**JS**

1. Комменты:

//текст или /\*текст \*/

1. Переменная (без пробелов, не начинается с цифры)

var a значение может меняться

const А переназначить нельзя, обозначение нельзя менять

let a можно переназначить

1. Математические действия

const a = 11%3 %-остаток от деления

а++ добавить 1 а=+3 добавить 3

а-- отнять 1 а-= отнять 3

а \*\*2 возвести во 2 степень

1. Функции

function (){

console.log (“Делаем что-то»)

}

var ourFirstFunction = function(){

console.log(“Привет, мир!»)

}

Вызов функции: ourFirstFunction()

Привет, мир!

function (argument){

Console.log(«Передан аргумент: »+ argument);

}

argument = name = “Аня»

function(argument1, argument2){

console.log(«Первый аргумент: » + argument1);

console.log(“Второй аргумент:» + argument2);

}

Оператор return - возвращает значение

var double = function (number){

return number;

}

double(3)

6

double(5)+double(6);

22

double(double(3));

12

var pickRandomWord = function (words){

return words[Math.floor(Math.random) \*Words.length];

}

var randomWords = [«я», «ты», «они»];

pickRandomWord(randomWords);

Я

1. Объекты

1 Доступ через строки (=ключи/свойства)

2 Элемент-значение «ключ-значение» = 1+2

var cat={

“legs”: 3;

“name”: “Гармония»;

«colour”: “черепаховый»;

}

Можно без «»(leds=“legs”)

Литерал - когда значение записывается сразу и целиком

cat[“name”]; - доступ к значению (=cat.name - точечная нотация)

Object.keys(объект) - вывод ключей (выводит массивом)

Если + массив, то доступ -

массив[0]

Массив[0][ключ]

Массив[0].ключ - только для объектов

1. Циклы

while(condition){

console.log(«Делаем что-то»)

I++;

}

For(настройка;условие; приращение){

console.log(«Делаем что-то»)

}

1. Условия и циклы

Это управляющие конструкции

<script> -js код

Условные конструкции

1. If

2. 2 if … else

if(condition){

console.log(“Делаем что-то»)

}

if(condition){

console.log(“Делаем что-то»)

} else {

console.log(“Делаем что-то»)

}

if(condition1){

console.log(«сделай это, если условие 1 истинно»)

} else if (condition2){

console.log(сделай это, если условие 2 истинно»)

} else if(condition3){

console.log(«сделай это, если условие 3 истинно»)

} else{

console.log(“иначе сделай это»)

}

Ранний выход из функции

var fifthLetter = function (name){

if (name.length<5){

return;

}

return “пятая буква:»+ name(4);

}

var medalForScore = function (scope){

if(scope<3){

return «Бронза»

}

If(scope<7){

return “Серебро»

}

return “Золото»

}

function double(number){

return number \*2;

}

=

var double=function (number){

return number \*2;

}

1. Массивы

Свойства - хранят сведения

Методы - изменяют его

Массив.length - длина массива

Массив[массив.length -1] - последний элемент массива

1массив.concat(2массив) - склеивает массивы (можно и 3 и 4)

Массив.indexOf(“элемент массива») - определяет индекс в массиве !!если в массиве нет элемента, вернет -1, а если он встречается несколько раз, вернет индекс, который ближе к началу массива

Массив.join() - объединяет все элементы массива в одну строку !! Что в скобках, такой и разделитель

Math.random() - рандомное число от 0 до 1

Math.random()\*10 - рандомное число от 0 до 10

Math.floor(3,6372) - округляет до целого

Math.floor(Math.random()\*5)- объединяет два предыдущих метода(массив->случайный элемент из массива)

1. Висельница

prompt - диалоговое окно между <script>

var a= prompt (“…”) можно что-то написать

confirm-да/нет

var a = confirm (“?”)

if (a){

console.log();

} else {

console.log();

