Введение

Не обещаю, что при помощи этой штукенции вы сможете изучить HTML, CSS, JS. Ну и конечно же я попробую научить создавать лёгенький сайтик.

Для лучшей ориентации по этому прекрасному документу советую включить навигацию, которую можно найти в поиске.

Я бы мог тут написать все как в интернете ну по-умному, но чёт так впадлу что аж лень так что приятного изучения!

Начну прям тут, потому что нужно объяснить сразу, вся писанина будет проводится при помощи VS code в котором нет необходимости писать все элементы через открывающий треугольник (<) можно и просто начинать писать тег иногда достаточно одной буквы чтобы код понял, что ты хочешь останется лишь выбрать из выпавшего списка необходимый тег и нажать tab и иногда работает через enter.

А также многие элементы имеют типы, подвиды, суть вот для таких мы после тега используем «:» ну пора и изучать.

Главный файл html обычно имеет название index.html.

Файл CSS будет иметь название style.css.

Каждый файл JS будет иметь то название для чего нам нужен тот или иной файл.

1. HTML

1.1. Структура написания

Чтобы создать базовую структуру в VS code мы в пустом файле пишем «!» и нажимаем «tab» и всё она готова далее весь код мы пишем в теге body. Под себя можно изменить внутрянку тега <title>Название страницы</title>. Также в теге head мы будем подключать файлы CSS изменять иконку сайта ну и многое ещё.

Структура html4 разметки:

```
<body>
    <div id=”header”>…</div>
    <div id=”main”>…</div>
    <div id=”footer”>…</div>
</body>
```

Структура html5 разметки

```
<body>
    <header>…</header >
    <main>…</main>
    <footer>…</footer>
</body>
```

С структурой разобрались теперь пора бы и перейти к изучению всех необходимых тегов.

1.2. Про теги

Создаем файл пишем любое название и пишем .html! Файл для написания html создан – начало положено.

В html существует множество тегов (элементов), например div, ul, li, h1-6 и т.д.

Каждому тегу можно добавить class (обозначение для быстрого написания, а также для обращения через CSS – «.»), а также id («#»).

Class может использоваться для любого количества элементов и добавляется при помощи … (быстрое написание: div.”…”).

ID используется **только для одного** элемента и добавляется при помощи … (div#.”…”).

Тег условно разделяется на название тега, название класса, название id, «внутритеговость» и «внутрянку» вот так:
<’название тега’ class=”’название класса’”
id=”’название id” ‘внутритеговость’>’внутрянка’</’название тега’>

Существует 2 типа написания это «старпёрский» и «мужской»:

«Старпёрский» написание через <> в VS code не имеет смысла т.к. он умный в отличии от нас, и он сам понимает чё мы от него хотим, поэтому существует мужской способ написания через название тега, например тег div с классом div и id div:
Пишем div.div#div и нажимаем
вот так всё просто а не как в «старпёрском» .

Про общее для каждого тега я написал значит пришло время для объяснения какие вообще виды тегов используются и зачем они вообще упали так что начнем, пожалуй, с блочных тегов или как их ещё можно назвать для тегов-оберток.

1.2.1. Основные теги

Под названием «основные теги» подразумеваются те теги, которые чаще всего будут использоваться при написании сайта.

Тег div – буквально тег, который чаще всего используется, причем абсолютно для всего что надо: необходимо создать обертку пишем тег div нужен блок в обертке да вообще без проблем создаем тег div в другом теге div и никаких проблем. До выхода HTML5 использовался для отделения блоков шапки(header), основного контента(main), подвала(footer) и т.д. Но с выходом HTML5 появились новые блочные теги, которые полностью заменили тег div с необходимым id.

Теги header, main, section, nav, footer – каждый тег буквально является сыном тега div, но позволяет проще описывать для чего используется тот или иной тег без необходимости добавления класса или id.

Тег ul – тег списка все списки в идеале создавать через него, потому что это немного, но всё-таки выделяет список из общего кода, потому что понимать, где что написано это значит быстро ориентироваться по коду, а это в свою очередь значит, что мы сможем быстрее исправлять недочёты, которые мы можем видеть при просмотре нашего сайта.

Тег li – используется в списке для обозначения элементов списка **важно!** Сначала мы пишем тег li только потом в нем все остальное.

Про основные теги поговорили пора бы и перейти к другим, тегам ввода данных.

