29.09

# 1. Web-дизайн. Введение: устройство веб-сайта.

Общая информация

* 0

Привет друг! Ты начинаешь свой путь в мире разработки, с чем я тебя и поздравляю! :)

Ты можешь пользоваться таким вот мини-конспектом, чтобы освежить свои знания перед выполнением домашнего задания или если ты по какой-то причине пропустил занятие.

На первом занятии мы изучили основные понятия, а именно:

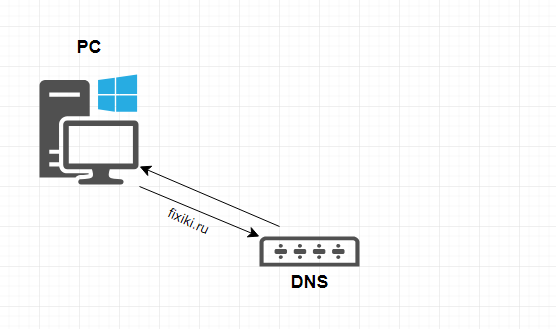
1. Клиент-серверная архитектура и с чем это едят.
2. Базовые понятия из мира веб-разработки (нет, это не скучная теория, а очень даже важная и полезная информация!).
3. Что такое теги и как их выгуливать..упс, то есть использовать и какие же бывают теги их виды.

Давай теперь обо всём по порядку.

Клиент-серверная архитектура

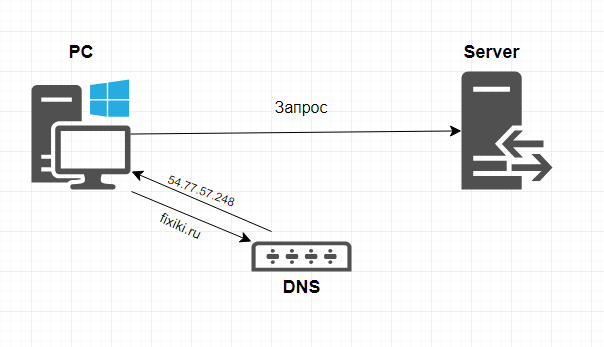
* 0

Начнём с вопроса про клиент-серверную архитектуру. Давай-ка посмотрим на эту картинку.

****

Компьютер с системным блоком - это ты, например, когда зашёл в интернет, чтобы посмотреть новую серию “Фиксиков”. :)

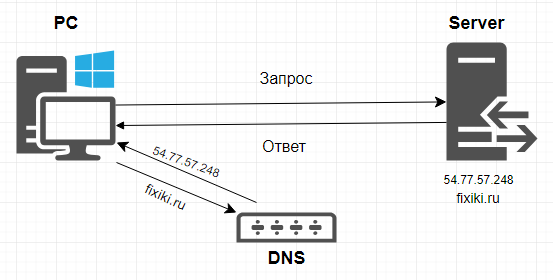
Когда ты открыл браузер, тебе нужно перейти в адресную строку. Теперь ты пишешь адрес: “fixiki.ru” и нажимаешь кнопку Enter, а вот тут то и начинается самое интересное…

****

Твой запрос отправляется на особенный компьютер, который по адресу сайта ищет ip-адрес сервера, на котором этот сайт расположен.

Такой компьютер называется DNS-сервер, а IP-адрес - это адрес, который назначается любому компьютеру в сети Интернет, чтобы устройства могли посылать друг другу сообщения, так же, как тебе надо знать номер телефона, прежде чем написать SMS-ку другу.

После того, как DNS-сервер перевёл твой запрос в ip-адрес вида: 54.77.57.248, он завершает твою работу, и запрос продолжает путь к нужному серверу, который находится по этому адресу.

****

Ты добрался до нужного тебе сервера - поздравляю! Теперь этот сервер генерирует для тебя ответ из множества файлов (об этом мы поговорим позже) веб-страницу, которая в свою очередь отправляется на твой компьютер в виде уже готовой страницы с картинками, текстом и анимацией.

Вот такой путь пришлось пройти фиксикам, чтобы попасть к тебе и отобразиться в окне твоего браузера. :)

HTML документ

* 0

Плавно мы подошли ко второму пункту - это важные понятия из мира веб-разработки.

Пожалуйста, возьми тетрадку и запиши их:

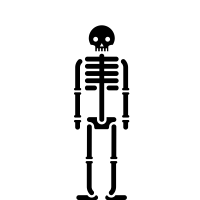
1. **Тег**- это кирпичик, из которых состоит весь HTML-код веб-страницы.
2. **Сайт**- несколько электронных документов (файлов) частного лица или организации в компьютерной сети, объединённых под одним адресом, который называется доменом.

Я ведь говорил, это будет не скучно, а знать такие важные вещи - очень даже полезно.

Мы начали говорить о тегах, это очень важный и нужный элемент, а сейчас рассмотрим, какие же они бывают. (Это ты тоже обязательно запиши в тетрадь, поверь - так они намного лучше запоминаются!):

1. **<!DOCTYPE html>** - говорит браузеру, какой перед ним документ.
2. **<html>** </> - контейнер, который заключает в себе всё содержимое веб-страницы.
3. </> означает завершение тега, а также что тег парный.
4. **<head>** </> - раздел, содержащий теги с информацией для браузера и поисковиков
5. **<meta>** - кодировка, информация для поисковика
6. **<title>** </> - заголовок документа
7. **<body>**</> - содержимое страницы, отображаемое в браузере.

Все теги, которые ты записал, то есть первые 7 штук, представляют собой “скелет” html-документа, или по простому - пустую, но уже готовую к творчеству html-страницу.

