Курс по JavaScript

1. Emmet находится в VS Code по умолчанию.
2. Плагин JShint нужен для нахождения ошибок в коде (не установлен по умолчанию).
   1. Смотрим документацию JShint и устанавливаем через командную строку необходимые плагины (одной командой)
   2. Скачиваем Node JS
   3. Можно создать .jshintrc (для создания настроек для файла, если надо для нескольеих проектов, то кидаем этот файл в родительскую директорию для данных проектов)

Глава №4

Что такое JS и как его подключить к странице

1. electron js 🡪 десктопный фреймворк для создания приложений

Подключение JS к странице

1. Мы можем подключить js к html разметке двумя способами, первый из которых 🡪 это написание кода js внутри разметки, данный способ является нежелательным и не используется в больших проектах.

<script>

     alert("Open the button")

</script>

Данный скрипт выводит на экран сообщение перед каждой загрузкой страницы.

2. Второй способ 🡪 написание script в отдельном файле. Мы подключаем script в нашей вёрстке, но можем выполнить script и без вёрстки.

2.1 Подключение к вёрстке

<script src="js/script.js"></script>

2.2 Написание script

console.log(1)

В консоль выводится цифра ”1”.

Глава №5

Переменные и строгий режим

1. Переменные в js могут быть изменяемы и неизменяемые. Переменные, которые можно изменить идут с типом let.

2. Переменные, которые перезаписывать нельзя имеют тип const, если у нас const на словарь или подобную структуру, то в переменную нельзя перезаписать новый словарь, но можно изменить значения внутри переменной.

let ageUser = 32;

const ageMaster = 45;

console.log(ageUser);

console.log(ageMaster);

3. Так же существует устаревший тип var, его проблема, что переменные объявленные с таким типом могут использоваться до объявления в коде, что приводит к логическим ошибкам.

console.log(name);

var ageUser = 32;

4. Переменные видны только внутри блока кода, в котором они объявлены.

{

    let age =32;

    console.log(age);

}

console.log(age);

Во втором console.log(age) будет ошибка, так как он не видит нашу переменную.

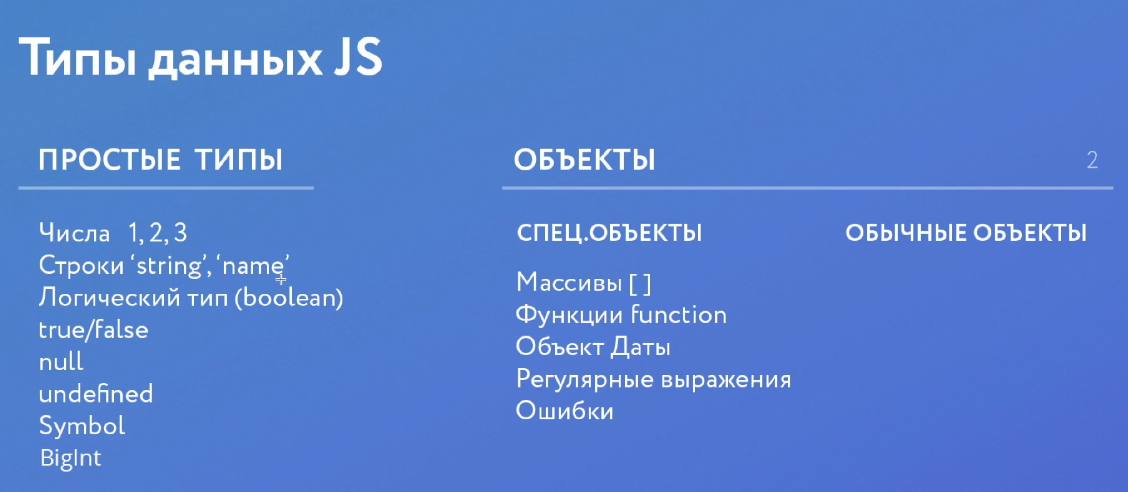
5. Чтобы указать, что мы используем современный код прописываем директиву “use scrict”, данная директива устраняет неточности при работе в современном режиме. Данную директиву прописываем в главном файле в самой первой строке.



6. В js переменные могут не иметь типа, но в современной нотации “use strict”, это неразрешено.

Глава №6

Классификация типов данных в JS



1. При делении на ноль мы не вылетим с ошибкой, а получим вывод infinity.

2. При умножении строки на число получим Nan

Типы ошибок

1. undefind 🡪 если переменная объявлена, но в ней ничего нет

2. null 🡪 если вызываем, то чего ещё не существует (код упадёт с ошибкой).

