Курс по вёрстке(HTML) и CSS

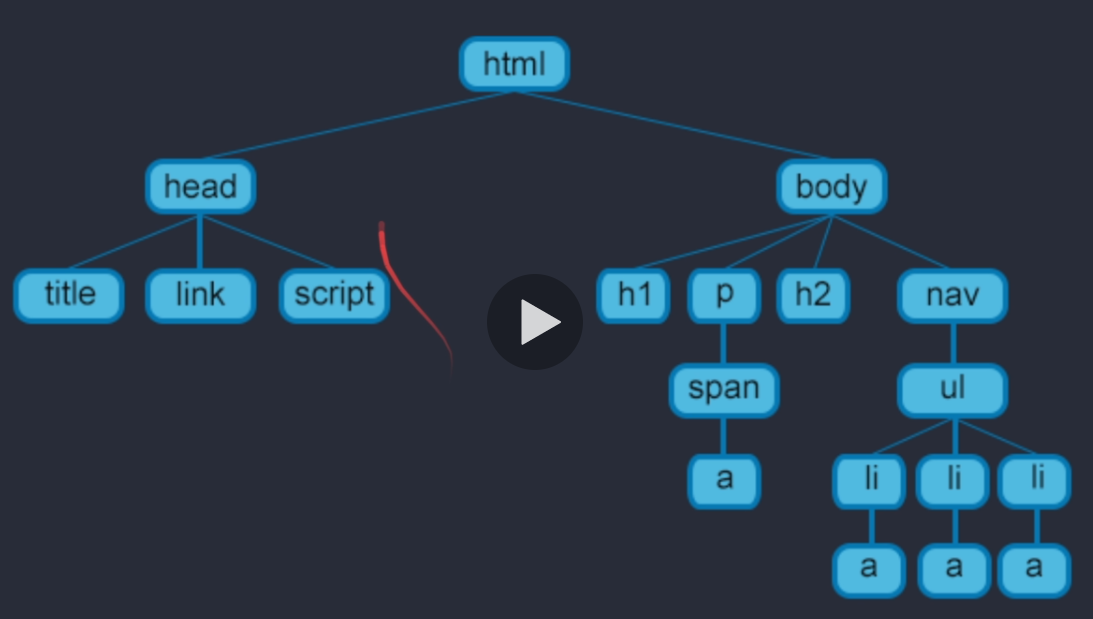
Раздел 1. Глава 2:

1. Ctr + F 🡪 поиск по всем файлам
2. Ctr + shift + H 🡪 поиск глобальный по файлам

Раздел 2

Глава №3

1. В тегах meta, находящихся в head находится информация для браузера и поисковых систем



Глава №4

1. Нажимаем ctr + shift + i и выходим в режим разработчика, где используя селектор получаем код любого элемента (переносимся к нему).
2. Так же в режиме разработчика можно посмотреть, как сайт будет отображаться на телефоне (иконка на верхней панели, так же можно выбрать модель телефона или самому создать)
3. ctr + u 🡪 посмотреть код страницы, более простой вариант, чем режим разработчика.
4. Блочный элемент занимает всю строку.

Глава №5

Заголовки и параграфы

1. Заголовки влияют на поисковую выдачу, в отличии от обычных блоков. Всего заголовков 6 видов от h1 до h6
2. Абзацы p нужны для выделения текста в абзацы с отступами.
3. Команда lorem + Tab заполняет строку текстом, что удобно для тестирования вывода сайта.

Глава №6

Ссылки и изображения

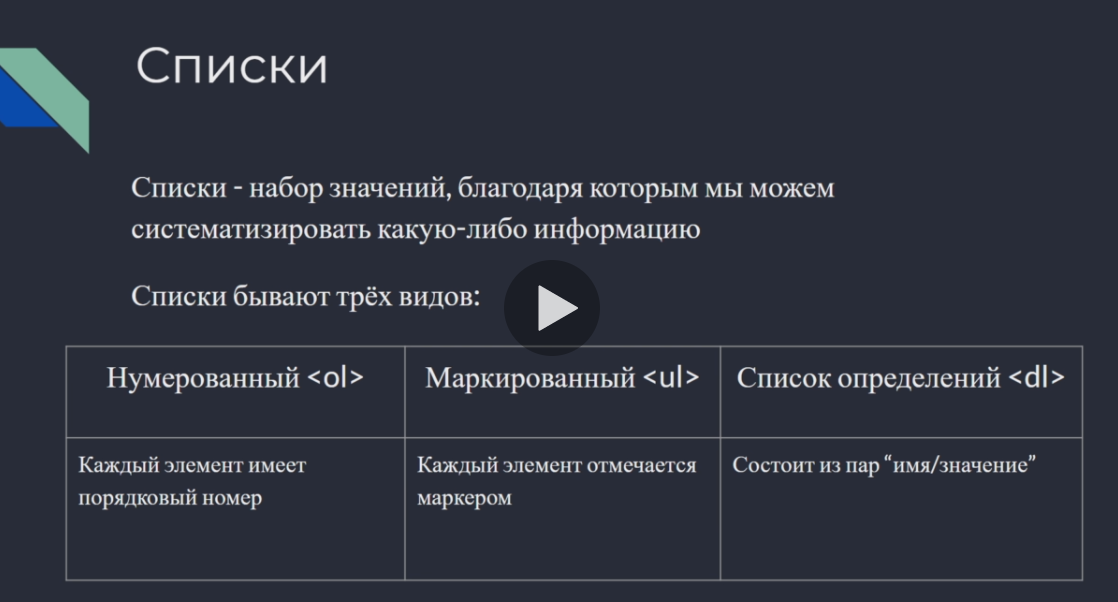
1. Ссылки помечаются тегом a, где обязательный параметр это href, который и содержит ссылку.
2. Внутри ссылок можно вставлять картинки, при нажатии на картинку сработает ссылка.
3. Тег img имеет url ссылку на изображение, и дополнительный параметр alt, который является текстом вместо изображения, если изображение не будет подгружено на страницу.