****

Но давай продолжим. Теперь запишем теги, которые у нас располагаются внутри <body> и будут видимы на странице:

1.  **<h1> </>...<h6> </>** - заголовки и подзаголовки от первого до шестого уровня

2.  **<p>** </> - абзац, если хочешь написать какой-то текст, то используй его :)

3.  **<br>** - перенос строки

4.  **<img src="адрес картинки">** - добавление картинки

Практика

* 0

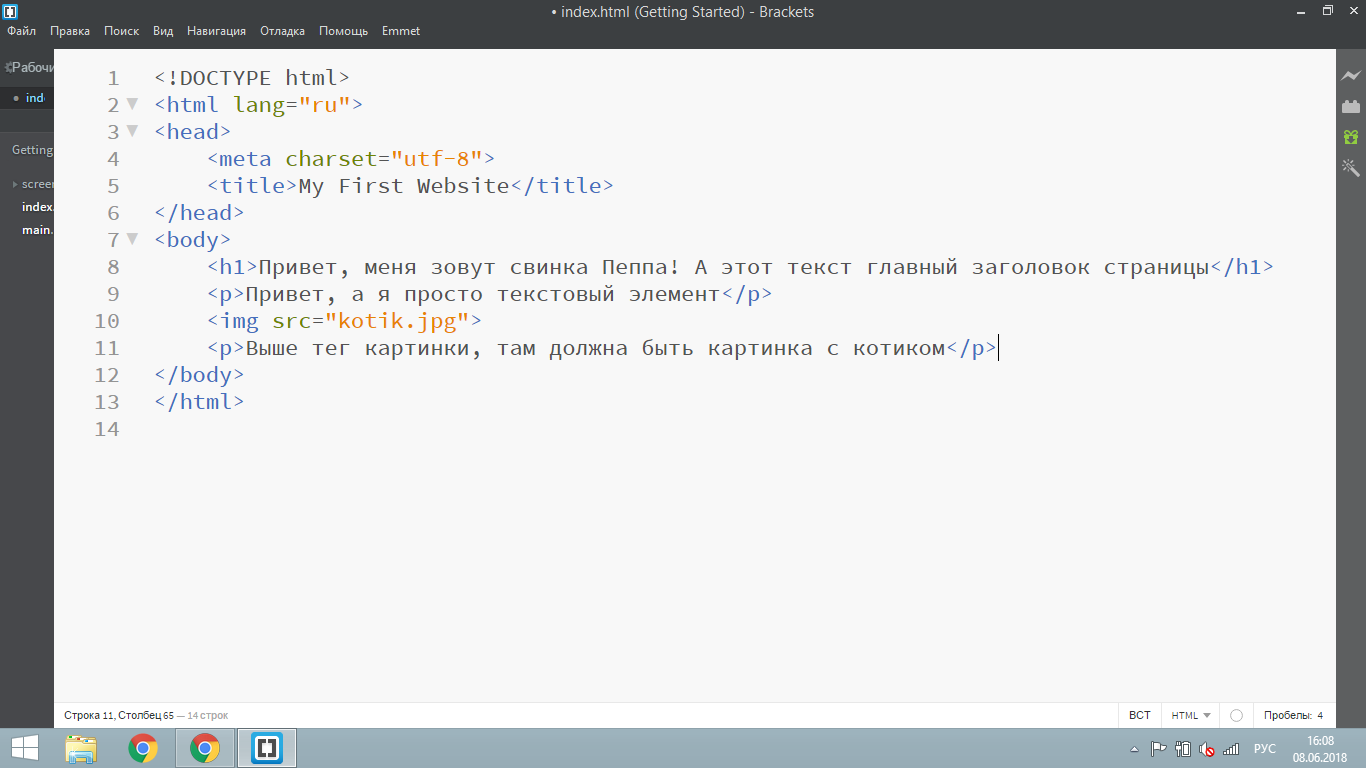
Что-то мы с тобой заболтались… Пора перейти к практике и создать свой первый сайт. Наконец-то!

Для разработки мы будем использовать редактор кода, который называется [Brackets](https://brackets.io/), скачай его и установи - это абсолютно бесплатно.

Теперь тебе необходимо создать новый документ, давай разберемся, как это сделать по шагам:

1. Открой **Brackets.**
2. Нажми сочетание клавиш **Ctrl+N** (оно создаст новый документ).
3. Теперь нажми **Ctrl+S** (это сохранение файла).
4. Открылось окно, видишь? Тебе нужно выбрать, куда сохранить файл и обязательно дать ему название: **index.html** (проверь внимательно, именно такое название, иначе далее ничего работать не будет)
5. Всё. Нет, правда, это всё - твой файлик создан и готов к работе.

Вот мы и подобрались к самому вкусному, тому, а как же создать свою первую страничку, для этого тебе потребуется напечатать вот этот код в твоём только что созданном документе.

****

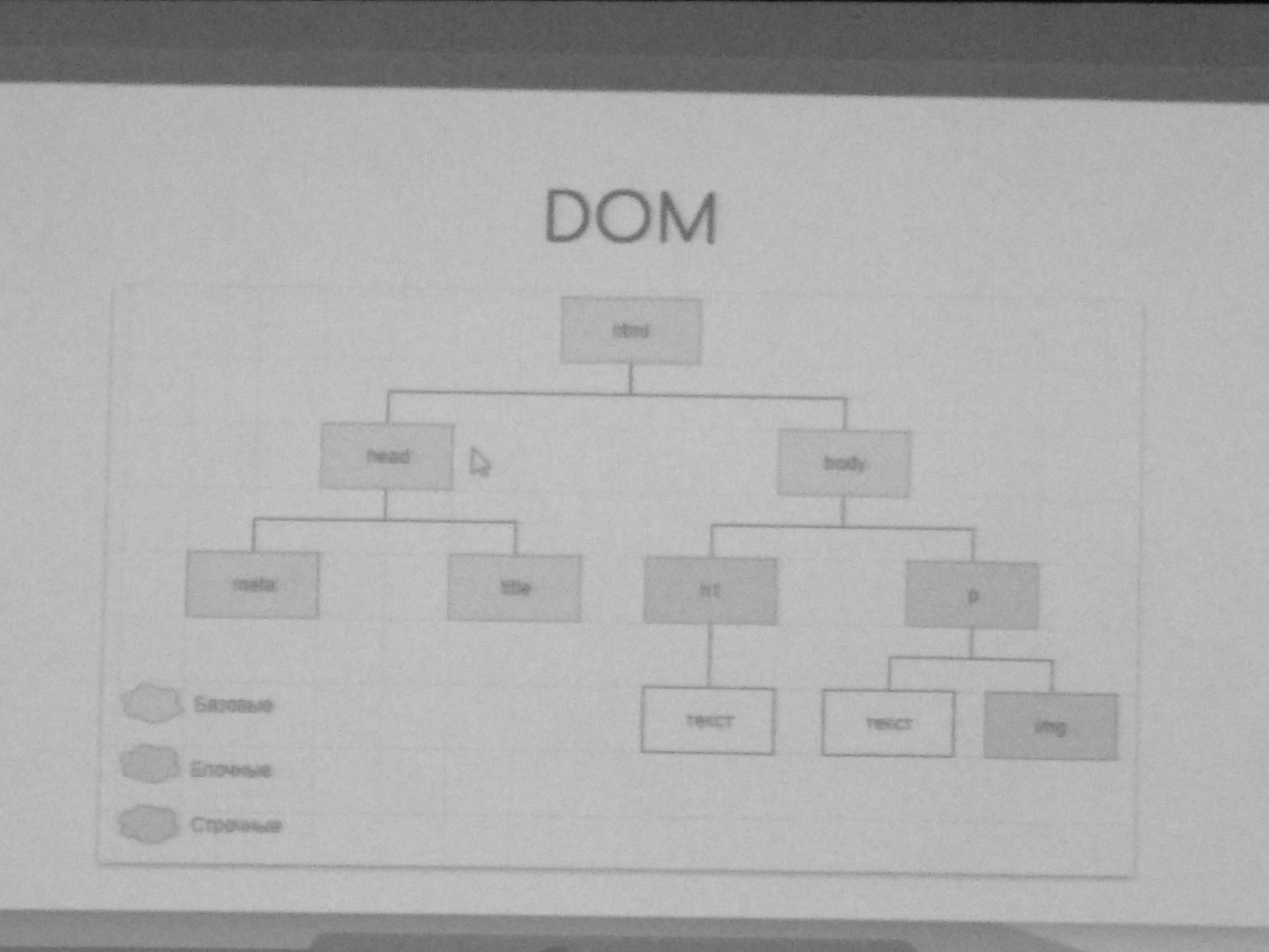
Теперь сохрани свой документ, для этого нажми сочетание клавиш **Ctrl+S**.