}

const user = {name: ”John”, age: 34}

const {name: userName, age: userAge} = yser

const user {

JohnDoe: {

age: 34,

email: “[John@jmail.com](mailto:John@jmail.com)”

}

}

Const {JohnDoe:{age, email}} = user

Const {JohnDoe:{age: userAge, email: userEmail}} = user

const[a,b] = [1, 2, 3, 4, 5, 6]

console.log(a, b) 1,2

const [a, b, , , c] = [1, 2, 3, 4, 5, 6]

console.log(a, b, c) 1, 2 5

Array.prototype.slice()

const[a, b, …arr] = [1, 2, 3, 4, 5, 6, 7]

console.log(a, b) 1,2

console.log(arr) 3, 4, 5, 6, 7

const profileUpdate = (profileData) =>{

const {name, age, nationality, location} = profileData:

}

=

const profileUpdate = ({name, age, nationality, location}) => { }

Object.freeze() – для того, чтобы данные не менялись

Const myF = function(){

Const a = “value”;

Return a;

}

=

Const myF = () => {

Const a =”value”;

Return a;

}

=

Const myF = () => “value”;

Const doubler = (item) => item \*2;

Doubler(4);

=

Const doubler = item => item \*2;

Const(multiplier) = (item, multi) => item \* multy

Const(greeting)=(name = “Ann”) => “Hi” + name

Function howMany(…args){

Return “You have passed” + arg.length + “arguments”

}

Var arr[6, 3, 89]

Var maximus = Math.max.apply(null, arr);

Или

Const arr = [6, 3, 89];

Const maximus = Math.max(…arr);

Math.max () – 2 аргумента в скобках. Возвращает наибольшее из нуля и более чисел

[arr1]=[…arr2] – добавляет значения массива, а не массив

Const user = {name: “John Doe”, age: 34};

=

Const {name,age} = user;

Const name = user.name

Const age = user.age

Do{

ourArray.push(i);

i++;

} while(i<5)

Const a=parseInt(“007”) 7

a?b:c a- условие, b – если да, с – если нет

function fruitsIs(val)

switch(fruit){

case “apple”:

console.log(“The fruit is an apple”);

break;

case “…”:

…

Break;

}

Switch (num){

Case value1:

Statement1;

Break;

Case value2:

Statemen2;

Break;

…

Default:

defaultStatement;

break;

}

\’ singlu quote

\” double quote

[\\ backslash](file:///\\backslash)

\n new link

\t tag

\r carriage return

\b backspace

\f formfeed

«строка».slice(1,5)-откуда и до куда вырезать

«Эй, ты!”.toUpperCase()

«ЭЙ, ТЫ!»

«Эй, ты!”.toLowerCase()

«эй, ты!»

a = b.length a=1.length считает количество знаков

a = bcd[1] a = c …[] – говорит, какая буква под номером в [], в джава скрипт счет начинается с 0

const a = “abc”

const b = a[a.length – 1] – если с -, то счет начинается с конца

[“a”, “b”, “c”] [“a”, 1] – массив

[[“a”, 1], [“b”, “c”] – вложенный массив

Const a = [1, 2, 3]

a[0] = 4

замена значений в массиве

.push() – значение в скобках в конец массива

.pop() – забирает элемент из конца массива

,shift() – забирает первый элемент из массива

.unshift() – добавляет элемент в начало массива

Function a(){

}

функция

Function (a,b) {

Console.log(a+b);

}

Функция с переменными, которая считает сумму переменных

Function a (){

Const b = “cc”;

Console.log (b);

}

Выводит на экран

A();

Console.log (b) не выводит

=()=>

== приравнивает и строки и цифры

=== строгое равно (или то, или то)

Typeof – определят тип переменной

i= - не равно

i== - без преображения

>больше <меньше

>=больше или равно <=меньше или равно

a.hasOwnProprty(свойство) – проверяет наличие свойства

булевые операторы:

