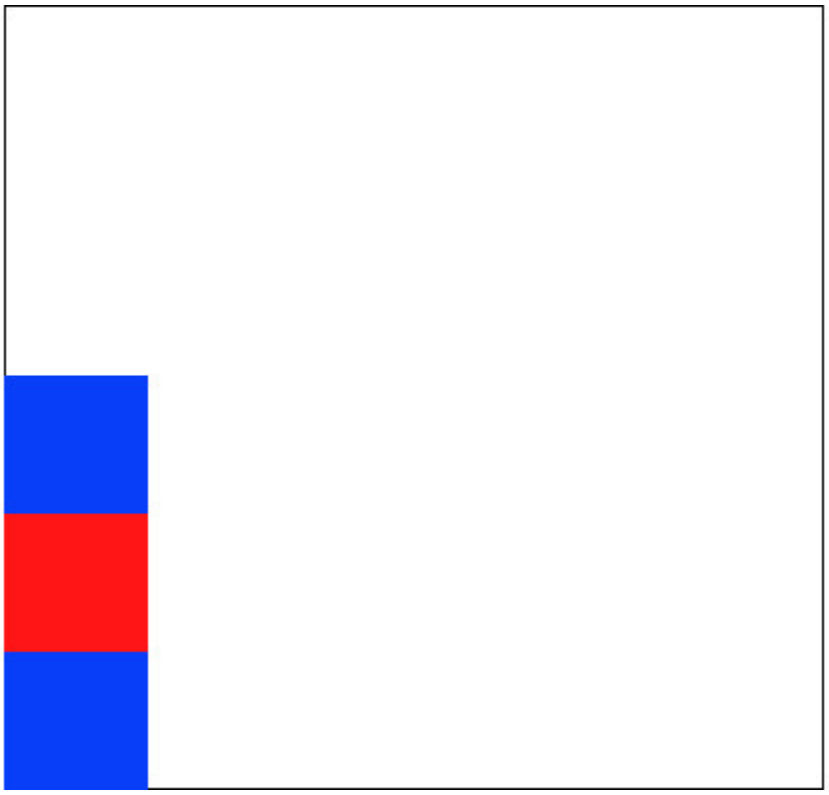
Значение row-reverse дает возможность выстраивать блоки в ряд, прижимав их к правому краю. Причем первый блок встанет как самый правый.



flex-direction: column; – выстраивает элементы в колонку.



flex-direction: column-reverse; – также выстраивает элементы в колонку, только колонка будет находится в нижней части веб-страницы и первый элемент будет самым нижним.



Приоритеты стилей в CSS

Иногда в коде, особенно в больших проектах, могут быть сформированы противоречия / конфликты стилей CSS.

Например, в нашем файле со стилями CSS мы в начале написали для класса. house цвет текста красный, а потом далее в коде написали зеленый. В итоге получился у нас один и тот же селектор, но с различным значением color, какой цвет будет у элемента в итоге?

```
.house {
color: red;
}
…
.house {
color: green;
}
```

В итоге цвет текста элемента будет зеленым. Почему?

Дело в том, что браузер выполняет код сверху вниз и сначала цвет элемента будет красным, далее он поменяется на зеленый.

Псевдоэлементы и псевдоклассы

Есть в верстке часть элементов и их состояний, которые не отображаются в структуре документа, однако им, например, также нужно применить некие стили в CSS.

Так, к примеру, нужно задать определенный стиль первой букве слова или определенной области, которая расположена под элементом и т. п.

Для того, чтобы получить доступ к данным «невидимым» элементам, используют специальные селекторы, которые называются псевдоэлементы и псевдоклассы.

Псевдоэлементы – это селекторы, определяющие область элементов, которая изначально отсутствует в структуре/дереве документа.

Псевдоклассы – являются селекторами, которые определяют состояние тех элементов, которые уже существуют, которое может меняться при конкретных условиях.

Иными словами, псевдоклассы определяют состояние элементов, уже существующих в документе, а псевдоэлементы являются искусственными, их изначально на странице не было.

Псевдокласс записывается как

E: link

Псевдоэлементов немного и они записываются так

E::first-line
E::first-letter
E::before
E::after

Существует официальная документация, в которой можно подробнее прочитать псевдоэлементы и псевдоклассы https://www.w3.org/TR/CSS/#indices

Медиа-запросы в CSS

Данный запросы являются базовыми инструментами для создания адаптивного сайта, так как они решают такую проблему как: контент веб-страницы подстраивается под конкретное устройство (смартфон, планшет и пр.), на котором данный контент отображается.

Их синтаксис

```
@media условия {
…
Стили CSS
…
}
```

Здесь условие может быть как одно условие, так и несколько условий. Если их несколько, то между ними будет стоять разделитель – некий оператор.

Условие 1 <Оператор> Условие 2…

Сами условия определяют какой тип или вид медиа-устройств будет отвечать конкретным стилям CSS, которые прописаны далее в квадратных скобках.

Получается, что медиа-запросы – это определенные логические конструкции, дающие возможность применить определенный набор стилей CSS к конкретному устройству. Иными словами, данные запросы говорят браузеры:

данное устройство – смартфон, к нему нужно применить вот такие стили CSS;

данное устройство – планшет, к нему нужно применить вот такие стили CSS и т. п.

