**JavaScript**

JS. Работа с переменными, условный оператор JS.  
Циклы и массивы  
JS Работа с объектами JS.  
Функции, методы  
Практика (Работа с JS)  
JS. DOM (поиск элементов и изменение значений)  
JS. DOM (создание элементов)  
JS. DOM (Обработка событий)  
JS. DOM (Работа со стилями)  
JS Обработка формы.  
Объект event  
Практика (DOM)знакомство с REACT и создание проекта  
REACT работа с компонентами и props  
REACT useStateREACT useState  
Практика (верстка сайта с React)  
REACT useEffect  
REACT useContext  
REACT useMemo useCallback  
Практика (хуки,клиент-сервер)  
REACT useForm  
REACT Роутинг  
Redux знакомство с библиотекой  
Redux разбор практического применения  
Практика (реализация интерфейса с React и Redux)

**Вопросы**let i = 'hello'; Зачем объявлять строковую переменную ?????  
for (let i = 1; i <= 5; i++) {console.log(i); }  
console.log('last i', i);

typeof Number(1) === 'number'; // никогда не используйте эту запись! ??почему??

Не решали  
2) Замените код Function Expression стрелочной функцией:

    let answer = true;

    function ask(question, yes, no) {

      if (answer) yes()

      else no();

    }

    ask(

      "Вы согласны?",

      function() { alert("Вы согласились."); },

      function() { alert("Вы отменили выполнение."); }

    );

**SyntaxError** – нарушение правил языка, синтаксическая ошибка (запятая, скобка и т.д.)  
**ReferenceError** – неправильное имя, ошибочное название функции, перемеренной и т.д.  
**TypeError** – перепутатть одно с другим, описали переменную, а вызываем ее как функцию.  
**LogicError** – перепутать

**“use strict”** — это модель разработки дает возможность разработчикам использовать только ограниченный синтаксис и не допускать лишних ошибок. Прописываем это в первую строчку кода, а в самом проекте должен быть файл настроек **.jshintrc**

&& - AND

|| - OR  
"Debug"

Shift+F9 - zapysk v regime otladki;

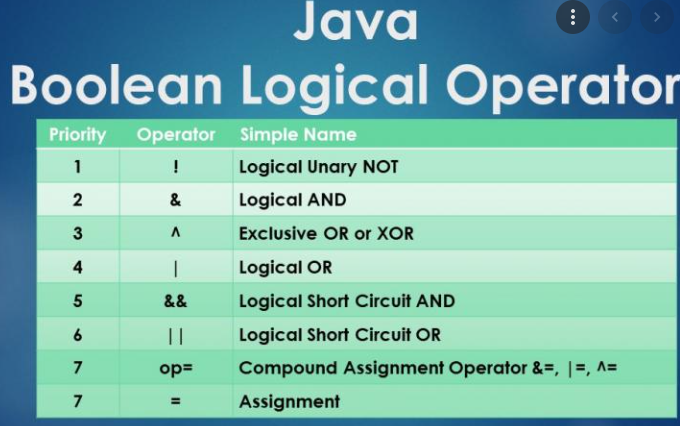
Ctrl+F8 - postavit 'break point';

F8 - one step;

F7 - one step(+metod);

F2 - pomenat znachenie vo vremia otladki;

Alt+F8 - vupolnit proizvolnii code;

|  |  |
| --- | --- |
|  | a &= b это a = a & b - операция Иa |= b это a = a | b - операция ИЛИ a ^= b это a = a ^ b - операция исключающее ИЛИ a += b это a = a + b a -= b это a = a - b |

**Операторы**

**Конкатена́ция** - операция склеивания объектов, например «микро» **+** «мир» == «микромир».  
**Интерполяция** - операция склеивания объектов через **` …${переменная}…`** (кавычки где Ё и ~)  
**Присвоение –** знак **=  
Отрицание** – знак **!  
Сравнение** – знак **== ,** 2\*4 == ‘8’ – сравнение по значению (true) **Строгое сравнение** - знак **=== ,** 2\*4 === ‘8’ строгое сравнение (false), сравнивается значение и тип переменных

**Инкремент** – операция увеличения переменной на 1 (обычно в счетчиках и циклах  
**Декремент** – операция уменьшения переменной на 1 (обычно в счетчиках и циклах  
Знак +/- можно ставить перед переменной (префикс) или после (постфикс).  
**Тернарный оператор/аргумент** (num===50)?(выполнение1 при true):(выполнение при false)

**Взаимодейсвие с пользователем в браузере**  
 **alert (‘предупреждение’)** – всплывающее окно в браузере с каким-либо предупреждением   
**confirm (‘вопрос пользователю’)** – предполагается только ответ **yes (true) и no(false)**  
**promt (‘вопрос пользователю’, ‘placeholder’)** – предполагается ввод данных пользователем. По умолчанию результатом будет тип string. Если +promt(‘’,’’) – то тип результата будет number

**typeof** *operand* - возвращает строку, указывающую тип операнда.  
**document.write(разметка)** - пишет строку в поток документа, открытый с помощью **document.open()** и после записи потока в файл закрытый с помощью **document.close()**