Уф… устал? Но ведь нужно проверить, что всё работает. Для этого открой папку, в которую ты сохранил свой файл, и сделай по нему двойной клик, если всё верно - то у тебя откроется браузер, в котором ты и увидишь результат своих трудов!

Поздравляю, ты это сделал и ты большой молодец! До встречи на следующем уроке.

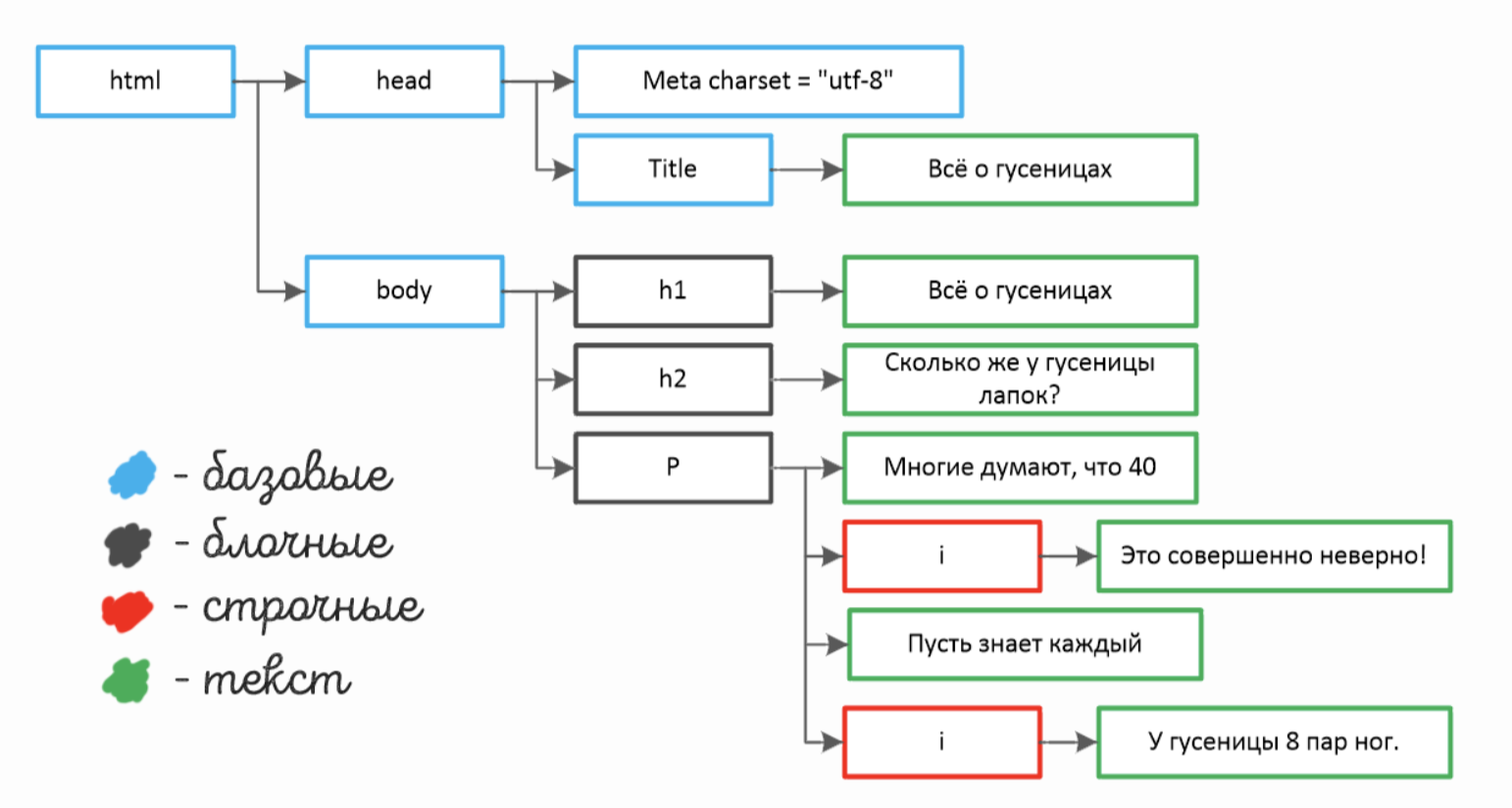
06.10

DOM – Document Object Module



<b>…</b> -жирный текст

<i>…</i> -курсив



HTML5 Семантика – смысловая нагрузка, определяющая предназначение элемента.