1.2.2. Теги ввода данных

Тег ввода пишется <input type=”…”/>

Тегов ввода существует очень доху… много короче, но можно использовать от силы 5 как основные меняются они при помощи изменения показателя type, существуют многие типы тупо для того, чтобы было проще понимать зачем нам нужен тот или иной инпут. В основе лежит тип text при помощи него можно передавать всё и циферки и букавки и номера телефонов и почты ну короче всё что можно написать на клавиатуре. Потом есть тип date ну тут всё понятно, чтобы записывать дату. Тип file позволяет в него запихивать файлы также если после типа написать multiple, то можно будет прикреплять не 1 файл получится такая структура: <input type=”file” multiple/>. Также существует тип checkbox создает коробочку, которая работает по такой системе что она просто является переключателем положений ну типо при нажатии меняет Boolean составляющую с false на true и обратно. Ну последний который мы рассмотрим будет тип radio создает круглую кнопочку, которая изначально имеет при себе такой показатель name, который работает по принципу взаимного исключения (первое умное слово бтв) если просто объяснить, то: если нажата одна, то все остальные не нажаты.

Также для многих существует показатель placeholder, который позволяет добавить текст, который будет показываться, когда инпут пустой работает только для тегов, которые имеют поле ввода.

Общее количество и все типы можно посмотреть, написав на новой строчке в файле .html: «input:».

1.2.3. Текстовые теги

Тут вообще всё супер быстро мы будем использовать только теги заголовков и один тег для абзацев.

Заголовки h1-h6 каждый меньше прошлого ну и как бы всё.

Тег абзаца … нужен для абзацного написания какого-либо текста, используется раз в 1000 лет.

1.2.4. Остальные теги

Тут вообще всё плохо тегов в html очень много, но их использование не всегда нужно, но всё-таки есть такие, которые я не отнес ни к одному из прошлых заголовков так что по быстренькому разберёмся с теми, кто нам иногда будет помогать.

Тег гиперссылок … - нужен для перехода на другую страницу либо для перехода к определённому элементу на данной странице достаточно лишь написать в href необходимый путь и всё мы в шоколаде.

Тег фото и видео , <video src=”…”/> ну что про них говорить тег img используется для показа фоток на сайте а тег видео нуже для показа видео как не удивительно но на теге видео надо немного остановиться всё таки есть что про него рассказать у него есть немного нужных фишек например автозапуск(autoplay) видео или выключение(muted) звука видео ну и как же без автоповтора(loop). Для них также можно добавлять классы и id.

Тег кнопок button – тег кнопок чё тут говорить кнопка как кнопка может также самое настраиваться через классы и id есть несколько видов кнопок, но они бесполезны, кроме шуток реально можно использовать без типов они все будут одинаковые.

Даже что-то и говорить больше нечего пора бы и перейти к заданию стилей так что перейдём к CSS.

2. CSS

2.1. Про CSS

Каскадная таблица стилей (второе умное словосочетание) используется для того, чтобы наш сайт не выглядел как кусок параши с просто накиданными картинками и текстом.

Из курса Власенко мы поняли, что существует три вида CSS стилей это внутритеговый, внутридокументный и внешний (из другого документа). Если коротко внутритеговый(style пишется во внутритеговости) это мы прогибаемся под каждый тег и каждому пишем как он будет выглядеть, внутридокументный пишем внутри тега head тег style. Ну а внешний это просто мы создаем файл с названием оканчивающимся .css. Потом мы его должны подключить в теге head пишем link:css и нажимаем tab после чего меняем показатель href на необходимый нам путь до файла, который мы создали. Использовать рекомендую именно внешний, потому что почти все стили мы будем использовать не только для одной страницы, а каждый раз копировать и вставлять тег style не прикольно + это расширяет возможное количество ошибок, а чем проще ошибиться, тем с большей вероятностью ошибка будет допущена, исправление некоторых ошибок может занять довольно продолжительное время из-за объемности кода.

2.2. Структура написания

Структурно написание выглядит что мы обращаемся к чему-либо после чего открываем {}, и пишем необходимые строчки для задания необходимых стилей. Пример для обращения к тегу header:

```
header{
    …
}
```

Пример для обращения к классу blocks:

```
.blocks{
    …
}
```

Пример для обращения к id left-side:

```
#left-side{
    …
}
```

2.3. Стилизация текста

Тексту можно изменить цвет, цвет заднего фона, задать картинку как задний фон, изменить размер, ширину, «семью» шрифта, расположение текста.