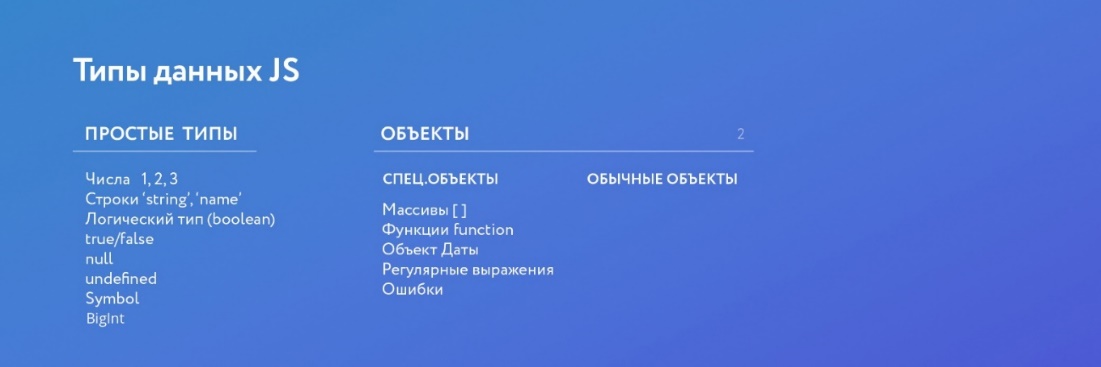
document.open(); - можно не использовать

document.write("<h1>Какой то заголовок</h1>");

document.close();

**Переменные**

**UpperCamelCase** – классы в Java (используется ли в JS?)  
**lowerCamelCase** – **переменные** с изменяемым значением  
**Upper\_Snake\_Case – константы** в JS  
**snake\_case** – для названия папок и файлов   
**PascalCase** – для названия **классов** в JS **Kebab-case** – для названия папок и файлов  
  
**Is not defined** – переменная не найдена (не объявлена)   
**undefined** – переменной не задано значение  
**Infinity** - бесконечноое значение (пример 4/0)

\ - экранирование переменной. \n - перевод строки \t – табуляция  
**Infinity** – бесконечное число, **true/false** – используется в циклах  
**NaN** является значением, представляющим не-число (Not-A-Number).  
**null** – пустое значение переменной, объявляется при объявлении переменной.   
**undefined** – -когда переменная не была инициализирована или не определилась  
**${b}** – вставка в текст значения переменной, текст заключается в обратные апострофы ` `  
let b = 10; console.log (‘Он сказал ${b} раз Привет’);  
 **false – это всегда: 0, null, ‘’, undefined, NaN  
true –** все остальные значения

**Объекты**let car = {

  color: 'green',

  maxSpeed: 250,

  audio: {

    brand: 'Sony',

    speakers: 12

  }  
car.color = ‘red’; // можно записывать car.[‘color’]

};

console.log(car);   
console.log(car.color);  
console.log(car.audio.brand);

**Условные инструкции if else и тернарный оператор**

**Конструкция иначе если:  
if** (car.color == 'red') {

console.log('Автомобиль красный');

}**else if** (car.color == 'green') {

console.log('Автомобиль зеленый');

}**else**{

console.log('Цвет не определен');

}

}

**Оператор отрицания:**if(!true) {  
 console.log('Истина');  
}

**Тернарный оператор, присвоение значения переменной:**let color = car.color == 'red' **?** 'Красный' **:** 'Зеленый';  
console.log(color);

**||** - оператор «или» **&&** - оператор «и»

**Функции**

**DRY – don’t repeat youself**let car = {

maxSpeed: 250,

};    
speedChange(300);   
function speedChange(theSpeed) {  
 console.log('Скорость, которая была=', car.maxSpeed);  
 car.maxSpeed = theSpeed;     
console.log('Скорость, которая стала=', car.maxSpeed);   
};

**Function ()** – Вызов /запуск /исполннение функции  
**Function без скобок** – обращение (**Referencing)**

let color = getColor;

color();

function getColor(color = "red") {

if (color == "red") {

return;

}

console.log(color);

}

**Cтрелочная нотация и анонимные функции**

**Методы в объектах и массивы**

let = array [1, 3, 10, 55];  
consol.log (array[2]) – выдаст число 2  
Массив может хранить данные разного вида в одном массиве – **негомогенный**

let drivers=['Ваня','Миша','Абрахам','Артём']   
let car={ driver:function(x ='Водителя под таким номером нету')   
{  
console.log('Текущий водитель: '+ drivers[x])}}   
car.driver(2)

let car = { color: 'red',   
drivers:['Sergey', 'Ivan'],   
getDriver: function(value)  
{  
console.log(`driver ${this.drivers[value]}`);//Шаблон в формате ES6 } } car.getDriver(1);

let myCar = { year: 2001, driver: ["Nikolas", "Victor"], getDriver: function() { console.log("Ваш водитель - " + this.driver[1]); } } myCar.getDriver();