Шапка – Header

Навигация – Nav

Главная область – Main

Статья – Article

Подвал (футер) – Footer

**CSS**

<div> - блочный элемент (самый часты используемый элемент в HTML)

# 1. Web-дизайн. Понятие объектной модели. HTML5-семантика.

Общая информация

* 0

Привет друг! Первое занятие уже позади и мы переходим к следующему.

Напоминаю, что ты можешь пользоваться таким вот мини-конспектом, чтобы освежить свои знания перед выполнением домашнего задания или если ты по какой-то причине пропустил занятие.

На втором занятии мы изучили следующие полезные штуки, а именно:

1. Объектная модель документа DOM и кто живёт в нём.
2. Что же такое семантика и откуда у HTML появилась цифра 5.
3. Как сделать списки разных видов за 2 минуты!

Давай теперь обо всём по порядку.

Объектная модель документа. DOM

* 0

Начнём с вопроса о том, что же такое за DOM и откуда это взялось.

Как ты уже догадался по заголовку, у этого DOM-а есть расшифровка: Document Object Model, но понятнее от этого не становится, поэтому идём дальше…

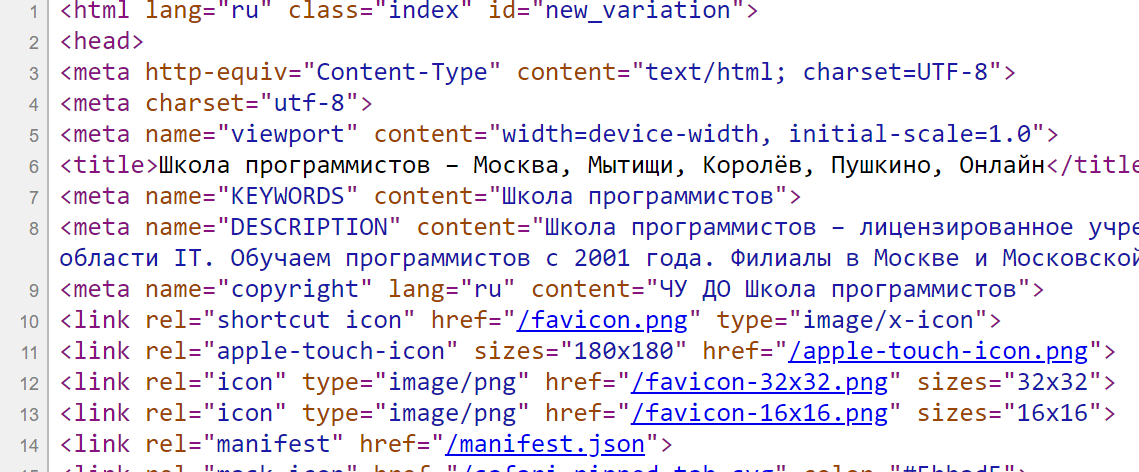
Давай попробуем задавать вопросы, тем самым узнаем новое и разберемся с нашим определением. Поехали!

**Вопрос № 1:** *DOM - это HTML код который я пишу?*

**Ответ:**Нет, не совсем так. HTML-код, который вы пишете, анализируется и преобразуется браузером впоследствии в DOM.

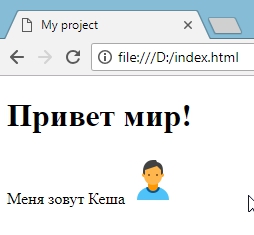
**Вопрос № 2:** *DOM - то что я вижу при нажатии Ctrl+U в браузере?*

**Ответ:**Близко, но всё ещё не то. . . Кстати, ты знал, что это называется “просмотром кода страницы”? Попробуй прямо сейчас. :)

****

**Вопрос № 3:** *DOM - то, что я вижу при нажатии F12 в Google Chrome?*

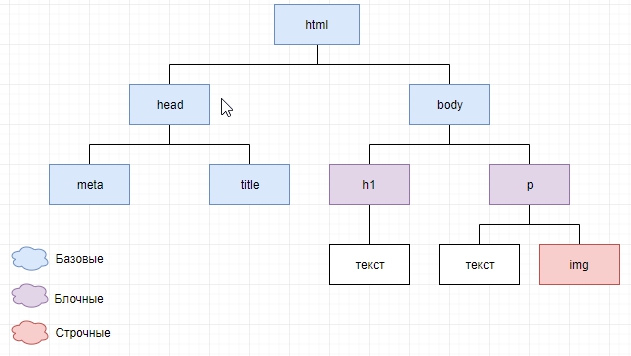
**Ответ:**В общем-то ДА! Этот блок называют панелью разработчика. А DOM – это представление документа в виде дерева.

****

*Пример странички*

****

*Пример кода такой страницы*

****

*Пример дерева по этому коду*

На самом деле про DOM можно было бы говорить очень много, но подробности ты сможешь узнать на курсе по JavaScript.

Семантика и HTML5

* 0

Прежде чем мы пойдем дальше, один важный момент по поводу тегов **<p>** и **<br>**. Ты помнишь их с прошлого урока, не так ли?

Так вот, тег **<p>** мы используем, чтобы сделать отдельный абзац текста, а тег **<br>** только лишь для того, чтобы сделать перенос на одну строку.

Использовать больше 2 тегов **<br>** подряд **нельзя**. Спустя несколько занятий ты сможешь узнать, как делают отступы опытные web-разработчики.

Чтобы лучше понимать о чём я говорю, посмотри на этот стих, ведь тут как раз используются теги **<p>**чтобы отделять между собой четверостишья, а тег **<br>** чтобы перейти на следующую строчку.

****

Мы несколько отвлеклись, вернёмся к понятиям. Возьми тетрадку и запиши их:

1. **HTML5** - это новая версия языка HTML, с новыми элементами, атрибутами и новым поведением.

**Семантика** - позволяет точнее описывать, что из себя представляет ваш контент.

Сейчас посмотри на картинку ниже. Это пример статьи. Подумай, какие части статьи ты можешь выделить. После пролистай ниже, чтобы проверить свои догадки.