Как работать с медиа-запросами

Итак, медиа-запросами (media queries) называют правила CSS, которые дают возможность управлять стилями элементов в зависимости от значений технических параметров устройств. Иными словами, их применяют тогда, когда необходимо применить различные CSS-стили, для разных устройств по типу отображения (для планшета, для смартфона и т.п.), а также для определенных характеристик устройства (ширина окна просмотра браузера и т.п.), или внешней среды (внешнее освещение и т.д.).

Таким образом, медиа-запросы – дают возможность адаптировать веб-страницу под различные устройства, это важный инструмент при создании веб-сайтов и приложений.

Медиа-запросы могут быть добавлены различными способами, напишу один из самых распространенных:

```
@media (max-width: 980px) {
.slider-nav {
display: none;
}
}
```

или может быть

```
@media (min-width: 980px) {
.slider-nav {
display: none;
}
}
```

min-width: – устройство должно иметь не менее …px.

max-width: – устройство должно иметь не более …px

Сперва через директиву @media указывается для какого экрана буду написаны стили. В данном случае прописано, что максимальная ширина экрана 980px. И потом в фигурных скобках пишутся стили для

нужных элементов. Данные стили подключатся, когда ширина экрана будет меньше или равна 980px. То есть до 980px включительно.

Брейкпоинтом называется точка останова или контрольные точки, существуют стандарты какими должны брейкпойнты в медиа-запросах: 575px, 767px и пр.

Отметим также такие моменты:

1. max-width и min-width можно применять в одном медиа запросе.
2. Медиа-запросы всегда следует писать в конце CSS-файла.
3. Для адаптивной верстки с помощью медиа-запросов нужно убрать все фиксированные значения width, например, в таких важных блоках, как контейнер меняем данное свойство на max-width или min-width.
4. Если вы делали верстку на флексах, то в колонках в CSS их свойство flex прописываем 50% для двухколоночного блока, 33,333% для 3х колоночного:

```
.colum {
flex: 0 1 33.333%;
}
```

Выравниваем блок по центру

Способов выровнять блок по центру множество, рассмотрим самые простые, но не менее эффективные из них.

Способ 1. Им мы пользовались уже в коде выше.

```
margin: 0 auto;
```

Свойство auto задает одинаковый отступ справа и слева, сверху и снизу у нас 0.

Способ 2.

```
{
width: 1000px;
position: absolute;
left: 50%;
margin-left: -500px;
}
```

Способ 3.

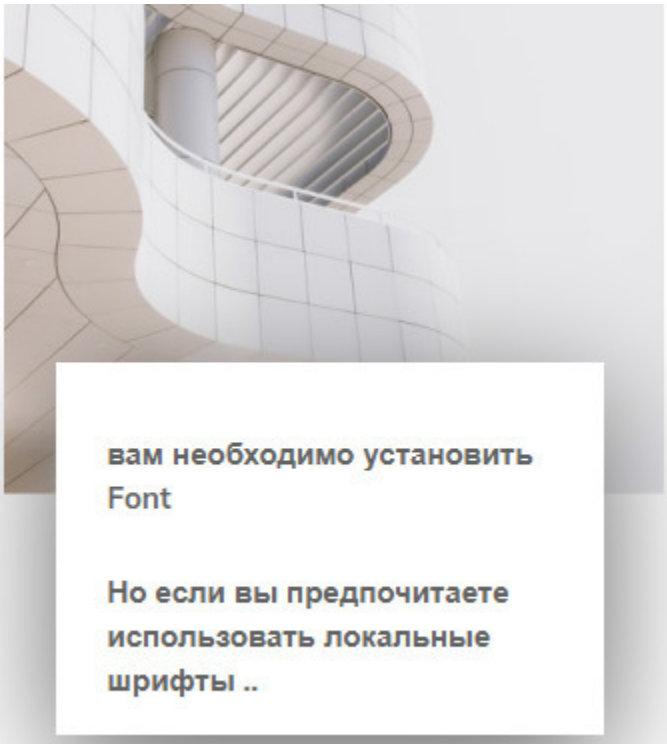
Если у блока указана ширина в %, а не px, то можно выровнять данный блок учитывая эти 100%:

```
{
margin: 0 50% 0 50%;
width: 100%;
}
```

Если блок имеет ширину 50%, то код будет выглядеть так:

```
{
margin: 0 25% 0 25%;
width: 50%;
}
```

Блок с текстом поверх картинки



в HTML

```
<div class=«col-4 card-1>
<img src="img/Image11.jpg» alt=«medal» class=«card-img»>
<div class=«text-card-11 card-all»>
<p class=«text-card»> вам необходимо установить Font </p>
<p class=«text-card-1»> Но если вы предпочитаете использовать локальные шрифты.. </p>
</div>
</div>
```

в CSS

```
.card-img {
position: relative;
}
.card-all {
position: absolute;
width: 274px;
height: 176px;
background: #FFFFFF;
box-shadow: 0 17px 65px;
margin-top: -70px;
}
```

Если нужно сделать блоку тень

Если нужно сделать блоку тень, как указано на рисунке выше, воспользуемся box-shadow. Функция box-shadow добавляет тень к изображениям. Ее синтаксис:

box-shadow (<сдвиг по x> <сдвиг по y> <радиус размытия> <цвет>);

где: <сдвиг по x>. – задает радиус размытия тени, смешение по ширине. Чем больше это значение, тем сильнее тень сглаживается, становится шире и светлее. Если сдвиг по оси x не задан, по умолчанию устанавливается равным 0, тень при этом будет чёткой, а не размытой.