<a href="https://vk.com/feed"><img src="img.png" alt="Переход на главную страницу vk"></a>

1. Картинки не всегда имеют тэг img, они могут являться задним фоном какого-то div

Глава №7

Списки и таблицы



1. Списки можно вкладывать один в другой, что бы получить к примеру содержание страницы.

 <h1>Моя первая статья</h1>

        <ol>

                <li>

                        <span>первый раздел</span>

                        <ul>

                                <li>No chapter</li>

                        </ul>

                </li>

                <li>

                        <span>Second part</span>

                        <ul>

                                <li>First part</li>

                                <li>Second part</li>

                                <li>third part(now working)</li>

                        </ul>

                </li>

        </ol>

1. Тэг table, позволяет создавать таблицы на странице, но границы данной таблицы задаются с помощью css. Ключевые слова, для таблицы это:
2. <tr> 🡪 table row
3. <th> 🡪 table head(используется для создания заголовков)
4. <td> 🡪 table data(используется для внесения в таблицу общей информации)

<table>

                <caption>Перечень всего</caption>

                <tr>

                        <th>Номер</th>

                        <th>Название</th>

                </tr>

                <tr>

                        <td>1</td>

                        <td>Молоко</td>

                </tr>

</table>

Глава № 8-9

Формы

1. При работе с form, у нас имеется много внутренних тегов. Таких как: input.

У данного тега, есть аргумент type, который говорит системе каким типом элемента будет данное поле.

2. У тэга input, есть дополнительный параметр placeholder, подсказка пользователю внутри контейнера, что над вводить.

2.1 Так же можно использовать тэг label, для метки около данного поля.

<form action="#">

                <label for="">

                        Цвет

                        <input type="color" placeholder="Цвет">

                </label>

                <label for="">

                        Пароль

                        <input type="password" placeholder="пароль">

                </label>

        </form>

3. У тэга input, так же есть аргумент name, который даёт данному тэгу имя для использования в других тэгах, как label.