Имя color используется для изменения цвета самого текста.

Имя background используется для изменения заднего фона, например можно задать какую-нибудь фотографию для заднего фона.

Имя background-color используется для задачи заднего фона заливкой цвета.

Имя text-align используется для задачи расположения текста относительно блока родителя (третье что-то умное) если коротко, то все теги имеют уровневую иерархию т.е. каждый тег является родителем следующего, который находится в нем. У text-align имеется несколько так скажем положений изначально text-align: start текст пишется максимально слева, text-align: end текст прижимается к правой стороне, text-align: center весь текст находится в центре объекта в котором он пишется, text-align:justify текст растягивается по всей длине объекта, образуя большие отступы.

Имя font-size изменяет размер шрифта.

Имя font-weight изменяет ширину шрифта.

Имя font-family изменяет шрифт.

Также можно добавить свой шрифт если его нет в font-family для этого используется функция @font-face для того чтобы подключить свой шрифт нам надо написать в @font-face два имени это font-family чтобы задать название шрифта, а также написать src: url() чтобы задать ссылку на шрифт после чего мы можем использовать этот шрифт для любого текста.

2.4. Изменение цвета

Можно поменять не только цвет самого текста и цвет заднего фона, но также можно поменять и цвет обводки. В главе 2.3 я уже говорил, как изменить цвет текста (color) и цвет заднего фона (background-color).

Но про обводку я еще не говорил изменение обводки производится через задачу стилей для border в него мы прокидываем три показания: 1) Размер в px; 2) цвет обводки; 3) степень заливки; Степень заливки solid, dotted, inset, dashed solid, dashed double none, dashed groove none dotted. Можно использовать только solid (сплошная заливка цветом).

2.5. Display: ...

Дисплей мы используем для изменения положения блоков. Изначально равен block, но также есть и другие имена row, grid, none, inline-block.

Block все блоки идут в столбик друг за другом.

Row все блоки идут в строчку друг за другом без отступа.

None убирает блок с экрана, блок не занимает места.

Inline-block блоки будут идти вместе с текстом в строке.

Grid блоки будут распределены по гридам строкам и колонкам.

2.5.1. Display: flex

У флекса очень много дополнительных показателей, но в основном используются всего два показателя, которые позволяют изменять положение всех компонентов в блоке, для которого задан флекс по двум осям X и Y и один показатель для изменения положения блоков в строчку или в столбик.

Flex-direction позволяет менять положение для блоков изначально имеет значение row (строчка), но также имеет значение column и обратные от этих двух reverse-row, reverse-column.

Align-items (Y) имеет несколько значений flex-start, flex-end, center, scretch, baseline. Start и end работают также как и с текстом т.е. прикрепляют к одной из сторон старт к верху энд к низу, center ставит все объекты максимально по центру блока, scretch растягивает блоки на максимальную высоту блока, baseline работает так же как и center только ставит по центру максимально высокого блока.

Justify-content(X) имеет немного больше значений flex-start, flex-end, center также как и align-items и работает так же как и в align-items только меняет положение по другой оси по ширине, а также и имеет новые показатели space-between, space-eleventy и space-around которые в свою очередь меняют количество свободного места между всеми блоками.

2.5.2. Display: grid

У grid мы будем использовать только одну настройку grid-template-columns: repeat(…, 1fr). В зависимости от изменения «…» будет и меняться количество объектов в каждом столбце, т.е. если мы напишем repeat(3, 1fr) то у нас любое количество объектов будет распределено по 3 в каждый столбец.

2.6. Анимации

Все анимации работают только при событиях (пункт 2.9) и\или при взаимодействии с JS, но пока что немного рано об этом говорить.

Анимации в CSS можно условно разделить на два вида:

1) Встроенные в CSS.
2) Придуманные и написанные человеком.

Встроенные пишутся через transition.

Придуманные человеком создаются при помощи @keyframe{} создаются путём написания постепенной анимации через «%» например

```
@keyframe «Название анимации»{
    0%{
        margin-left: -100px;
    }100%{
        margin-left: 3%;
    }
```

После того как мы создали keyframe нам необходимо задать её как анимацию для какого-нибудь объекта через название animation. Для работы анимации нам необходимо написать в animation название нашего фрейма, а также задать количество секунд, сколько будет выполняться анимация, после чего можно задать через сколько секунд она повториться (изначально повторение выключено), ну можно и просто написать animation и посмотреть, что можно туда написать так как рассказать можно много.