<сдвиг по y> – сдвиг по высоте.

<цвет> – цвет тени в любом доступном CSS формате, по умолчанию тень чёрная, это необязательный параметр. Например:

box-shadow: 0 17px 65px;

Разное

Декоративные линии

Если в проекте есть декоративные горизонтальные линии, как, например, на рисунке

В HTML пишем код

```
<hr class=«footer-line» color=«red» size=«1»>
```

Разделяем предложение

Если нужно разделить, например, в заголовке предложение на несколько строк, перед словом, которое нужно поставить с новой строки пишем одиночный тег
.

<h1> Kelvin
 Designer & Developer </h1>

Блок на всю ширину экрана

Если нужно, чтобы блок занял всю ширину экрана, ему надо задать ширину 100%:

```
.element {
position: fixed;
width: 100%;
height: 100px;
}
```

Кратко основы jQuery

Цель jQuery – повысить функциональность веб-страниц с минимальным выполнением монотонной работы. Данная библиотека – библиотека JavaScript – это набор готовых функций, которые помогают делать некоторые вещи проще и удобнее, чем если это сделать в чистом JavaScript.

Для начала работы с библиотекой, ее необходимо скачать с официального сайта https://jquery.com/download/. Далее необходимо положить полученный файл в папку с проектом, над которым мы работаем (наш сайт) и подключить его в HTML через код

```
<head>
<meta charset=«utf-8»>
<title> </title>
<script src=«путь к файлу с jQuery»> </script>
</head>
```

Работа с библиотекой jQuery реализуется через универсальную функцию $, пишем

```
$ (параметр)
```

Набор элементов jQuery – это группа выбранных элементов. jQuery дает возможность применять полезные функции прямо к набору, в виде методов. Почти все методы данной библиотеки возвращают набор, к которому данный метод был применен, что дает возможность формировать цепочки методов любой длины.

Например, получим все элементы с классом. house, поставим им зеленый цвет, и сменим их текст на «здесь живет новый кролик»:

```
$ (’house’).css (’color’, 'green').html («здесь живет новый кролик»);
```

Если мы видим код $elems.html («меняем текст на новый»); здесь означает $elems, что в данной переменной лежит elems группа

элементов, массив, и к данной переменной здесь применяются методы jQuery. В данном случае используется метод. html (), с его помощью можно поменять текст элемента.

Также с помощью метода. html () можно вывести текст на экран:

```
alert($('.house').html ());
```

Однако на экране будет выведен текст первого элемента с классом. house, чтобы вывести текст всех элементов данного класса, надо применить дополнительно метод. each ().

.each () – это специальный метод, дающий возможность применить определенную функцию для всех элементов набора jQuery. Причем внутри данной функции есть возможность разделить элементы и поступить с каждым из них по-разному.

Иными словами, данный метод является циклом, благодаря которому можно перебрать все элементы, которые были найдены. Ссылка на элемент, по которому проходит цикл будет лежать в this.

Работа со свойствами CSS в jQuery имеет свои особенности. Так, если нужно изменить лишь одно свойство, оно указывается через запятую.

Если нужно изменить несколько свойств определенного элемента, то свойство и его новое значение пишутся через двоеточие, а сами свойства следует писать в кавычках, иначе в самом jQuery они пишутся немного иначе и можно запутаться.

Как говорилось выше, в данной библиотеке можно писать любой длины цепочки, данная длина не ограничена, и в данной цепочке писать столько методов подряд, сколько необходимо.

Отмечу, что jQuery поддерживает все стандартные селекторы CSS и псевдоклассы и псевдоэлементы CSS. Помимо этого jQuery есть некоторые другие вещи, которых нет в CSS. Например, псевдокласс: header выберет одновременно все заголовки от h1 до h6 или псевдокласс: contains, который выбирает элементы по наличию в них конкретного текста.

```
$ (’p: contains («выбрать строку») ’;
```

Данная команда выберет все элементы, где будет фраза «выбрать строку», если данная фраза присутствует внутри элемента или внутри

одного из его потомков.

Не обязательно изучать в данный момент все возможности jQuery, однако хотя бы понимать основы синтаксиса и что тот или иной метод означает, та или иная команда делает можно изучить, хотя бы выборочно, а лучше уметь практически применять некоторые возможности данной библиотеки, например, знать как прикрутить анимацию с помощью jQuery-плагина Fancybox.

Прикрутим анимацию и видео с помощью jQuery-плагина Fancybox

Плагин Fancybox 3 – это универсальный плагин, который использует возможности библиотеки jQuery для быстрой вставки медиа-контента на сайте в виде галереи. Иными словами, с его помощью можно выводить на сайт всплывающие различные красивые окна, видео.

Как пользоваться?

– Проходим по ссылке http://fancyapps.com/fancybox/3/

– Скачиваем архив с исходниками хода, нажав кнопку Download

– Далее следовать быстрой инструкцией с быстрым стартом Quick start:

заходим в папку с кодом, которую скачали, находим в папке dist файл с названием jquery. fancybox. min. css и jquery. fancybox. min. js.

– Закидываем файл jquery. fancybox. min. css в папку CSS и файл jquery. fancybox. min. js в папку JS нашего проекта

– Далее подключаем библиотеку jQuery в наш HTML-файл: указанную ниже строку вставляем в самый низ HTML-страницы, перед закрывающим тегом </body>.

```
<script
src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.min.js">
</script>
```

Это делается для того, чтобы анимация подключалась именно в конце.