1. && и
2. || или
3. ! не

Undefined – нет иного значения

Null – чтобы явно обозначить «тут пусто»

firstArray.concat(otherArray) – склеивает массив

.indexOf («элемент») – поиск индекса элемента в массиве

.join() – превращает массив в строку

.join(разделитель) – ставит вместо «разделитель» - разделитель(пробел, запятая или что угодно еще)

switch (fruit) {

case "apple":

console.log("The fruit is an apple");

break;

case "orange":

console.log("The fruit is an orange");

break;

}

Еще раз про свитч, работает как иф элс, всегда строгое равно===

Это все равно, что написать иф фрут === эпл конс лог () элс иф

Так короче и это удобнее

switch (num) {

case value1:

statement1;

break;

case value2:

statement2;

break;

...

default:

defaultStatement;

break;

}

Дефолт будет работать, если случаи не подошли

let result = "";

switch (val) {

case 1:

case 2:

case 3:

result = "1, 2, or 3";

break;

case 4:

result = "4 alone";

}

Чтобы объединить несколько случаев, нужно писать без брэйк, тоже вполне удобно, не нужно много раз писать одно и тоже

switch (val) {

case 1:

answer = "a";

break;

case 2:

answer = "b";

break;

default:

answer = "c";

}

Вот еще про синтаксис

function isEqual(a, b) {

if (a === b) {

return true;

} else {

return false;

}

}

Можно сократить до

function isEqual(a, b) {

return a === b;

}

const ourDog = {

"name": "Camper",

"legs": 4,

"tails": 1,

"friends": ["everything!"],

"bark": "bow-wow"

};

delete ourDog.bark;

позволяет удалить строку из объекта

const ourArray = [];

let i = 0;

do {

ourArray.push(i);

i++;

} while (i < 5);

А тут еще раз про ду вайл. Тут сначала выполняется цикл, а потом происходит проверка

alert -диалоговое окно alert(“…”)

break - выход из цикла

const person = {

name: "Zodiac Hasbro",

age: 56

};

const greeting = `Hello, my name is ${person.name}!

I am ${person.age} years old.`;

console.log(greeting);

это литерал шаблона, теперь не нужно писать через +, и использованы обратные кавычки. В литерале объема может быть несколько переменных

const getMousePosition = (x, y) => ({

x: x,

y: y

});

Равно

const getMousePosition = (x, y) => ({ x, y });

const person = {

name: "Taylor",

sayHello: function() {

return `Hello! My name is ${this.name}.`;

}

};

Равно

const person = {

name: "Taylor",

sayHello() {

return `Hello! My name is ${this.name}.`;

}

};

// Explicit constructor

class SpaceShuttle {

constructor(targetPlanet) {

this.targetPlanet = targetPlanet;

}

takeOff() {

console.log("To " + this.targetPlanet + "!");

}

}

// Implicit constructor

class Rocket {

launch() {

console.log("To the moon!");

}

}

const zeus = new SpaceShuttle('Jupiter');

// prints To Jupiter! in console

zeus.takeOff();

const atlas = new Rocket();

// prints To the moon! in console

atlas.launch();

class Book {

constructor(author) {

this.\_author = author;

}

// getter

get writer() {

return this.\_author;

}

// setter

set writer(updatedAuthor) {

this.\_author = updatedAuthor;

}

}

const novel = new Book('anonymous');

console.log(novel.writer);

novel.writer = 'newAuthor';

console.log(novel.writer);

The console would display the strings anonymous and newAuthor.