2.7. @media

@media screen{} Используется для того чтобы мы могли настраивать на любом разрешении для этого нам надо написать после screen and (min-width: …px) and (max-width: …px). Ну и всё больше нечего говорить, просто здесь пишем необходимый тег и задаём ему стили ну и как бы всё.

2.8. Остальные используемые редко

Так-то настроить можно многое, но многое из этого не имеет смысла.

Border-radius закругление обводки.

Есть такая структура есть блок, например текст потом идёт padding после этого идёт border ну и в конце идёт margin если проще всего объяснить сначала блок потом внутренний отступ потом обводка ну и в конце внешний отступ.

Opacity настройка прозрачности настраивается по принципу 1 = 100%, 0.1 = 10%

Width, height ширина высота объектов.

Position позволяет, например закрепить какой-либо блок чтобы он показывался всегда например можно закрепить шапку сайта и не важно на сколько прокрутить страницу в низ шапка будет сверху. Из основных значений имеет relative (изначально), fixed, absolute. Про fixed я уже рассказал, теперь про абсолют позволяет сделать любой объект самым передним слоем, и из этого следует следующий возможный стиль z-index буквально позволяет изменить положения объекта по оси Z. Работает по принципу что z-index: 1; будет самым нижним слоем и если нам надо наложить например фотографию поверх этого блока то мы на фотку накладываем z-index выше чем у того блока поверх которого должна быть фотка.

Вроде про всё рассказал, если нет по мере написания расскажу.

2.9. События

Перейдём, пожалуй, к последнему что есть в CSS про что ещё не говорили, а именно условия их много, но расскажу подробно лишь про некоторые.

Все события добавляются к тегам без пробелов a:hover

1) :hover позволяет менять стили для тега при наведении.
2) :active позволяет менять стили при нажатии.
3) :focus работает так же как при нажатии, только оставляет стили после отпускания.
4) :nth-child() позволяет настаивать определённый тег-ребёнок, это значит что мы можем настраивать например определённый элемент списка вместо добавления для него нового id или класса.

В принципе все остальные бесполезны всё остальное можно сделать через православный JS, к которому мы и перейдём.

3. JavaScript

3.1. Структура написания

Структуры как таковой нет код пишется также как и в любом языке программирования, но нас не сильно будет интересовать написание классов, массивов про них я и говорить не буду, впрочем, как и про многое другое из этого следует что изучать мы будем лишь то, что мы будем использовать. Так что начнем с переменных.

3.2. Переменные

Есть всего три переменные: var, const, let.

Const в основном используется для записи данных.

Let используется в цикле for например для того чтобы пробежаться по списку для подсчёта количества строчек.

Var такой же как let, но в данный момент более не используется поэтому на него всё равно. Теперь перейдём к тому, как нам подключить js к html, чтобы использовать элементы текста для добавления новых возможностей нашему сайту.

3.3. Подключение к HTML

Чтобы подключить js html нам надо после закрывающего тега </body> написать строчку <script src="…"></script>, где внутри src указать путь до файла js, можно конечно и писать код просто в теге <script> без src, но тогда мы не сможем использовать наши функции на других страницах. С подключением разобрались пора бы теперь и научиться создавать функции для создания более интересного сайта.

3.4. Создание функций

В скрипте существует два вида функций:

Так скажем простые

Линейные

«Простые» функции создаются путём написания такой конструкции:

```
Function «название функции» («параметры») {
        … код функции
}
```

Для линейных функций используется следующая конструкция:

```
const «название функции» = () => {
  …
}
```

После создания функции есть два варианта подключения для работы функции:

1) В html непосредственно на объект через множество условий, таких как: onClick, onChange про все-все нет смысла рассказывать поэтому в пункте 3.5. будет более подробно лишь про некоторые.
2) .addEventListener(“условие”, «название функции») перед точкой надо написать название той константы на которую вы хотите навесить функцию.

3.5. Условия JS

Как я писал ранее их существует огромное множество, но про все нет смысла рассказывать, потому что некоторые бессмысленно использовать. Поэтому расскажу лишь про часто используемые.

Все условия в html пишутся в теге также как класс и начинаются с «on», но в js они употребляются без «on»

Click – При нажатии функция будет срабатывать.