<form action="#">

                <label for="url\_path">URL</label>

                <label for="pass">Pass</label>

                <input type="url" name="url\_path" placeholder="url">

                <input type="password" name="pass" placeholder="pass">

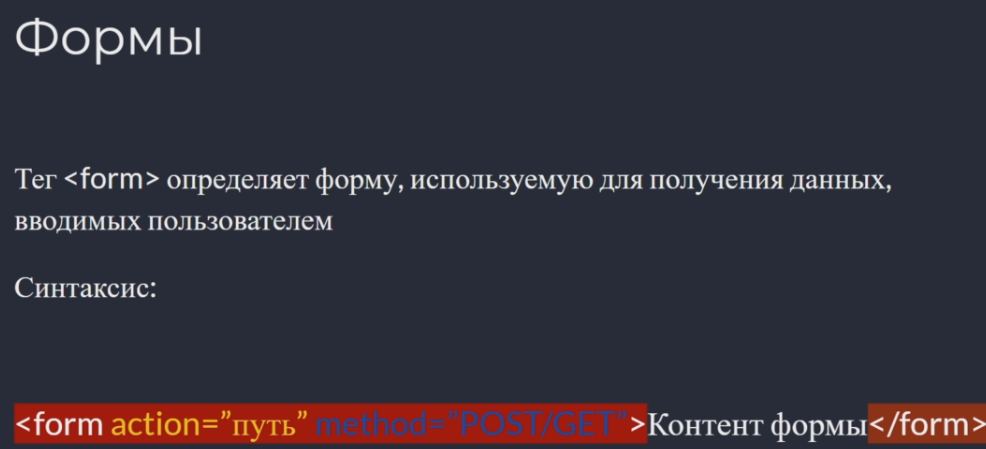
        </form>

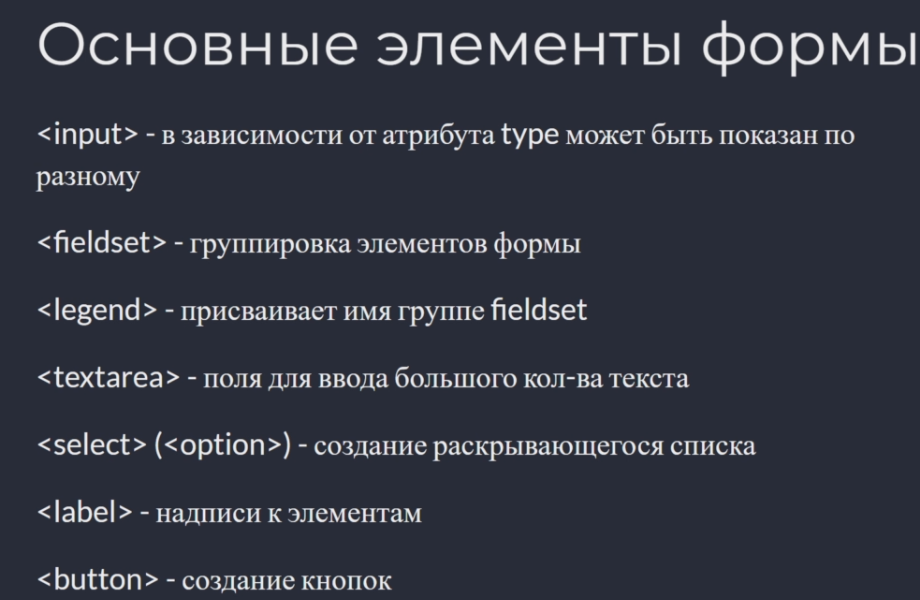
4. У тэга input, так же есть метод value, он меняет значение внутри поля на написанное, если это возможно (у checkbox не отобразится метод value), метод value имеет преимущество над методом placeholder и заменяет текст от placeholder текстом от value, так же текст от value надо удалять перед вводом новых значений, а в placeholder он автоматически пропадёт.

5. Тэг br переносит элементы после него на строку ниже

6. Тэг textarea создаёт большое текстовое поле, которое потом можно будет расширять.

<textarea name="txt" id="" cols="30" rows="10"></textarea>





Глава №10

Строчные и блочные элементы

1. Внутри блочных элементов могут находиться как блоки, так и строки, внутри строчных элементов могут находиться только строчные элементы

2. Блочные элементы занимают всю доступную ширину, строчные элементы занимают только область, выделенную им.

3. Для параграфа p, который является блочным элементом есть исключения, он не может содержать внутри себя другие блоки.

Глава №11

Теги div & span.

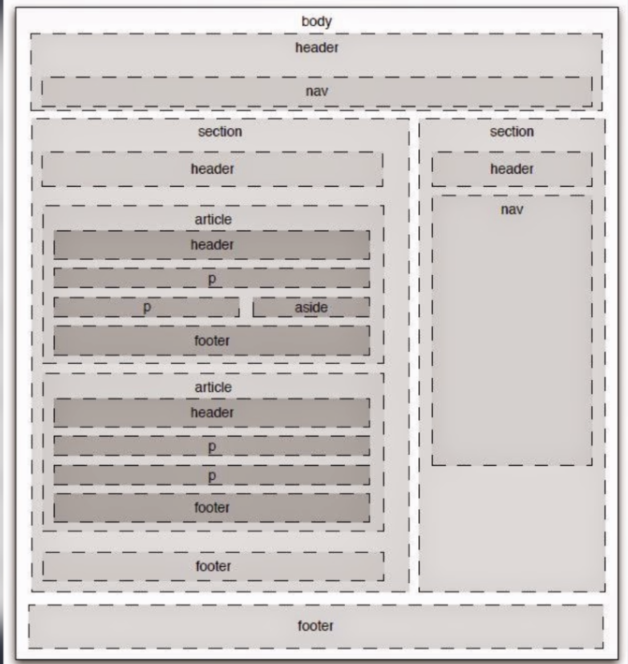
Семантические теги.

1. Для тегов div и span нет предустановленных стилей, мы их задаём сами.

1.1 div это блочный элемент

1.2 span это строчный элемент

2. Семантические теги в отличии от div изначально работают по определённой роли, как шапка сайта или footer или тело сайта. Данные теги появились в HTML5.