– Далее следующей строкой подключаем плагин Fancybox:

```
<script src=«jquery. fancybox. min. js»> </script>
```

или если он у вас лежит в папке JS

```
<script src=«JS/jquery. fancybox. min. js»> </script>
```

– Далее вверху перед подключением нашего CSS в теге <head> подключаем плагин:

<link rel=«stylesheet» href=«CSS/jquery. fancybox. min. css»>

– Заходим на сайте плагина в раздел Документация http://fancyapps.com/fancybox/3/docs/ в раздел Video, там указан код как подключить видео к нашему сайту.

Там указано, что нужно к нашему тегу <a>, где указано Смотреть видео / Смотреть трейлер / Смотреть… необходимо подключить data-fancybox:

<a data-fancybox href=«#» class=«play»> Смотреть видео

При этом в href вместо затычки # нужно поставить ссылку на видео, которое мы хотим подключить.

Делаем переключение в меню

– Подключаем на HTML-странице JavaScript:

```
<script src=«js/main. js»> </script>
```

– в папке js в файле под названием main. js пишем код:

```
$(document).ready (function () {
console. log («Страница загрузилась»)
});
```

– Проходим в браузер на наш сайт-проект и через F12, и вкладку Console проверяем, подключился ли наш JavaScript. Если на странице в данной вкладке стоит Страница загрузилась, значит наш JavaScript правильно подключен.

– В нашем проекте на HTML-странице находим наши пункты меню, к примеру, у меня элементы с меню имеют класс menu-item, соответственно в файле main. js пишем код:

```
$(document).ready (function () {
let tabsItem = $ (». tabs-item’);
console. log (tabsItem); / проверяем действительно ли наши элементы меню появились в консоле
});
```

– Далее пишем код:

```
$(document).ready (function () {
let tabsItem = $ (». tabs-item’);
tabsItem. on (’click’, function (event) {
event.preventDefault ();
let activContent = $(this).attr (’href’);
$('.visible').toggleClass (’visible’);
$(activContent).toggleClass (’visible’);
});
```

```
});
```

Данный выше код означает, что мы отслеживаем событие по клику на пункты нашего меню.

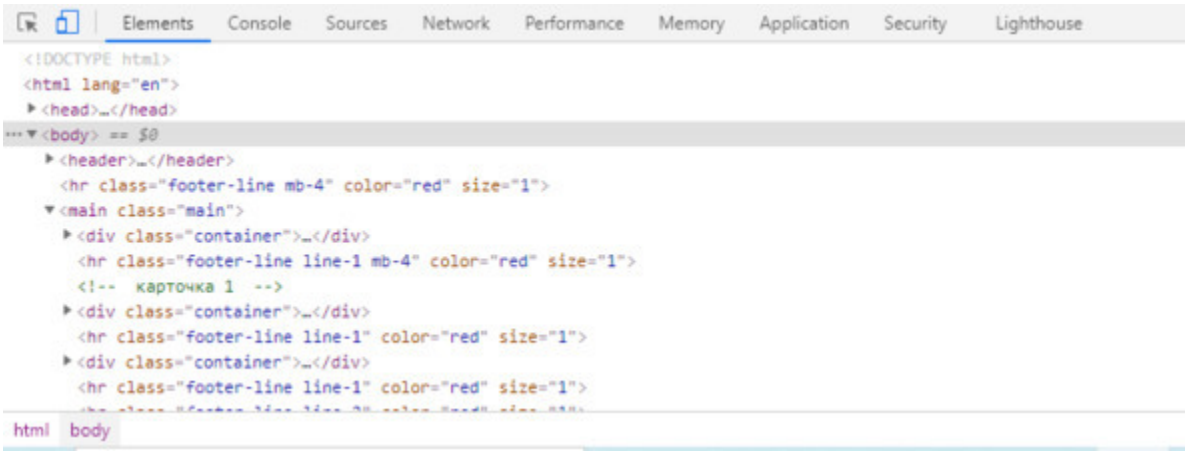
Полезное

Как работать с панелью разработчика

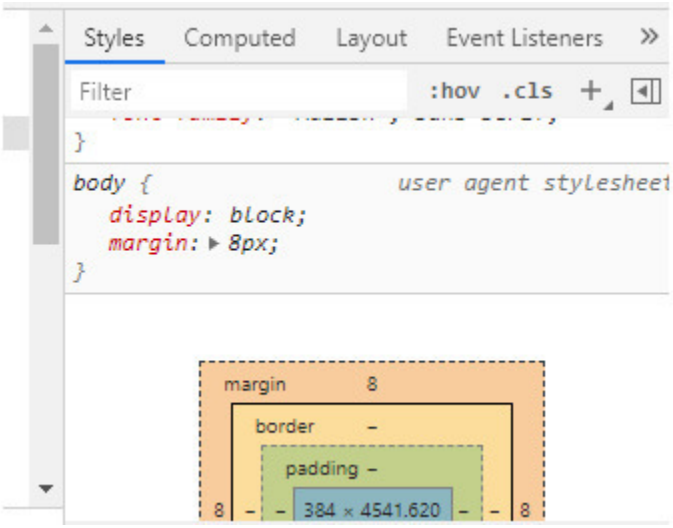
Панель или консоль разработчика в браузере является отличным инструментом для работы над версткой, с помощью данной панели можно быстро просмотреть HTML код страницы и CSS стили, выявить в данном коде ошибки, проверить сайт на адаптивность.