****

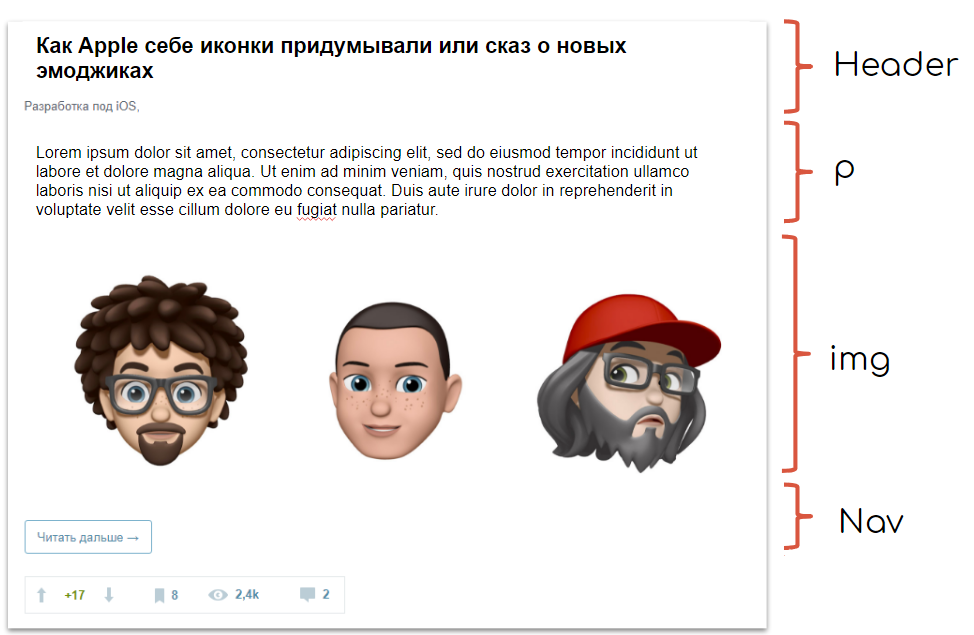
Хорошо, я уверен, что ты уже подумал и готов сравнить свой результат с ответом, прошу:

****

Первое, что ты, скорее всего, увидел, был заголовок статьи. Если мы говорим про сайт в целом, то эту часть называют шапка сайта, или header.

**header </> -**область заголовка, может включать в себя не только текстовую информацию, но и картинки, навигационные элементы.

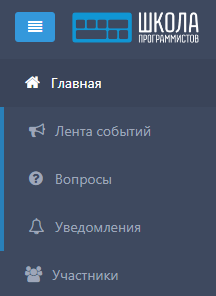
Далее ты можешь видеть основной блок, в котором находится весь контент нашей страницы. Там могут быть картинки, текст, графики, ссылки и т.д.

****

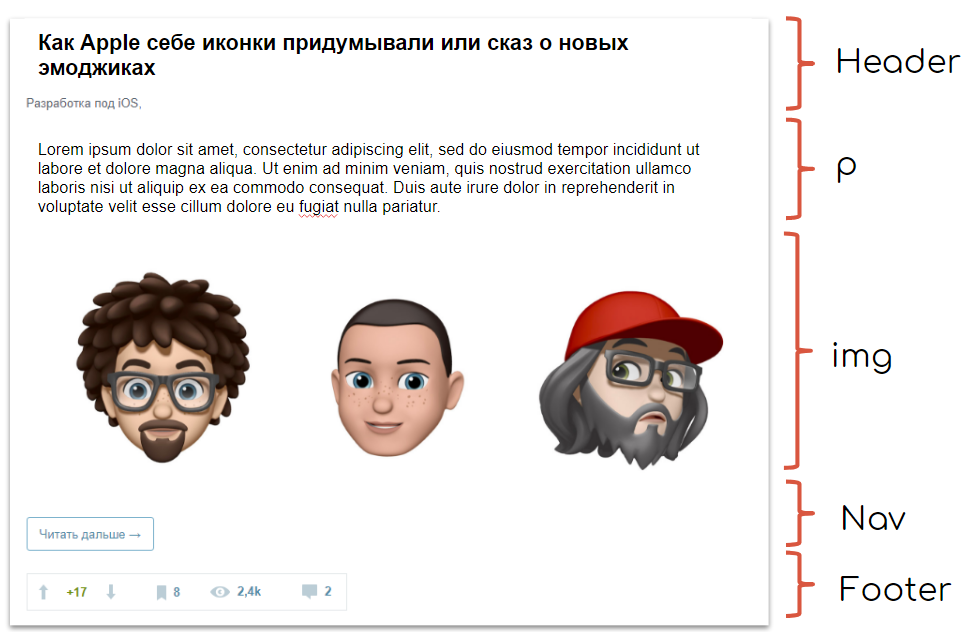
В рамках отдельно взятой статьи, вот эта кнопка “Читать дальше→” Действительно является навигацией, или по другому - элементом меню.

Еще есть тег **nav </>** - навигация, менюшка.

Если мы говорим про полноценную страницу, а не отдельную статью, то меню может выглядеть например вот так:

****

Вернемся к нашему примеру.