3. Семантические теги можно поделить как:

3.1 header 🡪 заголовок

3.2 nav(navigation) 🡪 блок для меню и информации

3.3 main 🡪 содержится главная информация

3.4 section 🡪 делим информацию по секциям

3.5 footer 🡪 подвал сайта(его нижняя часть)

Семантические теги нужны для более лучшего понимания структуры сайта и разбиения сайта на более понятные блоки.

4. Тэг span позволяет стилизовать часть информации, он не является блочным

Раздел № 3 “CSS”

Глава № 12 Основные понятия CSS

1. Для задания стилей существует три основных подхода, таких как:

1.1 Задать стиль в div

1.2 Задать стиль в том же файле, но в тегах style

1.3 Задать стиль в стороннем файле с расширением .css

<div style="width: 50px; height: 50px; color: red;">

           <h1>Second part</h1>

        </div>

        <style>

            .div1{

                color:yellow;

                background-color: red;

            }

        </style>

С помощью ссылки задаём стиль в стороннем файле с расширением css

.div3{

    color: green;

}

Глава №13

Атрибуты class и id

1.Можно задать цвет и параметры всем элементам на странице, к примеру, всем div или задний фон body.

2. Так же можно использовать семантические теги для задания параметров для всех элементов внутри них.

div{

    color:red;

}

body{

    background-color:orangered;

}

main{

    background-color: red;

}

3. Можно так же для каждого элемента указать class, и определять стиль только для данного объекта. Так же class можно переиспользовать для других объектов в отличии от id, который уникален для каждого объекта.

4. В одном объекте может быть несколько ссылок на class, которые будут совмещать все параметры обоих class. Приоритет повторяющихся настроек отдаётся более нижнему блоку.

<div class="div1 div4">

            <h1>First part</h1>

        </div>

5. Чтобы обратиться к объекту по id, используем синтаксис через #

#item2{

    color: red;

}

6.Теги для настройки стилей css имеют приоритет друг над другом, приоритет такой: id > class > div , то есть чем выше приоритет тега, тем выше приоритет выполнения повторяющихся настроек, если на один объект наложено несколько тегов стилей. У одинаковых тегов стилей, приоритет отдаётся тегу, который ниже.

Глава №14

Основные селекторы CSS

1. Можно обращаться к элементам не только по классам, но и по ссылкам, к примеру, мы можем с помощью особого синтаксиса [href], обратиться ко всем ссылкам или к одной из них, которую можно выделить по адресу или на что оканчивается ссылка (настроек много).

main.main1{

    color: red;

}

[href="https://vk.com/q2"]{

    color: orange;

}

/\* Ссылка содержит часть пути в кавычках \*/

[href^="https://vk.com/q"]{

    color: green;

}

/\* Ссылка содержит в конце символ \*/

[href$="r"]{

    color:red;

}

Глава №15

Шрифты и работа со стилями текста

1. Шрифты можно использовать в проекте на странице двумя способами, можно подгрузить их в свой проект и подключить или использовать сайт google fonts (или другие подобные) и шрифты даже без подгрузки будут доступны в вашем проекте.

2. При способе работы с каким-либо сайтом наподобие google fonts, мы выбираем шрифт для проекта, получаем на него ссылку или сразу импортируем в css, и настраиваем уже шрифт в программе.

\* {

    font-family: 'Roboto', sans-serif;

}

2.1 sans-serif обозначает, что шрифт без засечек.

3. Второй способ — это скачать шрифт в папку с проектом, и настраивать его внутри программы. В файле css стилей так же прописываем название семейства шрифта и sans-serif.

4. Так же любому шрифту кроме font-family, можно задать свойство жирности font-weight. Насыщенность шрифта может быть от 100 до 900.

5. font-size размер шрифта можно указывать в пикселях, а также в других единицах измерения

6. font-style изменяет наклон шрифта.

7. letter-spacing изменяет размер между символами в слове.

8. line-height изменяет расстояние между строками.

Важно не забывать указывать правильное название шрифта в font-family, иначе шрифт просто не распознается, обычно на google font всё указывается при выборе шрифта.

.div3{

    font-family: 'Xanh Mono', monospace;

    font-weight: 400;

    font-size: 45px;

    letter-spacing: 10px;

    line-height: 0%;

}

Глава №16

Свойства width&height. Единицы измерения и способы задания цвета.

1. Для каждого элемента можно настроить его высоту и ширину, делается это с помощью параметров в css width, height

2. Задавать размеры можно через px, но для адаптивности сайтов лучше использовать проценты, процент зависит от родительского элемента.

.div2{

    width: 50%;

}

3. Размеры можно так же задать с помощью em, который зависит от размера шрифта, чем больше число перед em, тем линейно больше ширина или высота. Если в блоке не найден размер шрифта, то переходим на уровень выше родительского блока.

.div2{

    font-size: 25px;

    width: 50%;

    height: 10em;

}

В данном случае высота равна 250px для данного блока

4. Для картинок img, так же можно задать размеры, они сожмутся или станут больше в зависимости от установленных значений, так же можно использовать процент от родительской формы.