Открывается данная панель клавишей F12 (в некоторых браузерах может открываться иначе).

По умолчанию открывается вкладка «Elements», в ней содержится HTML-код открытой в браузере страницы, файлы CSS и JavaScript. Их содержимое можно посмотреть, если кликнуть на название файла.



В отдельной колонке содержатся CSS-стили. По умолчанию это стили всей веб-страницы, но если выделить отдельный блок или элемент, то в колонке будет указан код конкретного элемента.



Над некоторыми блоками может быть написано «Inherited from». Это означает, что свойства данного элемента унаследованы от определенного родительского элемента.

Во вкладке «Elements» можно редактировать HTML. Необходимо кликнуть на элемент и выбрать «Edit as HTML». Все внесенные изменения будут видны на веб-странице сразу, как в HTML, так и в CSS. В отдельной колонке можно отредактировать конкретное правило, удалить его или добавить новое. Однако, чтобы данные изменения сохранить нужно переписать их в свой код, так как все данные изменения сбросятся после перезагрузки страницы.

Вследствие чего панель разработчика следует использовать только для мелких изменений.

Во вкладке Network можно узнать сколько времени заняла загрузка страницы, какие ресурсы подключились или не подключились к ней. При первом открытии вкладка может оказаться пустой, тогда нужно перезагрузить страницу.

В ней можно отслеживать все запросы, которые отправляются к серверу во время загрузки текущей страницы, получить ответы сервера в итоге данного запроса.

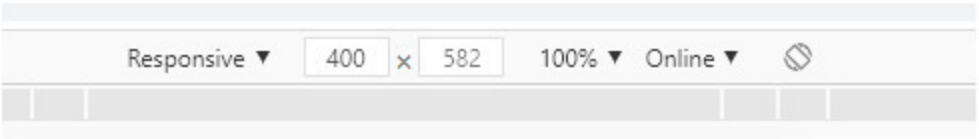
В данной вкладке можно, например, найти ответ на вопрос Почему посетитель отправляет личное сообщение в чате, а оно не отправляется? и т. п. Ответ от сервера можно получить во вкладке Preview данного запроса.

Вкладка Console отображает информацию об ошибках в коде, также сюда можно ввести команду, и она выполнится. Если, например,

на веб-странице не подгрузились шрифты, картинки или стили, то именно в данной вкладке будут выведены сообщения об ошибках.

Информация из вкладки Console очень полезна, она дает возможность сразу отследить возникновение конфликтов в скриптах при загрузке страницы, ошибки. Текст ошибки сообщает какого типа данная ошибка, место ее возникновения, указывается js-файл и место в данном файле, где находится ошибка. В итоге данная вкладка помогает в решении ошибок.

Посмотреть мобильную версию сайта можно через кнопку Toggle device toolbar. Сверху можно регулировать ширину вьюпорта, масштаб, плотность пикселей, выбрать мобильное устройство или десктоп, поворот экрана.



Отключается мобильная версия повторным нажатием Toggle device toolbar.

Во вкладке Sources показываются все подключенные к странице ресурсы. Чаще всего данная вкладка используется при отладке кода.

Некоторые приемы для работы с панелью разработчика

Если нажать на {} можно изменить форматирование исходного кода и вернуться к нормальному виду.

В HTML выбрать элемент внутри DOM и два раза кликнуть на него. Закрывающиеся тэги будут автоматически отредактированы. Все изменения будут сразу же отображены на веб-странице.

Аналогично предыдущему можно редактировать и CSS.

Можно быстро производить поиск в исходном коде с помощью комбинации Ctrl + Shift + F в HTML.

Можно быстро производить поиск в исходном коде с помощью комбинации Ctrl + F в CSS.

Если нажать комбинацию Ctrl + O можно перейти к нужному номеру строки в коде через специальный синтаксис, к примеру,:300:10, это значит, что переход произойдет на строку 300 и колонку 10.

Если нужно одновременно ввести одно значение в нескольких местах или редактировать какое-то свойство, например, необходимо нажать Ctrl + Click и одновременно вводить информацию в нескольких местах.

Окно DevTools имеет три положения: с одной из сторон, внизу и плавающее. Через комбинацию Ctrl + Shift + D можно переключаться между ними.

Иногда нужно очистить cookies. На вкладке Resources найдите папку Cookies и удалите куки через контекстное меню.

Если нажать в CSS-стилях на нужный цвет в разметке, то откроется цветовая паллета, где для любого элемента можно выбрать подходящий под дизайн цвет. А также с помощью пипетки можно посмотреть цвет любого элемента на странице.

Если нажать Console, затем Emulation, далее Sensors можно получить эмуляцию сенсорного экрана.

Если нажать Shift + Click на блоке с цветом, то можно поменять формат, переключаться между различными отображениями цветов: RGBA, HSL и пр.

Что такое Git

Это программа, которая устанавливается на компьютер, ее официальный сайт https://git-scm.com/download, нужно только выбрать свою версию операционной системы.

Это консольная программа, системы контроля версий, которая следит за изменениями в любой папке на компьютере. Смысл данного контроля в том, что если мы захотим откатиться к определенным изменениям, если стало понятно, что сейчас сделали что-то не то, мы может откатить изменения в тот конкретный момент времени, который нам нужен.

В Git можно создавать ветки, в которых можно сохранять текущее состояние положения дел проекта, а также можно данную ветку протестировать.