Change – При изменении в основном используется с инпутами, т.е. при изменении значения инпута будет срабатывать функция.

Mouse… - Любое взаимодействие с курсором например mouseEnter будет вызывать функцию при том условии если курсор зайдет на заданное поле например на кнопку.

Ну и на этом в принципе всё остальные конечно тоже иногда используются, поэтому, когда пригодится тогда и изучите как-то так.

3.6. Условный оператор If

Ну вот что про него говорить буквально в названии всё написано – условие, но если супер просто объяснять, то условный оператор позволяет создать такой код, который будет обрабатываться только когда определённое условие выполнено.

```
If(«условие»){
        …
}
```

3.7. Цикл For

Про цикл тоже нечего говорить, если просто сказать цикл будет выполнять определённый код только пока число, заданное в переменную let будет меньше заданного числа.

```
for (let index = 0; index < array.length; index++) {
        const element = array[index];
}
```

3.8. Функции JS для взаимодействия с HTML

Их очень много, но как обычно мы поговорим только про самые нужные, например про возможность изменения класса для задания определённого класса для последующей настройки стиля используется для создания выдвигающегося меню навигации, к примеру. Для этого мы используем «…».classList.toggle(“название класса”).

А также можно изменять напрямую абсолютно все стили, но это также немного бесполезно.

В js одно равно не равно, а присваивание, два равно это сравнение, а три равно это строгое сравнение.

Есть конечно в js и встроенные функции для обратной связи с пользователем на типу всплывающих окон в браузере, но кому это вообще надо разве что есть смысл объяснить как выводить в консоль

информацию, для этого существует функция console.log() позволяет вывести всё что угодно в консоль.

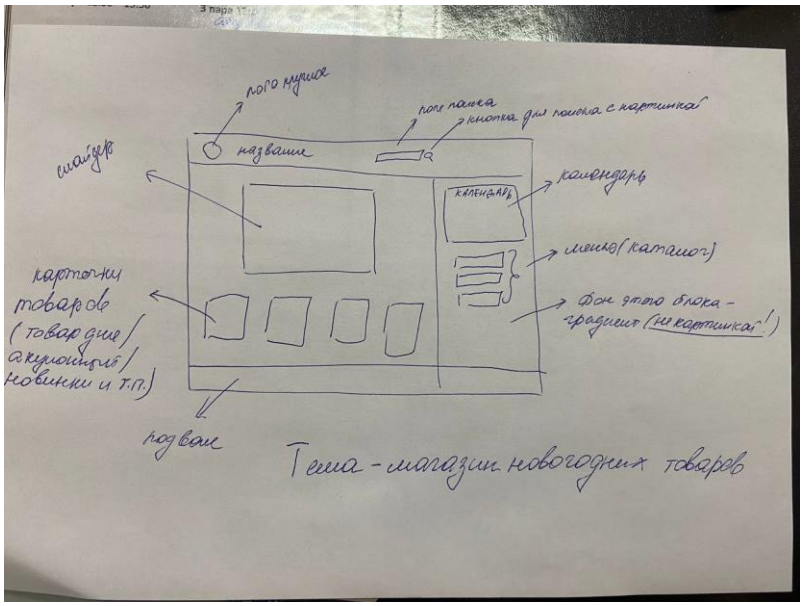
Конструкция switch case break это своего рода автоподборщик работает очень просто в switch вносим условие после через case задаем значение всё что задано в каждом case будет работать до того момента как мы не напишем break, проще разобрать на примере есть у нас загаданное число 1 и его нам надо подобрать счёт начнём с 0 запишем загаданное число в переменную number

```
const number = 1;
switch(number){
    case 0:
    console.log(«Загаданное число больше!);
    break;
    case 1:
    console.log(«Ты угадал!);
    break;
    case 2:
    console.log(«Загаданное число меньше!);
    break;
}
```

Так же можно записывать несколько значений case под в одни и те же break.

Ну вроде как всё с основой покончено остается лишь сделать какой-нибудь сайтец, на котором мы сможем применить наши приобретённые (надеюсь) знания.

4. Написание сайта



Шаблон.

Ссылки на видео:

1 часть: https://www.youtube.com/watch?v=hUAP5QWmScM

2 часть: https://www.youtube.com/watch?v=a9lblk0ptJU

Ссылка на скачивание фото:

https://disk.yandex.ru/d/6edS4dqHZZpVmg