**CSS**

<style> - расположен в пределах <head>, позволяет контролировать стиль создаваемого элемента

element {

property: value;

}

Расположено в нем

 h1{

        text-align:center

      }

Теперь все, что в h1 расположено в центре

можно написать h1,h2,p, если все в одном стиле, чтобы не писать одно и тоже для каждого отдельно

так же для удобства можно создать отдельный файл со стилями, чтобы не перегружать код, а в самом коде оставить ссылку

в отдельном файле не нужно писать <style>

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

Ссылка, которая расположена в <head>!

rel="stylesheet"  - определяет, что ссылка содержит информацию, как стилизовать страницу

<meta name="viewport" content="width=device-width, initial-scale=1.0" /> расположено в <head>, чтобы на телефоне и на компе выглядело одинаково

body{

  background-color: brown;

} создает коричневую заливку, а burlywood кофейную, нежную

<div> - тег для дизайна

<div id=”menu”> - создает ссылку

#example-id {

width: 250px;

}

#menu{

  width: 300px;

}

Для того, чтобы тег диф не занимал всю страницы, а определенную часть страницы – в ширину

Можно использовать вместо пикселей проценты, аля 80% вместо 300 рх

/\* comment here \*/ для того, чтобы оставлять комментарии

 margin-left: auto;

  margin-right: auto;

меняет по горизонтали, чтобы было по центру все

.class-name {

styles

}

.menu {

  width: 80%;

  background-color: burlywood;

  margin-left: auto;

  margin-right: auto;

}

Классовый-селектор, теперь не через айди (<div class="menu">) Id меняем на class

body {

  background-image: url(https://cdn.freecodecamp.org/curriculum/css-cafe/beans.jpg);

}

Чтобы добавить фото на фон

<article> - еще один вариант элементов, обычно содержат несколько элементов, имеющих связанную информацию (<p> может быть внутри)

.flavor{

  text-align: left;

}

Текст влево

  <h2>Coffee</h2>

          <article class="item">

            <p class="flavor">French Vanilla</p>

            <p class="price">3.00</p>

          </article>

Чтобы было в 1 строчку, а не разбросано

.item p{

  display: inline-block;

}

Все элементы вложенные в р ведут себя как встроенные элементы (что-то я запуталась)

.flavor {

  text-align: left;

  width: 49%;

}

.price {

  text-align: right;

  width: 49%;

}

Так можно сделать, чтобы цена и кофе были с краев

.menu {

  width: 80%;

  background-color: burlywood;

  margin-left: auto;

  margin-right: auto;

  padding-left: 20px;

  padding-right: 20px;

}

 padding-left: 20px;

  padding-right: 20px;

тут создает пространство молочным коричневым и началом текста

 padding-top: 20px;- отступ сверху

padding-bottom: 20px; - отступ снизу

если все стороны одинаковые, то чтобы не писать

padding-left: 20px;

  padding-right: 20px;

 padding-left: 20px;

  padding-right: 20px;

можно просто написать padding: 20px;

 max-width:500px;– для того, чтобы на очень большом экране не было так, что цена и кофе слишком далеко

body {

  background-image: url(https://cdn.freecodecamp.org/curriculum/css-cafe/beans.jpg);

  font-family: sans-serif;

}

Тут font-family: sans-serif; задает шрифт для всего что в боди

h1, h2 {

  font-family: Impact, serif;

}

Так можно задать запасной шрифт(на всякий случай, мало ле)

h1{

  font-size: 40px;

}

Задает размер шрифта

 <hr> - для разделения !!самозакрывающийся

hr{

  height: 3px;

}

Изменяет толщину линии

hr {

  height: 3px;

  background-color: brown;

  border-color: brown

}

А так можно изменить цвет и границу заливки этой линии. По умолчанию толщина границы 1 пиксель

.item p {

  display: inline-block;

  margin-top: 5px;

  margin-bottom: 5px;

}

Настроить отступы между позициями в меню (межстрочный интервал что ли)

a{

  color: black;

}

Так можно изменить цвет ссылки, чтобы она не меняла цвет вне зависимости нажимали на нее или нет

a:visited{

  color: grey;

}

А вот так можно поменять цвет ссылки, если на нее нажали

a:hover{

  color: brown;

}

Теперь, если навести мышкой на ссылку, она станет коричневой, а не серой

a:active{

  color:white;

}

Теперь после того, как на ссылку нажали, она будет белого цвета, а не серого

**HTML**

 <!DOCTYPE html>- тэг в самом начале

<html> - 2 фраза, тут можно написать <html lang="en">

чтобы обозначить язык(английский)