'use strict';

// Работаем с прмитивными типами данных

let age = 32;

console.log(age);

let name = "Sanya";

console.log(name);

// работаем с объектами

// массив и объект их создание и использование

// к объекту можно обратиться двумя способами,

// через точку или скобки

// массив не является собственным типом данных, он является

// разновидностью объекта.

let object = {

    age: 32,

    name: "Tolan",

    isMarriade: true

};

console.log(object.name);

console.log(object['age']);

const mass = [1, 2, 5, 6, 33];

console.log(mass[1]);

console.log(mass[3]);

Глава №7

Простое общение с пользователем

1. Команда alert, помогает связаться с пользователем, её основной минус, что мы не можем стилизовать окно данной команды, так как это является частью браузера.

2. Так же существуют ещё две команда для работы с пользователем, это confirm и prompt

'use strict';

// все функции : alert, confirm, prompt

// могут использоваться только в браузере

// так как они существуют только внутри браузера

// данные функии начинают своё выполнение ещё до создания разметки <body></body>

// при использовании alert

// пользователю выводится текст внутри alert

alert("Hello user");

// данная команда выводит пользователю окно ы выбором (да или нет)

// так же имеется возвращаемое булево значение в зависимости от нажатой кнопки

let bool = confirm("Вам есть 18?", ">18");

console.log(bool);

// данный метод предлагает пользователю ввести что-либо

// ответ пользователя можно записать в виде строки

// так же данные можно сразу конвертировать, в числовой формат,

// как показано ниже с помощью +

let inform = +prompt("Введите свой возраст");

console.log(inform, typeof(inform));

Глава №8

Интерполяция

'use strict';

//Для более кдобного фоматирования строк, можно использовать ососбые скобки

// как преведено ниже, мы внутри строки без знака + можем добавить в url

// часть переменной полученной из другого метода prompt

let path = prompt("Input path");

const url = `https://some.url//${path}`;

console.log(url);

Глава №9

Операторы в JS

'use strict';

// в JS практически все операторы имеют

// такой же смысл, как и в java

// но стоит выделить оператор сравнения

let age = 32;

let age\_1 = 33;

console.log(age + age\_1);

// в данном случае в первых двух строках будет выведено true

// так как мы сравниваем по значению, а не потипу

// в третьей строке будет выведено false

// так как мы уже сравниваем по значению и типу

console.log(2 \* 4 == 8);

console.log(2 \* 4 == '8');

console.log(2 \* 4 === '8');

// булевы операторы в JS так же стандартны

const isbool = true,

        isfalse = false;

console.log(isbool && isfalse);

console.log(isfalse || isfalse);

console.log((isfalse && isbool) || isbool);

console.log(isbool || isfalse != isfalse);

// важно помнить, что == сравнивает по значению,

// поэтому 6 == '6' будет true

// для сравнения по значению и типу используем

// оператор === и тут 6 === '6' будет false

Глава №12

Работа с объектом

'use strict';

let object = {

        movie: {}

};

// Считываем две переменные

let age = prompt("age", "23");

let name = prompt("Name", 'Tolan');

// внесли значение "age" в кавычках

// во втором случае внесли значение без кавычек

object.movie["age"] = age;

object.movie[name] = name;

// при выводе первый ключ выведется корректно

// второй ключ выведет undefind

console.log(object.movie["age"]);

console.log(object.movie["name"]);

Глава № 13

Условия

'use strict';

let age = 32;

// после первого верного условия мы выходим из кода

if(age == 32){

        console.log("Good");

}

else if(age < 50){

        console.log("Good1")

}

else{

        console.log("Bad");

}

// тернарный оператор позволяет выводить одно из двух условий

// после сравнения

(age > 30) ? console.log(age) : console.log(age + 1);

// switch case делает проверку по строгому ставрению

// по умолчанию можно поставить default

switch(age){

        case 30:

                console.log(30);

                break;

        case 32:

                console.log("32 + 1");

                break;

        case 100:

                console.log(100);

                break;

        default:

                console.log("Finallys")

                break;

}

Глава №14

Циклы

'use strict';

const num = 50;

// рассмотрим виды циклов в js

// первый цикл while

while(true){

        console.log(32);

        if(num == '50'){

                break;

        }

}

// второй цикл do while

do{

        console.log("iter");

}while(num > 50);

// третий цикл for

for(let i = 1, k = 2; i < 10 && k < 20; i++, k++){

        console.log(i, `${k}`);

}

Если мы преобразуем null в число, то он превратиться в 0, если преобразование просто невозможно мы к примеру пытаемся преобразовать строку в число, то вернётся NaN