Например, мы можем создать отдельную ветку сайта, там протестировать какие-то новые для него блоки, элементы. Если данные элементы работают как нужно, можно данные нововведения слить с базовой версией сайта.

Git позволяет работать с удаленными серверами, репозиториями (хранилище, где хранятся файлы). Иными словами, можно работать из любой точки мира, не зависеть от определенного компьютера, рабочего места.

Плюс Git позволяет работать над одним и тем же проектом команде разработчиков.

Кратко о CSS Grid

Grid – это пересекающийся набор горизонтальных и вертикальных линий – столбцы и строки, которые составляют сетку.

CSS Grid дает возможность при создании макетов, легко создавать гибкие сетки сайтов. В будущем он, наверняка, будет использоваться повсеместно. CSS Grid схож с Flexbox, однако в отличии от Flexbox, он может одновременно работать в двумерных измерениях (столбцы и строки).

Также замечу, что CSS Grid не заменяет flex-box, наоборот это два разных инструмента для различных целей и задач. Однако они очень хорошо работают вместе, так flex может быть внутри сетки и наоборот.

Простейшая сетка выглядит в HTML так:

```
<div class=«wrapper»>
<div> 1 </div>
<div> 2 </div>
<div> 3 </div>
</div>
```

В CSS Grid 2 ключевых объекта:

класс wrapper – родительский, он оборачивает все внутренние, составляющие его блоки

items – дочерние (это непосредственно элементы).

Родительский объект является сеткой, а все внутренние элементами являются ее наполнением. Чтобы div class=«wrapper» стал сеткой, ему надо задать соответствующее значение grid:

```
.wrapper {
display: grid;
}
```

Чтобы сделать сетку двумерной нужно указать параметры строк и колонок с помощью grid-template-row и grid-template-colum

```
.wrapper {
```

```
display: grid;
grid-template-columns: 100px 100px 100px;
grid-template-rows: 50px 50px
}
```

grid-template-rows – это ряды, пространство между двумя последовательными горизонтальными линиями, grid-template-columns – колонки, пространство между двумя последовательными вертикальными линиями. В примере выше у нас 3 колонки и два ряда.

Или можно записать тоже самое, но немного по другому:

```
.wrapper {
display: grid;
grid-template: 50px 50px / 100px 100px 100px;
}
```

Здесь сначала идут ряды, а через косую черту колонки.

Функция CSS minmax () определяет диапазон размеров, больший или равный минимальному и меньший или равный максимальному.

Зазоры между линиями рядов – row-gap, зазоры между линиями колонок – column-gap:

```
.grid-container {
row-gap: 5px;
column-gap: 8px;
}
```

Опять данный выше код мы может написать еще короче, причем также опять сначала пишем параметры для рядов, потом для колонок.

```
.grid-container {
gap: 5px 8px;
}
```

Также для того, чтобы указать начальную и конечную позицию элемента сетки внутри сетки, применяют такие свойства:

grid-row-start – определяет начальную позицию элемента сетки в ряду сетки, добавляя ряд, интервал или ничего (автоматически)

grid-row-end – указывает конечную позицию элемента сетки в строке сетки, добавляя строку, интервал или ничего (автоматически)

grid-column-start – указывает начальную позицию элемента сетки в столбце сетки, добавляя строку, интервал или ничего (автоматически)

grid-column-end определяет конечную позицию элемента сетки в столбце сетки, добавляя линию, интервал или ничего (автоматически).

Для того, чтобы получилась данная разметка сетки, пишем код

```
.item1 {
grid-column-start: 1;
grid-column-end: 3;
}
.item3 {
grid-row-start: 2;
grid-row-end: 4;
}
.item4 {
grid-column-start: 2;
grid-column-end: 4;
} и т. п.
```

Здесь, к примеру, item1 начинается в первой колонке, заканчивается в третьей и т. п.

Bootstrap

Bootstrap – это открытый и бесплатный HTML, CSS и JS фреймворк, с которым работает веб-разработчик для быстрой верстки адаптивных дизайнов сайтов и веб-приложений.

Данный фреймворк – это набор CSS и JavaScript файлов, которые сначала нужно просто подключить к странице, тогда станут доступны такие инструменты Bootstrap как колоночная система (сетка Bootstrap), классы и компоненты.

Как он работает данный фреймворк?

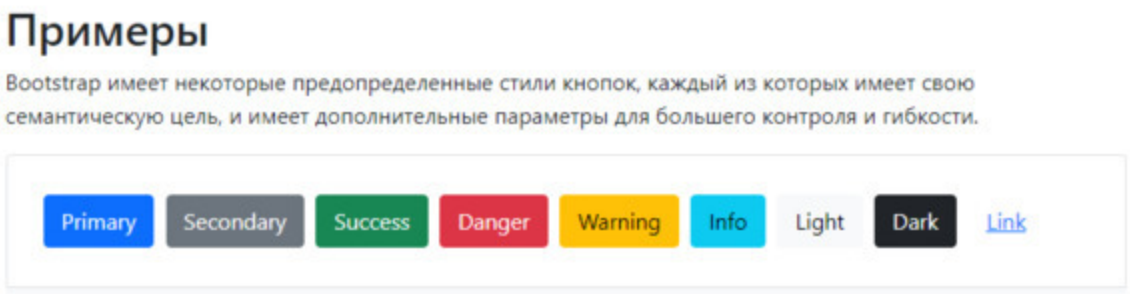
К примеру, создадим кнопку. Для этого достаточно к элементу button добавить два класса btn и btn-success или любой иной соответствующий выбранной нами кнопки.

```
<button type=«button» class=«btn btn-primary»> Primary </button>
```