5. Цвета можно задать как hex или rgb или если надо сделать цвет прозрачным можно использовать rgba.

5.1 Для hex используем комбинацию из 6 знаков, где числа от 0 до 9, а буквы от a до f

5.2 Для rgb используем три числа или если у нас rgba, то четыре, где четвёртое число — это параметр прозрачности от 0 до 1.

.div1{

    color: #00ab32;

    width: 50%;

}

.div2{

    color: rgb(231, 145, 89);

    font-size: 25px;

    width: 50%;

    height: 10em;

}

.section2 {

    color: rgba(231, 178, 190, 0.7);

    width: 80%;

}

Глава №17

Блочная модель CSS. Margin & Padding

1. Margin 🡪 устанавливает величину отступа от каждого края элемента.

2. Padding 🡪 устанавливает значение полей вокруг содержимого элемента.

3. Border 🡪 устанавливает толщину, стиль и цвет границы вокруг элемента.

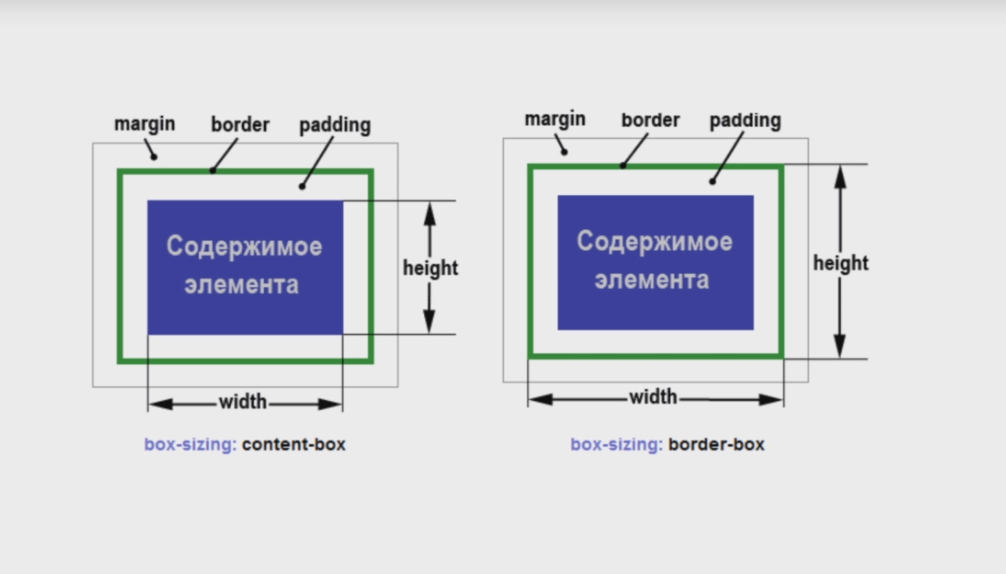
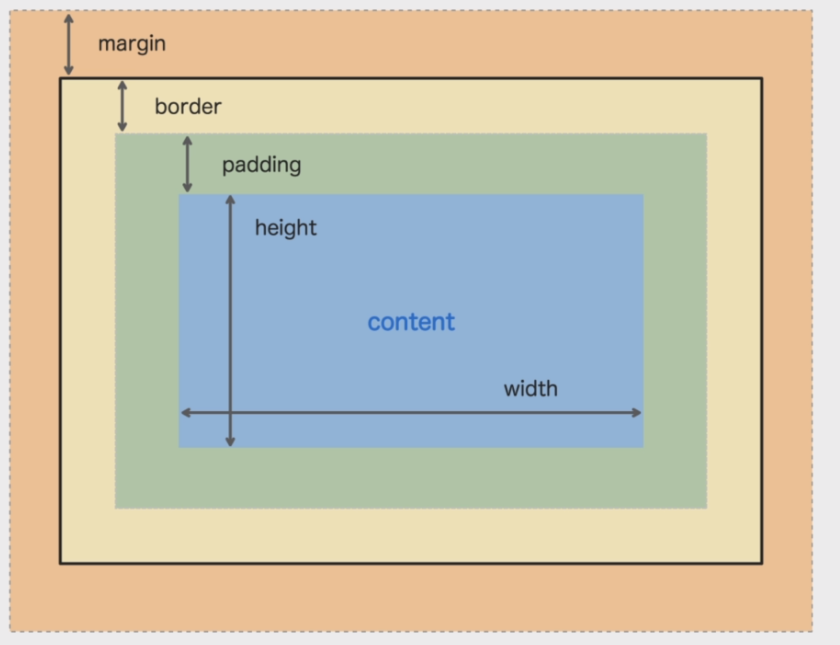
4. У блочных элементов есть отступы по умолчанию, их можно убрать командой margin.