<head> - это голова, тут можно дать название веб-страницы (будет написано на вкладке в браузере) при помощи <title>Cafe Menu </title>

<meta charset="utf-8"> расположено в <head> дает информацию браузеру, как кодировать элементы на этой странице !!самозакрывающийся

<body> идет после <head>, тут основной код

<main> расположен в <body>, несет основную информацию из страницы

<h1> тег, который делает большие буквы, 2 3 4 5 6 все меньше и меньше

<p> параграф, дает возможность писать текст буквами поменьше с новой строки, перед ним при написании кода нужно делать 2 пробела

<img> - для картинок, !!самозакрывающийся

src – атрибут, который указывает адрес, например для фотки: <img src-“https…”

alt- атрибут, который отображает текст, если, например, картинка не привязывается: <img src=”cat.jpg” alt=”A cat”>”

<a href=”ссылка”>тут может быть текст ссылки</a> - переход на другую страницу, а чтобы открыть ссылку в новой вкладке, нужно написать <a target=\_blank href=”ссылка”></a>

<section> - отделяет, создает автономный раздел

<ul> <ol>

<li>milk</li>

<li>cheese</li>

</ul> </ol>

Ul – создает список, когда перед пунктом будет стоять точка, а ol – список, когда перед пунктом цифра (1,2,3)

<figure> - объединяет надпись с картинкой

<figcaption> - создает текст, который описывает картинку (под картинкой)

<figure>

<figcaption>

</figcaption>

</figure>

<em>- подчеркивает текст <strong>-делает текст жирным

<form> - веб-форма

<action> - куда отправить данные формы, например: <form action="ссылка”>

<input> - собирает данные !!самозакрывающийся

<type> - тип сбора, например <input type=”text”>

Чтобы добавить комментарий:<!--комментарий -->

<name> - чтобы получить доступ к данным из месторасположения: <input type=”text” name=”email”>

Placeholder – подсказка, что ввести

Required – проверяет, заполнено ли поле (можно не задавать значение)

<button> - чтобы кликать=) если не ввести значение между <>, отправиться само собой

Синтаксис type: <input type=”text” name=”username”>

<input type=”submit” value=”отправить”>

<button type=”submit”>Submit<button> пример из фриКодКэмп

Radio – переключатель

Label – связывает с input- элементами

<label><input type=”radio”>cat</label> - позволяет щелкнуть на cat

Id – атрибут идентификации

<fieldset> - группирует связанные входные данные + делает новую строку, создает рамку

<legend>- элемент, действует как подпись к содержимому элемента fieldset – создает заголовок для этой группы, текст вписан в рамке

Type=”checkbox”- флажок, если есть несколько вариантов

Привязка к флажку через for:

<input id=”loving” type=”checkbox”>

<label for=”loving”>Loving</label>

Chtckbox – можно добать input, тогда будет вариант по умолчанию

<footer> - нижний колонтикулум

Основы html

Гипертекстовый - фрагменты текста свойственны гиперссылкам (ссылки на другие объекты)

Язык разметки - способ встраивать в текст дополнительную информацию

HTML - страница состоит из элементов, который начинается с открывания тела, заканчивается закрыванием

Блоковые элементы:

<h1> - заголовок 1 уровня

<p> - создает параграф

Строчные элементы:

<em> - курсив

<strong> - жирный шрифт

<!DOCTYPE html> - это “HTML-документ»

<html>- закрывается к конце, должен быть в каждом коде

<head> инфа про док

<title> текст на закладке в браузере

<body> - то, что будет в браузере

<a> - ссылка на интернет-адрес

<p><a href=“ссылка»>Нажмите сюда</p>

Можно добавить атрибут title

<a htef=“ссылка» title=“fhj:комиксы!»>Нажмите сюда</a>

Теперь при наведении курсора на ссылку будет виден текст (комиксы)