****

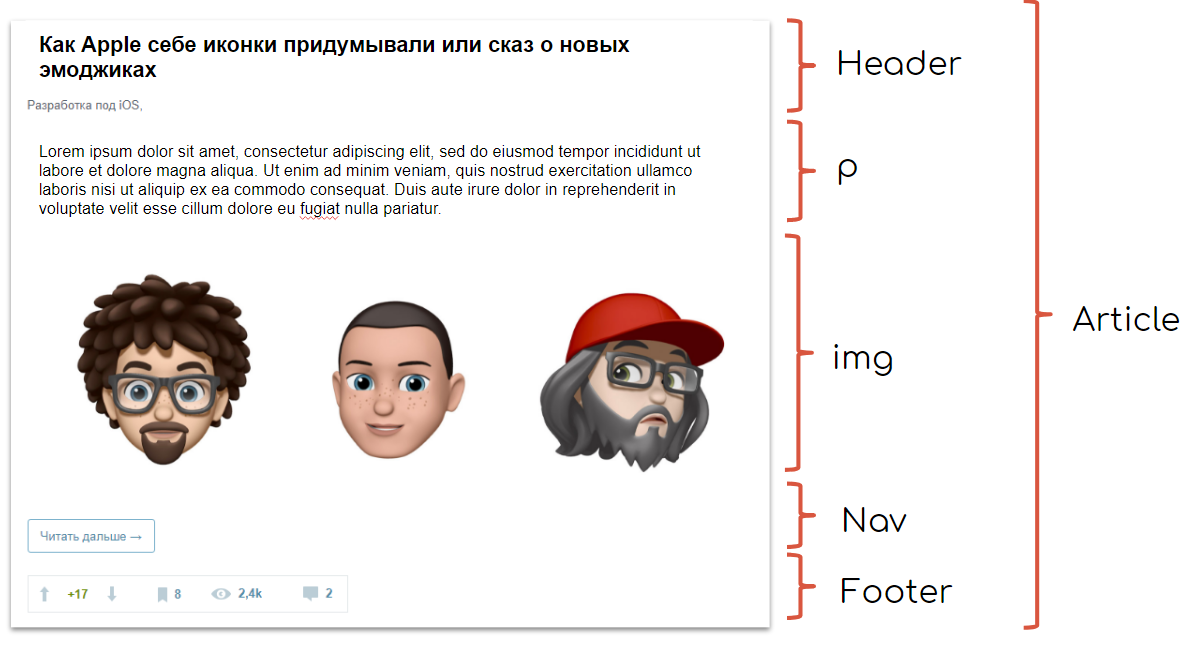
А вот и подвал или в кругах веб-разработчиков footer.

**footer </>** - подвал, определяет завершающую область сайта, документа, статьи или раздела.

Он есть, как у статьи, так и у сайта, через минуту расскажу о чём это я, но а пока ещё один тег.

Конечно же это тег обозначающий саму статью, без него просто никак!

**article </>** - статья.

****

Не могу оставить тебя без вкусного примера в виде кода.

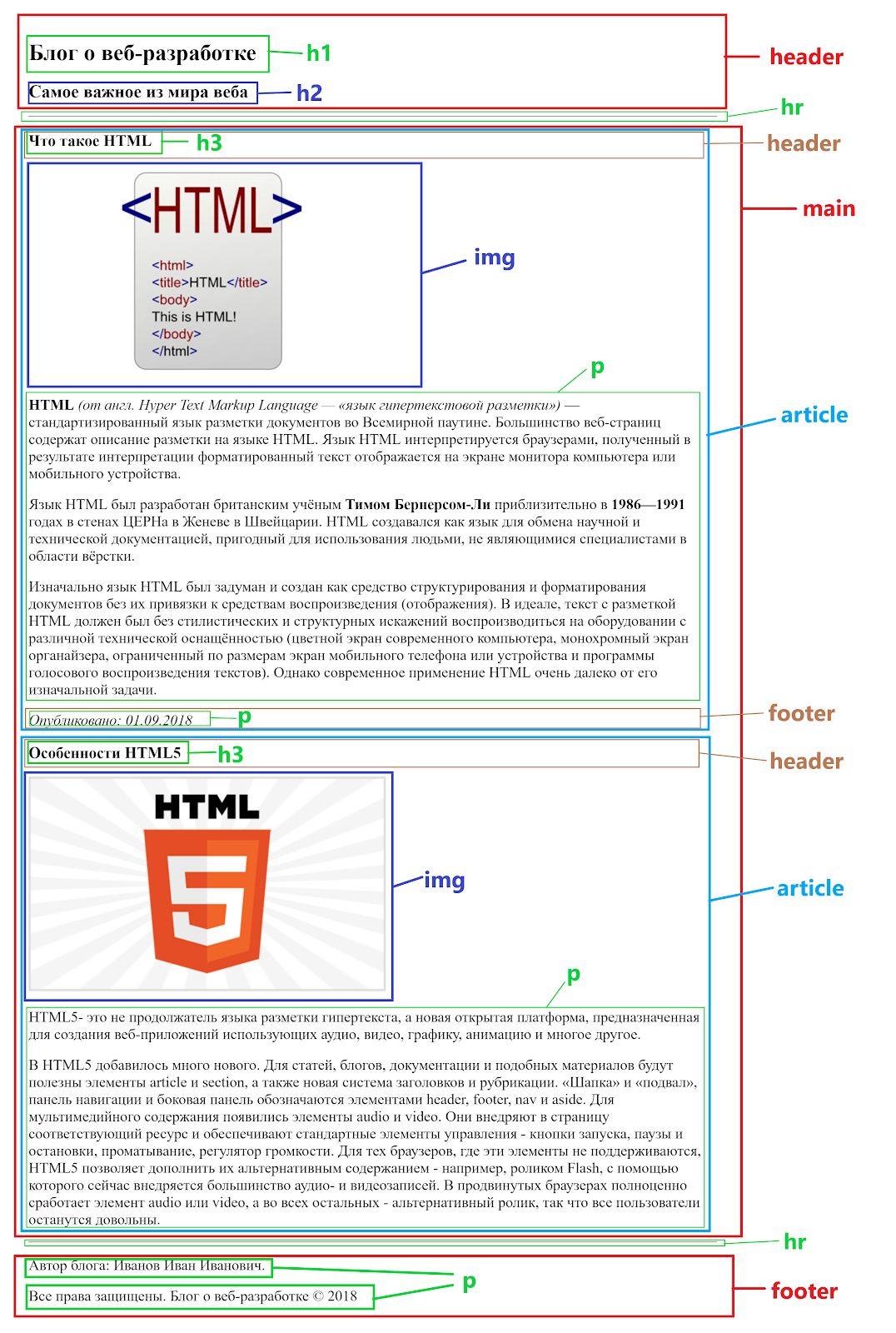
****

Посмотри внимательно на то, как строится структура страницы.

Теги вкладываются один в другой.

Есть ещё один очень классный и к тому же полезный тег, это тег **main </>** - область страницы, содержащая основной контент. Данный тег как и **<h1>** может быть **только один на страницу!**

Давай, чтобы тебе было понятнее, покажу полноценный пример. Посмотри внимательно на всю структуру в целом и на отдельные элементы данного макета.

****

Списки

* 0

Ты наверняка задался вопросом, а где же навигация? Ведь её нет на последнем скриншоте...

Хвалю за наблюдательность!