5. Alt + z 🡪 перенос текста на следующую строку.

**Margin and Padding**

1. box-sizing: border-box; Свойство задаёт всем блокам неизменность размеров, то есть текст из блоков будет вылезать при переполнении, но блок сам больше не станет.

1.1 По умолчанию стоит box-sizing: content-box; Свойство противоположное border-box; Блок расширяется под текст и текст и части блока могут залезать на другие блоки.



2. margin и padding можно задавать, как padding: 20px 10px 0px 5px

(обход идёт по часовой стрелке)

\*{

    margin: 0px;

    box-sizing: border-box;

}

.first{

    /\* width: 200px; \*/

    /\* height: 100px; \*/

}

.second{

    /\* margin: 40px 0px 0px 0px; \*/

}

.test1{

    width: 200px;

    margin-left: 45%;

}

.test2{

    margin-top: -144px;

    max-width: 200px;

}

3. При сдвиге одного margin все остальные margin в том же направлении сдвинутся.

4. width и height могут принимать отрицательные значения и значения больше 100%

Глава №18

Продвинутая работа в консоли

1. В консоли разработчика можно просмотреть значения высоты и ширины для блока

2. В консоли разработчика можно изменять параметры на странице, без изменения кода, но после перезагрузки страницы всё вернётся к первоначальной разметке.

3. Можно менять размеры экрана и тестировать, как при разных скоростях Интернета поведёт себя сайт.

Глава №19

Свойство display и выравнивание контента

1. Не у каждого блочного элемента, есть отступы margin сверху и снизу, смысл блочного элемента, что он занимает строку целиком.

2. С помощью свойства display строчные элементы можно превратить в блочные и наоборот.

.test1{

    display: block;

    width: 50%;

    margin-top: 50px;

    margin-bottom: 50px;

}

3. Чтобы заменить блочный элемент на строчный используем:

.test1{

    display: inline;

}

4. Надо помнить, что строчный элемент не имеет параметров высоты и ширины, он автоматически подстраивается под информацию внутри блока.

5. Чтобы два блока могли стать в один ряд надо задать свойство в display:

Inline-block, тогда элемент будет иметь свойства строчных и блочных элементов, и не будет занимать всю строку. Надо помнить, чтобы блок мог сдвинуться должно быть свободное место, поэтому если сдвигаем блок, то у обоих блочных элементов прописываем свойство inline-block.

.second1{

    background-color: purple;

    width: 60%;

    height: 50%;

    display: inline-block;

}

.second2{

    background-color: sandybrown;

    width: 30%;

    height: 50%;

    display: inline-block;

}

6. display: none убирает элемент полностью из вёрстки, все дочерние элементы тоже пропадут.

.second1{

    background-color: purple;

    width: 60%;

    height: 50%;

    display: none;

}

7. margin для выравнивания можно использовать, с несколькими разными единицами исчисления внутри:

margin: 0px auto 0px 25%;

8. Для выравнивания текста используем команду text-align: center. У данной команды много параметров выравнивания по правому краю и другие, так же надо помнить, что изначально выравнивание по левому краю.

Глава №20

Свойства border & box-shadow

1. У тега input есть значение max и min, для типов, которые позволяют работать с числами, устанавливают максимальное и минимальное значение соответственно.

<form action="#">

                        <label for="cart\_amount">number</label>

                        <input type="number" class="cart\_amount" min="1" max="100" value="1">

                        <label for="cart\_btn">buy</label>

                        <input type="submit" class="cart\_btn" value="купить">

                    </form>

2. height указываем с помощью единицы измерения vh

.wrap{

    height: 70vh;

}

3. Чтобы центрировать товар, можно использовать margin с параметрами:

.cart{

    width: 260px;

    height: 440px;

    padding: 10px;

    margin: 100px auto;

}

Border:

1. border позволяет задать границы элементу, их можно задавать по отдельности свойствами, но можно и одной строкой.

border: 2px grey solid;

1.1) 2px 🡪 это толщина границы

1.2) grey 🡪 это цвет границы

1.3) solid 🡪 Это вид границы (цельные линии, пунктиры, и др)

2. Для задания границы с нескольких сторон используем синтаксис, как

border-top: 2px grey solid;

border-left: 2px grey dashed;

3. Для стилизации границы, чтобы можно сделать её закруглённой, используем синтаксис:

border-radius: 4%;

border-radius: 15px;

Края границы станут закруглёнными, можно сделать с помощью процентов, и граница просто станет круглой.

4. Для создания тени используем box-shadow:

 box-shadow: 40px 50px 30px 15px rgb(211, 211, 211);

4.1 первый параметр 40px показывает (ось x)

4.2 второй параметр 50px показывает (ось y)

4.3 третий параметр 30px показывает размытие тени

4.4 четвёртый параметр показывает отдалённость тени во все стороны на кол-во px

4.5 задаёт цвет тени

5. Чтобы изменить прозрачность цвета, используем свойство opacity. Данное свойство изменяется в пределах от 0 до 1.