Глава №16

Функции. Стрелочные функции

'use strict';

// переменные функции не видны вне функции

// так как это локальные переменные

// но внутри функции видны глобальные переменные

// если их изменить, то изменится и глобальная переменная

function showWhat(text, number){

        console.log(text, number);

        num = 10;

}

// функция с возвращаемым значением

function calc(firstArg, secondArg){

        return firstArg + secondArg;

}

let num = 35;

for(let i = 0; i < 3; i++){

        showWhat("Hello world", 2020);

        console.log(num);

}

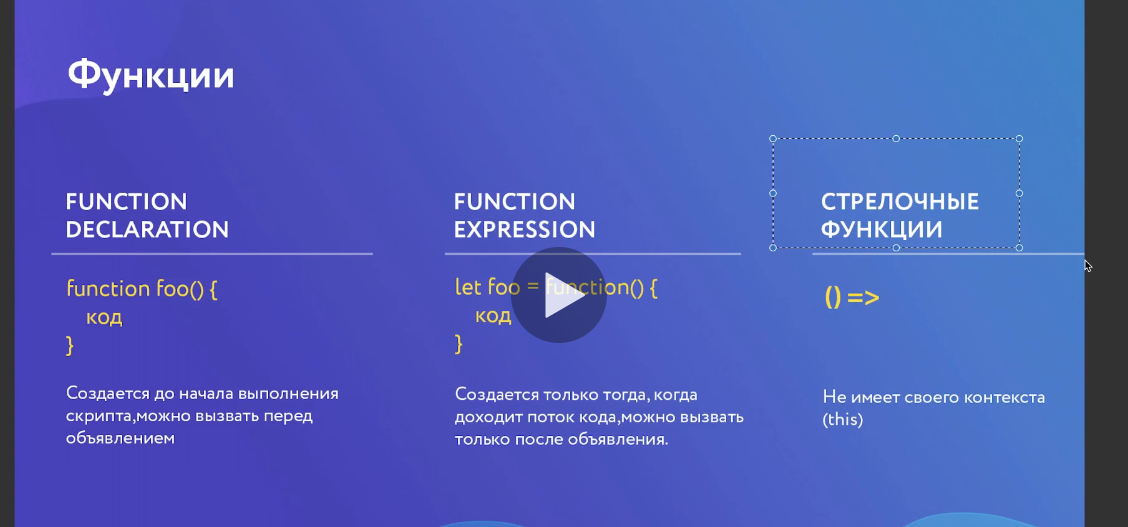
// проверяем функцию с calc, которая имеет

// возвращаемое значение

for(let a = 0, b = 1; a < 3; a++, b++){

        console.log(calc(a, b));

}



1. Функция function declaration, как и переменные var, создаются до начала работы программы.

'use strict';

// данный тип функции не создаётся до начала

// работы программы, а создаётся тогда, когда

// до него доходит код

let temp = function(firstArg, secondArg){

        console.log(firstArg + secondArg);

};

temp(1, 5);

// теперь можно рассмотреть стрелочные функции

// явлется более сжатой версией обычной функцией

// так же не забываем возвращать что-либо функции

// или вернйтся undefind

const calc = (a, b) => {return a + b};

console.log(calc(1, 7));

Глава №17

Методы и свойства строк и чисел

`use strict`;

// для строк уже написано множество методов

// у строки всего одно свойство length

// свойства пишутся без скобок в отличии от методов

// методы могут как менять строку, так и

// возвращать новую строку

const str = "Test\_String";

// переводит все символы в нижний регистр

console.log(str.toLowerCase());

// переводит все символы в верхний регистр

console.log(str.toUpperCase());

// показывает с какого момента начинается

// переданный аргумент в метод, где если

// часть аргумента не найдена вернётся -1

console.log(str.indexOf("Str"));

// делает срез строки, если не указать последний индекс

// то сделает срез до конца строки

// при указании последнего элемента, он не будет включаться

// метод поддерживает срез отрицательных чисел

// отсчёт идёт с конца строки начиная с -1

console.log(str.slice(5, 7));

console.log(str.slice(-3, -1));

// делает то же самое, что и slice

// не поддерживает отрицательные числа

console.log(str.substring(5, 11));

// для работы с математиой в js существует

// встроенна библиотека Math

const num = 12.5;

// округление вверх

console.log(Math.round(num));

// округление вниз

console.log(Math.floor(num));

// перевод чисел в другую систему счисления

// перводит строку в число по основанию 10

// надо указывать radix вторым параметром

// чтобы знать в какой системе находилось число изнчально

const text1 = "01111";

const num1 = parseInt(text1, 2);

console.log(num1, typeof(num1));