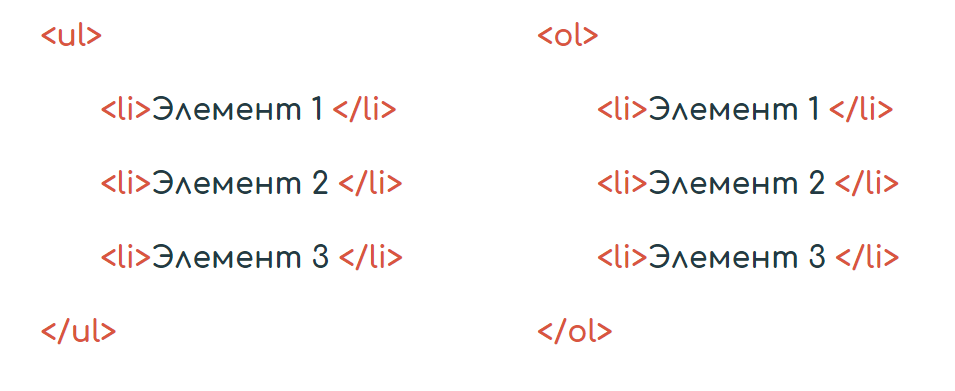
Но для того чтобы создавать навигацию, нужно изучить ещё несколько моментов, один из которых мы разберём прямо сейчас, а второй уже через урок. После этого можно будет перейти к изучению способа создания навигации.

ol </> - нумерованный список.

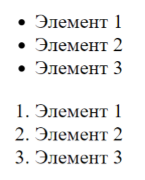
ul </> - маркированный список.

li </> - элемент списка

Так, ещё три новых тега?! Это всего лишь списки, не переживай, сейчас покажу пример и всё станет понятнее, вот как это выглядит в коде:

****

А вот как будет выглядеть результат в окне браузера:



На сегодня всё, ты осилил этот сложный урок и ты большой молодец!

Не забудь теперь выполнить домашнюю работу и до встречи на следующем занятии

13.10

<input type=”text” placeholder>

Calc() – считаем от ближайшего элемента у которого задан размер

20.10

Justify-content: center – центральное расположение

Display: flex – способ задания

Align-items: center

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

27.10

**Введение в vue.js**

https://cdnjs.cloudflare.com/ajax/libs/vue/2.5.17/vue.min.js

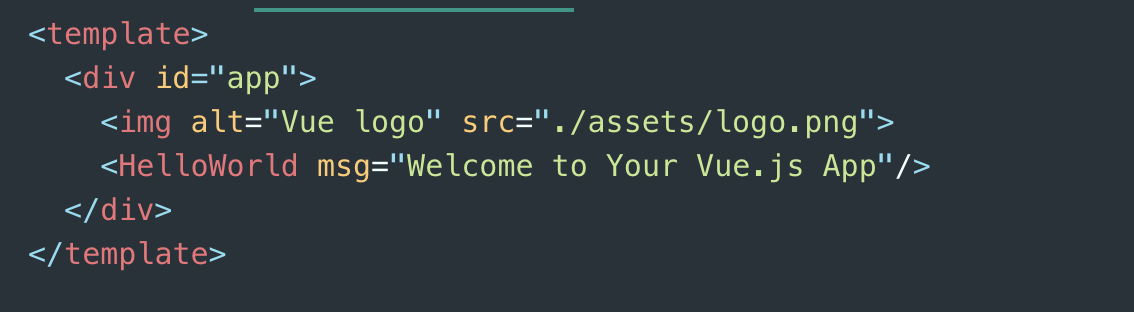
El: ‘#app’

Data : - данные тип {{type}}

Computed – выч свойсчва

Methods -методы

10.11



Блок template – html 1 корневой ник



Блок скрипт – логика, код

17.11

v-if – это отображение тегов по условию

24.11

Class User{

Constructor(username, age) {

This.username = username

This.age = age

}

printInfo(){

console.log(“Iam” + this.username)

console.log(“My age is “ + this.age)

}

}

08.12

Веб приложения

1. SPA – Single page application
2. PWA – Progressive web app
3. SSR – Server side recording

В html пропитывать в head в блоке мета (мета теги)

1. Keywords – ключевые слова
2. Title – заголовок страницы
3. Description – описание страницы

npx create-nuxt-app

Vuex – централизованное хранилище для различных данных

Сущности:

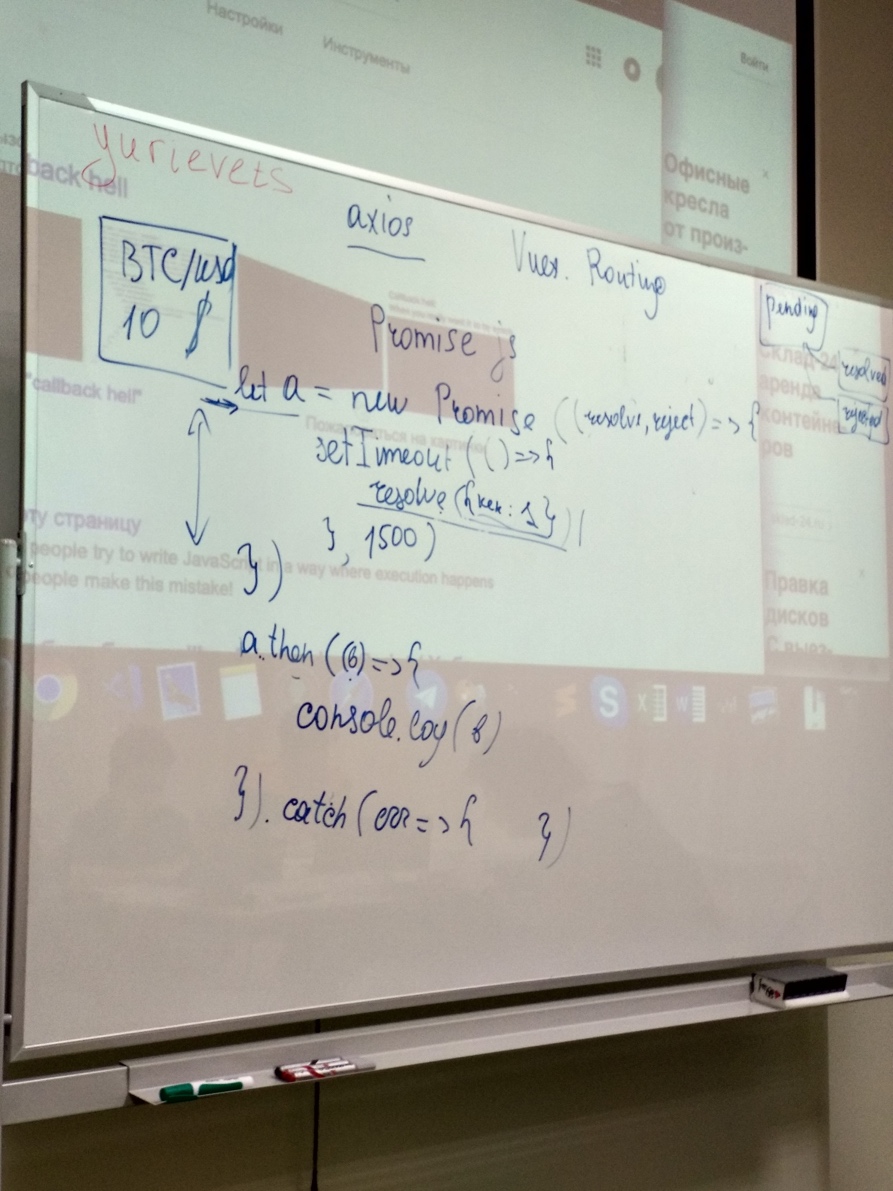
1. State – состояние, аналог даты
2. Mutation – изменение состояния, мутации делают mutate (как у гита), синхронны
3. Actions – мутации вызываться из action (изменяют мутации коммитами), явл. асинхронной
4. Getters – прослойка между состоянием из компонентом, возвращает данные из стейта
5. Из компонентов делается dispatch в actions