.cart\_cost{

    font-size: 18px;

    margin-top: 10px;

    opacity: 0.8;

}

6. outline убирает обводки на сайте у изменённых границ кнопок

outline: none;

7. Чтобы изменить курсор:

cursor: pointer;

Глава №21

Технология Flexbox

1. Для группы элементов, можно задать свойство группового перемещения, с помощью display: flex. У данного метода есть две оси (главная и побочная)

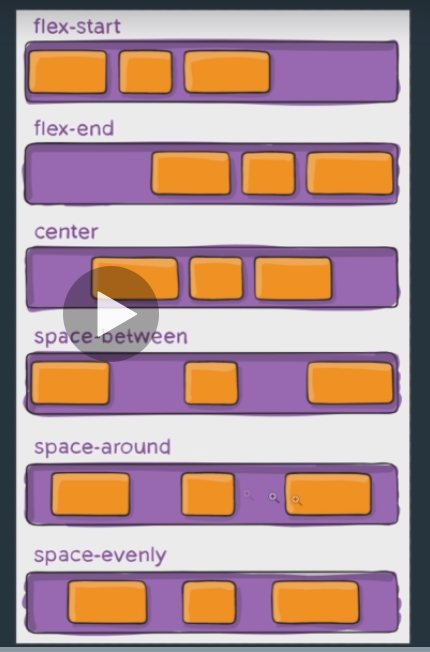
.container{

    min-height: 600px;

    display: flex;

    justify-content: center;

}



2. Метод jusify-content: center отвечает за изменение расположения элементов по главной оси. У данного метода есть много различных аргументов, которые представлены выше.

3. Метод align-items: flex-end; отвечает за расположение по второстепенной оси.

.container{

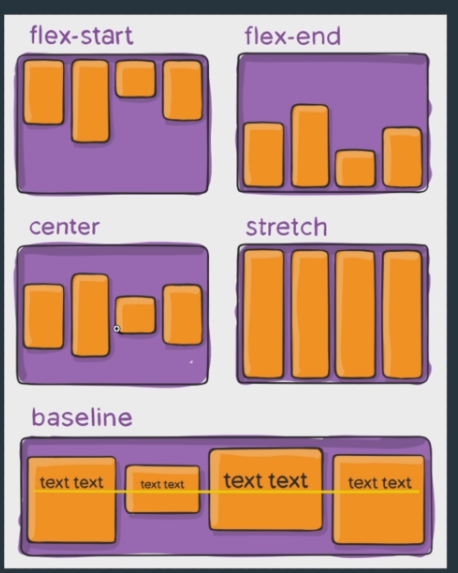
    min-height: 600px;

    display: flex;

    justify-content: space-between;

    align-items: center;

}



4. Метод flex-direction: column или (row) меняет направление главной оси.

5. Чтобы при появлении новых элементов в flex-box, надо установить перенос элементов flex-wrap: wrap;

.container{

    min-height: 600px;

    display: flex;

    justify-content: space-between;

    align-items: center;

    flex-wrap: wrap;

}

Статья по flex-box: https://habr.com/ru/post/313938/

Глава №22

Позиционирование элементов

1. Чтобы изменить расположение элемента на уже готовом макете, можно использовать, абсолютное расположение (и задать координаты относительно родительской системы отсчёта). У элемента дочернего прописываем position: absolute, а у родительского: position: relative

    position: absolute;

    bottom: 0px;

    right: calc(50% - 50px);

2. Чтобы вывести объект на пересечение границ, нам надо указать отрицательное значение для требуемой стороны:

position: absolute;

    bottom: -50px;

    right: calc(50% - 50px);

Фиксированное расположение

1. Мы можем зафиксировать расположение элемента на странице, чтобы при перемещении страницы элемент перемещался вместе с ней

.mains2 a{

    position: fixed;

    bottom: 10px;

    right: 20px;

}

2. При позиционировании надо всегда задавать width и height у элемента, так как иначе позиционирование fixed работать не будет.

3. Элемент со значением position: fixed, может быть родительским элементом для абсолютного наследования.

4. Можно закреплять целые блоки, чтобы делать меню, которая двигается вместе со страницей, способов применения данной технологии много.

Глава №23

Псевдоклассы. Transition

1. Чтобы при взаимодействии пользователя с элементами интерфейса, нас надо указать особый класс для данного элемента.

1.1 hover 🡪 класс для изменения стиля при наведении на элемент интерфейса.