или

```
<button type=«button» class=«btn btn-success»> Success </button>
```

Кнопки, учитывая официальную документацию Bootstrap 5 могут быть таких видов как

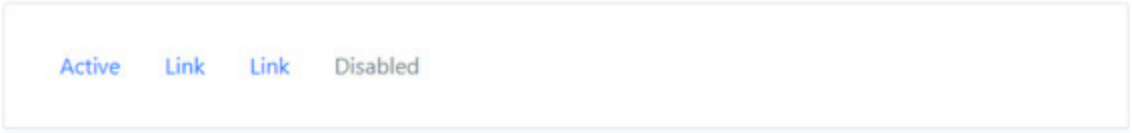


Указанная документация находится по ссылке https://bootstrap-4.ru/docs/5.0/components/buttons/

Если не нужен текст, который изначально указан в кнопке, можно добавить к ней класс. text-nowrap.

Далее, например, нужно создать навигацию, то для этого нужно только скопировать готовый HTML-фрагмент с указанной выше документации:

```
<ul class=«nav»>
<li class=«nav-item»>
<a class=«nav-link active» aria-current=«page» href=«#»> Active </a>
</li>
<li class=«nav-item»>
<a class=«nav-link» href=«#»> Link </a>
</li>
<li class=«nav-item»>
<a class=«nav-link» href=«#»> Link </a>
</li>
<li class=«nav-item»>
<a class=«nav-link disabled» href=«#» tabindex="-1» aria-disabled=«true»> Disabled </a>
</li>
</ul>
```



Здесь базовый компонент класса. nav создан на флексбоксе, что обеспечивает хорошую базу для создания всех типов навигационных компонентов. В него входят несколько стилей, которые «перебивают» остальные (для работы со списками), добавлены паддинги ссылок для увеличения «зоны клика», и базовые стили «выключенных» состояний.

Причем можно в двух вариантах создать навигацию, в виде списка, как указано выше, или в виде ссылок:

```
<nav class=«nav»>
<a class=«nav-link active» aria-current=«page» href=«#»> Active </a>
<a class=«nav-link» href=«#»> Link </a>
<a class=«nav-link» href=«#»> Link </a>
<a class=«nav-link disabled» href=«#» tabindex="-1» aria-disabled=«true»> Disabled </a>
</nav>
```

При этом можно, например, выбрать выравнивание по левому краю, по центру или по правому краю https://bootstrap-4.ru/docs/5.0/components/navs-tabs/

Для адаптивности Bootstrap включает шесть контрольных точек по умолчанию, для быстрого реагирования.

	Контрольные точки	Инфикс класса	Размеры
	X-Small	*Нет*	<576px
	Small	sm	≥576px
	Medium	md	≥768px
	Large	lg	≥992px
	Extra large	xl	≥1200px
	Extra extra large	xxl	≥1400px

Так как Bootstrap разработан для мобильных устройств, в документации находятся несколько медиа-запросов для создания разумных точек останова. Данные точки главным образом основаны

на минимальной ширине области просмотра и дают возможность масштабировать элементы при изменении области просмотра.

Итак Bootstrap состоит из:

– инструментов для создания макета (оберточные контейнеры, мощная система сеток и т.д.);

– классов для стилизации базового контента: текста, изображений, кода и пр.

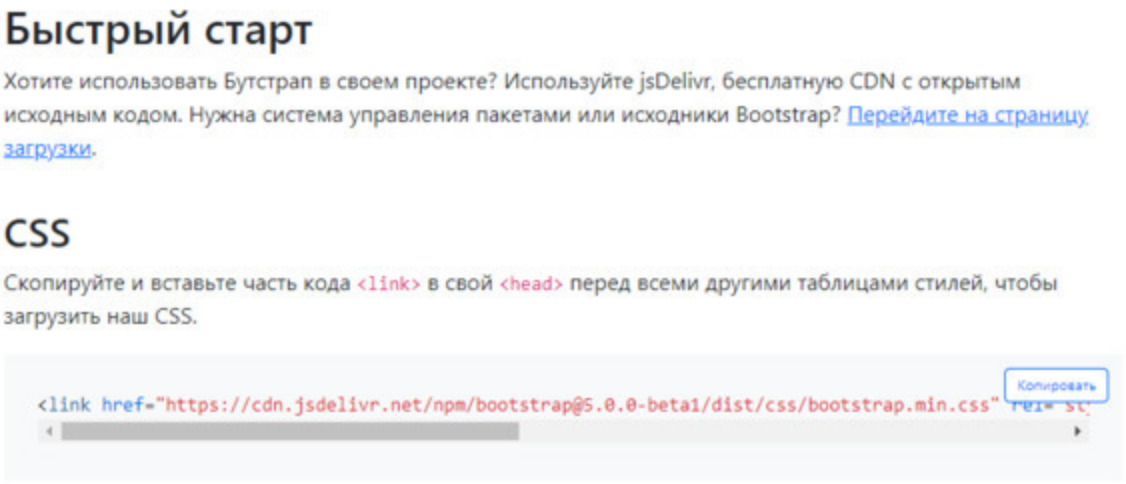
– готовых компонентов: кнопки, формы, различные навигационные панели, слайдеры, модальные окна и пр.;

– утилитных классов для решения такие классических, часто возникающих задач как: выравнивание текста, отображение и скрытие элементов, задания цвета и т. д.