// возвращает число с плавающей точкой

const text2 = "34.567";

const temp1 = parseFloat(text2);

console.log(temp1, typeof(temp1));

Глава №19

Callback – функции

`use strict`;

// для проверки, что значение равно NaN

// для undefind и +undefind и пустого значния выведется true

if(isNaN()){

    console.log("good");

}

else{

    console.log("bad");

}

// в данной функции используется задержка

function first(){

    setTimeout(function(){

        console.log(1);

    }, 500);

}

// данная функция без задержки

function second(){

    console.log(2);

}

first();

second();

// создадим callback функцию

// передадим внутрь ней другую функцию

// вторая функция выполнится только после первой

function learnJS(arg1, arg2){

    console.log(`Я учу ${arg1}`);

    arg2();

}

// используем анонимную функцию

// анонимная функция исчезнет сразу после завершения метода, в который она

// была передана

learnJS('JS', function(){console.log("I end this lessons");});

Глава №20

Объекты, деструктуризация объектов

`use strict`;

// создаём объект

const options = {

    name: 'Sanya',

    width: 1024,

    height: 1024,

    color: {

        border: 'black',

        bg: 'red'

    }

};

// для удаления элемента из объекта

// используем оператор delete

// свойство будет полностью удалено

delete options.name;

console.log(options);

// для перебора всех свойств объектов

// используем цикл

// внутри цикла проверяем, что является ли наше

// свойство вложенным объектом

for (let key in options){

    if(typeof(options[key]) === 'object'){

        for(let i in options[key]){

            console.log(`войство ${i} имеет значение ${options[key][i]}`);

        }

    }

    else{

        console.log(`Свойство ${key} имеет значение ${options[key]}`);

    }

}

`use strict`;

// создаём объект

// внутри объекта создадим свойства

// так же создадим методы

const options = {

    name: 'Sanya',

    width: 1024,

    height: 1024,

    color: {

        border: 'black',

        bg: 'red'

    },

    makeTest: function(firsttArg, secondArg){

        return firsttArg + secondArg;

    }

};

// мы можем получить массив ключей из объекта

// исользуем interface Object и его метод keys

// потом берём у массива метод length и получаем

// кол-во свойств в объекте

console.log(Object.keys(options).length);

// вызовем метод из объекта

console.log(options.makeTest(1, 41));

// для деструктуризации, можно вытаскивать объекты

// из родительского объекта

const {border, bg} = options.color;

console.log(border);

console.log(bg);

Глава №21

Массивы и псевдомассивы

`use strict`;

// в данном блоке кода рассмотрим работу

// с массивом, а так же его методы

const arr = [1, 3, 4, 5, 0];

// удалить элемент из конца масива

arr.pop();

console.log(arr);

// добавить элемент в конец массива

arr.push(10);

console.log(arr);

// проходим циклом по всем элементам

for(let elem in arr){

    console.log(elem);

}

// важно понимать, что массив сожет иметь пустые ячейки

// если у нас массив имеет 6 заполненных элементов

// но его последний индекс 99, то его length = 100

let arr1 = [3, 5, 7];

arr1[99] = 0;

console.log(arr1, arr1.length);

// метод для перебора элементов в массиве

// его можно настраивать с помощью трёх параметров

// первый это получаемый элемент

// второй это его индекс в массиве

// третий это сам массив

// в данном массиве не работают break и continue

arr.forEach(function(item, i, arr){

    console.log(`${i} : ${item} внутри массива ${arr}`);

});

// чтобы превратить строку в массив используя

// разделитель применяем метод split()

let str = "apple, orange, bannana";

const product = str.split(", ");

console.log(product, product.length);

// чтобы объеденить массив в строку используем

// метод join, который принимает разделитель

// между элементами в строке

let str1 = product.join(", ");

console.log(str1, typeof(str1));

// метод для сортировки массивов sort()

// сортирует их как строки, поэтому для

// сортировки по значению пишем собсивенную

// функцию для сравнения значений

function sorts(a, b){

    return a - b;

}

const arr2 = [2, 13, 0, 234, 18, 1];

arr2.sort(sorts);

console.log(arr2);

Глава №22

Передача по ссылке или по значению

`use strict`;

// при передаче примитивов в переменную записывается

// байт код назначенного значения

// объекты передаются по ссылке

// чтобы скопировать объект можно использовать цикл

// данный метод может скопировать отдельные переменные

// но если у нас есть объект в объекте, то метод

// не сделает отдельную копию, а кинет уже существующую ссылку

function copy(mainObj){

    let objCopy = {};

    let key;

    for (key in mainObj){

        objCopy[key] = mainObj[key];