.head\_block\_1{

    width: 60px;

    height: 60px;

    background-color: red;

    margin-left: 40%;

    margin-top: 10px;

    border: 4px blue solid;

    border-radius: 10%;

}

.head\_block\_1:hover{

    background-color: darkblue;

}

1.2 active 🡪 класс для изменения стиля при нажатии на элемент

Интерфейса.

.temp :active{

    background-color: hotpink;

}

.test3:active{

    width: 500px;

}

1.3 focus 🡪 класс для изменения элемента интерфейса при фокусировке, т.е когда мы можем вводить в него, какие либо данные.

.test3:focus{

    width: 500px;

}

2. Можно комбинировать несколько классов состояний для одного элемента нтерфейса.

.test3:active{

    background-color: rosybrown;

}

.test3:focus{

    width: 500px;

}

3. Для задания временного промежутка при использовании псевдоклассов можно использовать transition

transition: all 0.3s linear;

3.1 Первый параметр отвечает на какие элементы будет наложена скорость выполнения

3.2 Второй параметр отвечает за скорость выполнения

3.3 Третий параметр отвечает за переход, может быть linear (равномерно) или easy (быстрый начальный переход)

Глава №24

Трансформация элементов

1. Для изменения линейных размеров элемента, не меняя его реальные размеры надо использовать transform

1.1 transform: scale(1.2) 🡪 растягиваем по X и по Y на 20%

1.2 transform: scaleX(1.2) scaleY(1) 🡪 можем растянуть по отдельной оси

.step2{

    margin-left: 50%;

    width: 100px;

    height: 100px;

    background-color: red;

    transform: scaleX(1.5) scaleY(1.3);

}

2. Для изменения наклона объекта используем rotate. В качестве угла можно указывать отрицательные значения.

2.1 transform: rotate(45deg) 🡪 поворачиваем фигуру на угол заданный в скобках

3. Можно комбинировать различные настройки в transform

3.1 transform: scale(1.3) transform(-45deg)

transform: scale(1.2) rotate(-32deg )

4. Для изменения положения объекта можно использовать translate, мы передвинем по нашу фигуру относительно начальной позиции по координатным осям.

transform: scale(1.2) translate(90px)

Данным способом можно двигать элементы интерфейса по странице. Можно двигать по каждой оси по-отдельности

5. Для изменения (сжатия) элемента можно использовать skewX(7deg), scewY(5deg), skew(5deg).

transform: scale(1.2) translate(30px) skew(20deg)

С помощью transform и псевдоклассов, мы можем сделать анимацию, при медленном воспроизведении с помощью параметра

transition: all 0.3s linear;

Надо только указать большее время, для более долгой анимации.

Глава №25

Анимация в CSS

1. Для создания анимации используем @keyframes <имя фрейма>

@keyframes rotate{

    from{

        transform: rotate(0deg)

    }

    25%{

        transform: rotate(90deg);

    }

    50%{

        transform: rotate(180deg);

    }

    75%{

        transform: rotate(270deg);

    }

    to{

        transform: rotate(360deg);

    }

}

2. Так же надо настроить анимацию для объекта, так как изначально мы делаем шаблон, который встраивается в объекты интерфейса

animation: rotate 2s infinite linear;

2.1 Первый параметр имя анимации

2.2 Второй параметр время её выполнения

2.3 Третий параметр сколько раз она будет выполняться (Infinite, значит бесконечное число раз)

2.4 Четвёртый параметр означает, как быстро пройдёт анимация

3. Чтобы поднять один элемент над другим используем z-index. Изначально он равен нулю, но при использовании z-index: 1, он поднимется выше других элементов на форме.

4. Чтобы при выходе за блок элементы интерфейса исчезали, надо прописать overflow: hidden.

Глава №26

Вендорные префиксы. Кроссбраузерность

1. Чтобы новые функции и анимации поддерживались старыми браузерами, надо использовать пометки, как webkit:

@keyframes rotate{

    from{

        transform: rotate(0deg)

    }

    25%{

        transform: rotate(90deg);

    }

    50%{

        transform: rotate(180deg);

    }

    75%{

        transform: rotate(270deg);

    }

    to{

        transform: rotate(360deg);

    }

}

@-webkit-keyframes rotate{

        from{

            transform: rotate(0deg)

        }

        25%{

            transform: rotate(90deg);

        }

        50%{

            transform: rotate(180deg);

        }

        75%{

            transform: rotate(270deg);

        }

        to{

            transform: rotate(360deg);

        }

}

2. Для браузеров, чтобы они корректно отображали ваш интерфейс надо задавать не только для анимации webkit, но и для методов, где нам для метода надо сделать пометку.

2.1 Для проверки можно использовать сайт “can I use”, где указаны все свойства возможные для браузера.

3. Чтобы автоматически расставить префиксы, можно использовать сайт

“автопрефиксер css онлайн”, на этом сайте так же можно поставить фильтр префиксов для некоторых браузеров.