Как учить Bootstrap?

1. Устанавливаем фреймфорк https://bootstrap-4.ru/docs/5.0/getting-started/download/

Скопируйте и вставьте часть кода <link> в свой <head> перед всеми другими таблицами стилей, чтобы загрузить CSS из Bootstrap.



– Из папки, которая была скачена в нашу папку CSS в проекте переносим файл bootstrap-grid. min. css

– Подключаем данный файл в HTML перед link, который подключил CSS:

<link rel=«stylesheet» href=«css/bootstrap-grid. min. css»>

Аналогично подключается JavaScript https://bootstrap-4.ru/docs/5.0/getting-started/introduction/

2. Далее изучаем сетку, нужно научиться ее использовать для создания макетов страниц и разметки блоков.

3. Потом необходимо изучить компоненты и классы Bootstrap.

4. Когда мы знаем базу Bootstrap, то есть все перечисленное выше, можно перейти к верстке, на практике закрепить полученные знания, опробовать варианты и возможности работы с Bootstrap.

Полезные библиотеки и веб-ресурсы

Библиотека с анимациями – https://daneden.github.io/animate.css/

Сайт Github – http://github.com/

Google Fonts – большая коллекция шрифтов

Colorscheme – специализированный сайт, с помощью которого можно узнать название того или иного цвета, его цветовые модели https://colorscheme.ru/color-names.html

jQuery Code Snippets – плагин сниппетов. Сниппеты кода – это фрагменты кода, которые можно многократно использовать в проектах. Создаются они обычно для того, чтобы ускорить время разработки и не тратить на стандартные решения, «старый» код.

Compressor.io – сервис для оптимизации изображений. Работает без потери качества. Однако есть недостаток, изображения загружать можно только по одному.

Создание 3d обложек мэйкапы http://www.3dcoverdesign.ru/

Bootstrap https://bootstrap5.ru/

HowToCenterInCss – сервис, который на основе входных данных о разметке демонстрирует CSS-код для центрирования элемента (горизонтально и вертикально).

Валидность сайта, то есть соответствие его стандартам кода можно проверить:

– html на validator.w3.org/

– css на jigsaw.w3.org/css-validator

http://ft.vremenno.net/ – это шрифтотренажер, на котором можно посмотреть как будет выглядеть определенный шрифт на вашем сайте.

Fribbble – сервис, с которого можно скачать бесплатные PSD-макеты, его источник dribbble.com.

Cssdesk.com – очень полезный ресурс для начинающих верстальщиков, здесь можно написать код html и css и посмотреть как это будет выглядеть. Очень удобный ресурс для тренировки.

Font awesome – коллекция векторных иконок, их можно вставить на сайт, как шрифт. Иконки векторные, получается, что их можно масштабировать до любого размера, их легко подключить и легко настроить. Все свойства можно прописать в css, как для обычного шрифта: размер, цвет, тень и пр.

pxtoem.gamecoll.com – конвектор, который считает, например, сколько пикселей в em, в пунктах? и пр.

www.colorzilla.com/gradient-editor помогает с помощью css залить градиентом определенную область, при этом код генерируется как в формате css, так и в SCSS.

Полезные плагины для верстки

Emmet – помогает ускорить вёрстку. С помощью! и tab можно создать базовую структуру кода за 1 секунду, либо быстро создать вложенные теги.

Material Theme окрашивает код с разными акцентными цветами.

Live Server – применяют для того, чтобы сразу увидеть результат работы. Необходимо навести на строчку кода курсор, нажать на правую кнопку мыши, выбрать «Open with Live Server» и в браузере отобразится страница, она будет автоматически перезагружаться. Иными словами, любые изменения будут отображаться без обновления страницы. Если необходимо отключить плагин, сделать это нужно через «Stop Live Server»

CSS Peek – показывает всплывающее окошко с CSS во время работы над HTML, поэтому нет необходимости не искать требуемое свойство среди множества классов в большом CSS-файле.

Auto rename tag – полезен, если нужно переименовать теги в HTML. Если тег повторяется, при замене одного из них, прочие будут переименованы автоматически.

Path autocomplete – показывает возможный путь к файлу в кавычках. Вследствие чего нет необходимости искать папку вручную.

HTML CSS Support – данный плагин автоматически дополняет название ID или HTML-атрибута для определений, которые найдены в рабочей области, на которые ссылается link.

Prettier – плагин для выравнивания кода.

Что почитать?

– Дженнифер Нидерст Роббинс «HTML5, CSS3 и JavaScript. Исчерпывающее руководство»

– Дэвид Макфарланд «Большая книга CSS3»

– Хоган Б. «HTML5 и CSS3. Веб-разработка по стандартам нового поколения»

– Терри Фельке-Моррис «Большая книга веб-дизайна»

– Робин Никсон «Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5.»

– Эстель Вейл «HTML5. Разработка приложений для мобильных устройств.»

– Джон Дакетт «HTML и CSS. Разработка и дизайн веб-сайтов.»

– Бен Фрэйн «HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств.»

– Нейт Купер, Ким Джи «Как создать сайт. Комикс-путеводитель по HTML, CSS и WordPress.»

– Кит Грант «CSS для профи.»

Эберт Елена

Шпаргалки для начинающего верстальщика HTML/CSS

Читает вся страна