    }

    return objCopy;

}

const numbers = {

    a: 2,

    b: 5,

    c: {

        x: 8,

        y: 4

    }

};

const numbersNew = copy(numbers);

numbers.a = 323223;

console.log(numbers);

console.log(numbersNew);

`use strict`;

// чтобы соеденить несколько объектов в один

// используем метод интерфейса Object

// данный метод можно использовать для поверхностного клонирования

// объектов

const numbers = {

    a: 2,

    b: 5,

    c: {

        x: 8,

        y: 4

    }

};

const add = {

    d: 17,

    e: 20

};

// переносим один объект в другой

const tempConcat = Object.assign(numbers, add)

// клонируем поверхностно в пустой объект

// уже существующий массив add

const clone = Object.assign({}, add);

// При клонировании массивов

// можно использовать slice

// Получим два независимых массива

const arrayOld = [1, 3, 4];

const arrayNew = arrayOld.slice();

// чтобы переместить несколько массивов в один

// используем синтаксис преведённый ниже

// называемый оператором разворота

const video = ['youtube', 'vimeo', 'facebook'],

    blogs = ['wordpress', 'leveblog', 'blogger'],

    internet = [...video, ...blogs, 'vk', 'mail'];

console.log(internet);

console.log(typeof(internet));

// чтобы разложить массив на отдельные

// элементы используем (...mass)

let mass = [1, 7, 22];

function log(a, b, c){

    console.log(a, b, c);

}

log(...mass);

// данным методом можно тоже создать копию

// получаем ссылку на новый объект

const array1 = [...mass];

console.log(array1);

// для копирования объектов

const q = {

    one: 1,

    two: 2

};

const obj = {...q};

console.log(obj);

Глава №23

Основы ООП. Прототипно-ориентированное наследование.

`use strict`;

// в JS при использовании метода

// примитив оборачивается в объект

// потом применяется метод и возвращается обратно примитив

let str = 'some';

let strObj = new String(str);

console.log(typeof(str));

console.log(typeof(strObj));

console.dir([1, 3, 4]);

Для создания прототипов используем

`use strict`;

// создадим два объекта один из которых

// будет более глобальным чем второй

const soldier = {

    health: 400,

    armor: 100,

    sayHello: function(){

        console.log("Hello All");

    }

};

// отдельный объект

const john = {

      health: 120

};

// отдельный объект

const mark = {

    health: 200

};

// устанавливаем прототип для

// объекта john, данный метод

// является устаревшим

john.\_\_proto\_\_ = soldier;

console.log(john.sayHello());

// более современный метод создания прототипов

Object.setPrototypeOf(mark, soldier);

console.log(mark.armor);

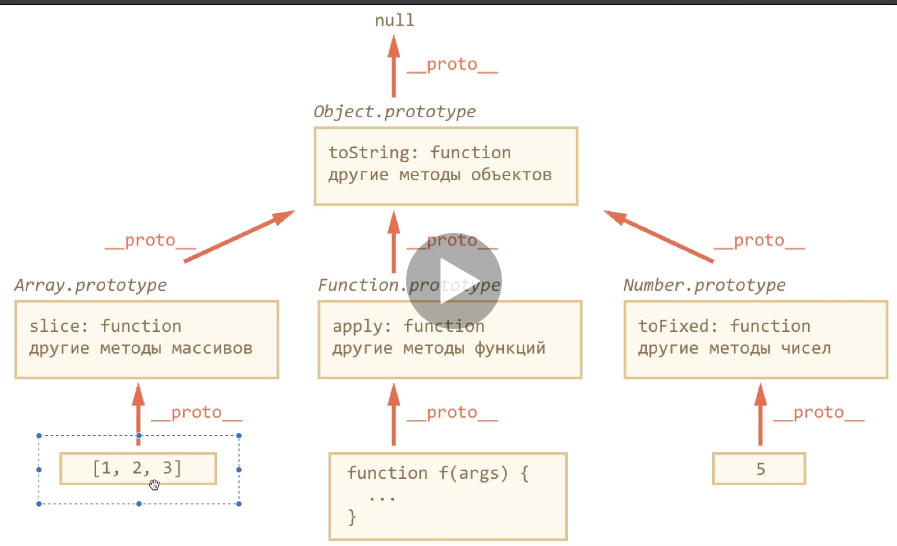
// для создания прототипа до инициализации объекта

const maks = Object.create(soldier);

console.log(maks.sayHello());

maks.armor = 400;

console.log(maks.armor);



Глава №26

Динамическая типизация в JS