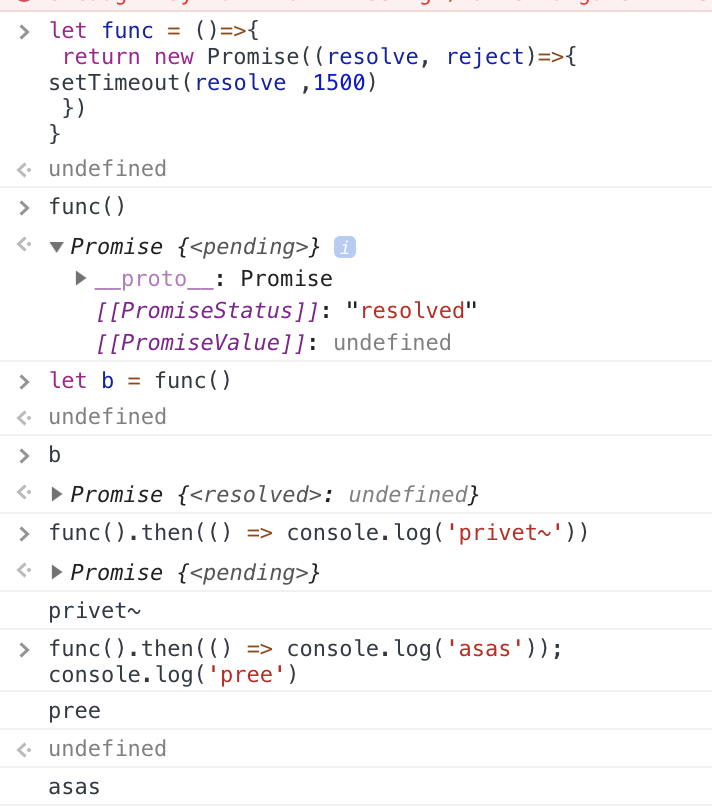
Получение глобальных переменных осуществляетсяй через $

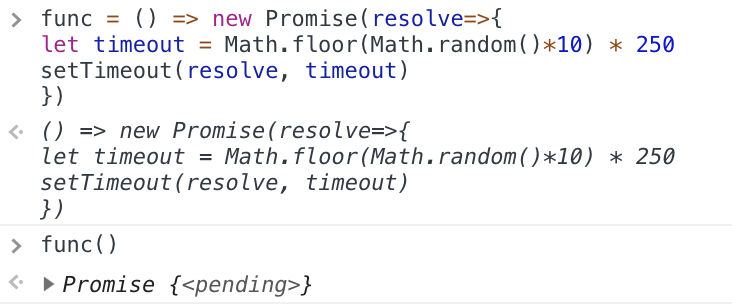
15.12

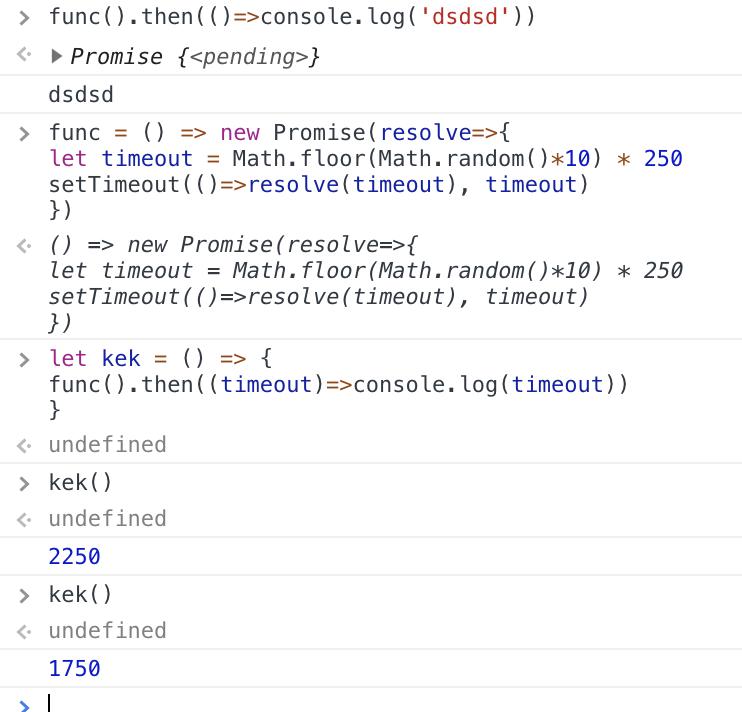
**Vuex Routing Access**

Promise(resolve,rejected)









Promise.all([]) – промис из промисов

