# НАЧАЛО

**Emmet**

Ставлю ! и нажимаю TAB получается базовая HTML разметка

.class>button>div

<div class="class">

<button>

<div></div>

</button>

</div>

Alert – выводит какое-то сообщение с кнопкой OK

Confirm – выводит заданное сообщение с OK и Отмена. В зависимости от того, что нажмёт пользователь – ответ либо true или false.

const result = confirm("Are you here?");

console.log(result);

Promt – позволяет пользователю ввести данные в диалоговое окно

const answer = prompt("Вам есть 18 лет?", "18");

То, что введёт пользователь – всегда СТРОКА (нельзя складывать)

Если перед promt поставить +, то будет числовое значение

const answer = +prompt("Вам есть 18 лет?", "18");

Интерполяция – значит внутри строки можно вписывать переменные.

const category = 'cat';

console.log('https://domain.com/' + category); // конкатенация (сложение)

console.log(`https://domain.com/${category}`); // интерполяция

const user = "Ivan";

alert (`Привет, ${user}`);

## ОПЕРАТОРЫ JS

console.log("arr" + " - object"); //arr - object

console.log(4 + " - object");     //4 - object

console.log(4 + +" - object");    //Nan - not a number + после сложения означает, что в кавычках числовое значение

console.log(4 + +"5");            //9

/\* ИНКРИМЕНТ И ДИКРИМЕНТ \*/

let incr = 10,

    decr = 10;

   // префиксный - делает сразу операцию и возвращает результат СРАЗУ

console.log(incr++); //11

console.log(decr); //9

// ЕСЛИ СПЕРВА ВЫПОЛНЯЕМ ОПЕРАЦИЮ А ПОТОМ ВЫЗЫВАЕМ КОНСОЛЬ – то работает

incr++;

decr--;

    //ПОСТФИКСНЫЙ   - СНАЧАЛА ВОЗВРАЩАЕТ СТАРОЕ ЗНАЧЕНИЕ, чтобы получить значений переменной её нужно вызвать ещё раз

/\* ++incr;

--decr; \*/

/\* console.log(incr++); //11

console.log(decr--); //9 \*/

/\* console.log(incr);

console.log(decr);  \*/

console.log(--incr);

console.log(++decr);

console.log(incr);

console.log(decr);

console.log(5%2); // вернёт 1. т.к. это остаток.

Сравнение

console.log(2\*4 == 8);

выдаст **TRUE**

Сработает даже если 8 записать строкой

console.log(2\*4 == '8');

Для строгого равенства нужно ===

console.log(2\*4 === '8');

False (nfr как число сравнивается со строкой, если кавычки убрать, то снова true

Логические операторы И, ИЛИ (&&, ||)

Опертор && возвращает правду только, если оба выражения равны

|| возвращает когда оба верны или хотя бы 1.

const isChek = true,

        isClose = true;

    console.log(isChek&&isClose);

Результат true

Отрицание !

    const isChek = false,

        isClose = false;

    console.log(isChek||!isClose);

вернёт true, Так как второе было сперва false, но восклицательный знак дал отрицыние и оно превратилось в true. 1 из 2 условий выполнено.

# GIT система контроля версий

В рабочей папке

git init

git config --local user.name "Sasha"

git config --local user.email [hfgghf@gmail.com](mailto:hfgghf@gmail.com)

**Глобально:**

git config --global user.email [hfgghf@gmail.com](mailto:hfgghf@gmail.com)

**КОММИТ – контрольная точка**

**Статус репозитория:**

Git status

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

.jshintrc

JS.docx

index.html

js/

~$JS.docx

Добавить файл в отслеживаемые:

Git add-A (ВСЕ ФАЙЛЫ)

Git add main.css – (только 1 конкретный файл)

**Добавить файлы в КОММИТ**

git commit -a -m"5 commit"

-a = значит все файлы. M – месседж для сохранения (то, что в кавычках).

**Посмотреть все логи**

git log

**ГИТХАБ**

Заходим на гитхаб. Открываем новый репозиторий. <https://github.com/proektion1?tab=repositories>

git branch -M main

git remote add origin <https://github.com/proektion1/projekt.git> //адрес репоз

git push -u origin main

Потом просто с добавление нового файла или изменения

Git add-A – добавление

git commit -a -m"first commit" - коммит

git push – ЗАПУШИТЬ НА ГИТХАБ!

**СКАЧАТЬ НА ВТОРОЙ КОМП с ГИТХАБА СВОЙ ПРОЕКТ**

1. **Сперва в терминале переходим в нужную папку cd “** **C:\Users\SUPER GAMER\Desktop\work\”**
2. **Git clone** [**https://github.com/proektion1/projekt.git**](https://github.com/proektion1/projekt.git) **project2**
3. **Новый проект скачается в папку work, внутри папки project2**

**ЕСЛИ работаете с нескольких устройств – нужно обновлять папки**

**git pull**

**выйти из терминальнеых сообщений**

**Ctrl + С (2 раза)**

**:wq!**

# УСЛОВИЯ:

const num = 50;

 if (num<49) {

     console.log('Error');

 } else if (num>100) {

     console.log('More');

 } else {

     console.log ('Ok');

 }

**Тернарный оператор – в его работе учавствуют 3 аргумента.**

 const ff = 50;

 (ff === 50) ? console.log ('Ok') : console.log('Error');

**Бинарный аргемент 4 + 4. Унарный: +4.**

const yy = 50;

 switch (yy) {

     case 49:

        console.log('Неверно');

        break;

    case 100:

        console.log('Неверно');

        break;

     case 50:

        console.log('В точку!');

        break;

    default:

        console.log('В другой раз!');

        break;

 }

# ЦИКЛЫ

let num = 50;

while (num <55) {

    console.log(num);

    num++;

}

**ИЛИ**

let num = 50;

do {

    console.log(num);

    num++;

}

while (num<55);

**или**

for (let i = 1; i <8; i++) {

    console.log(num);

    num++;

}

**До 6 шага, потом прекратить**

for (let i = 1; i <8; i++) {

    if (i === 6) {

        break;

    }

    console.log(i);

}

**Исключить 6, и продолжить**

for (let i = 1; i <8; i++) {

    if (i === 6) {

       continue;

    }

    console.log(i);

}

# Функции, стрелочные функции

// создаём функцию

function showFirstMessage (text) {

    console.log(text);

}

showFirstMessage("Hello, world!"); // запускаем функцию и передаём ей параметр

Если создадим переменную внутри функции – она не выйдет из неё.

**Она называется локальной!**

function showFirstMessage (text) {

    console.log(text);

    let num = 20;

}

showFirstMessage("Hello, world!"); // запускаем функцию и передаём ей параметр

console.log(num);

**Глобальная может перезаписаться**

let num = 20; //глобальная переменная

function showFirstMessage (text) {

    console.log(text);

    num = 10; // перезаписываем переменную

}

console.log(num); // выведет 20

showFirstMessage("Hello, world!"); // запускаем функцию и передаём ей параметр

console.log(num); // выведет 10, так как запустилась функция, и в ней перезаписалась переменная

**Локальная и глобальная переменная**

let num1 = 20; //глобальная переменная

function showFirstMessage (text) {

    console.log(text);

    let num1 = 10; // локальная переменную

}

console.log(num1); // выведет 20

showFirstMessage("Hello, world!"); // запускаем функцию и передаём ей параметр

console.log(num1); // выведет 20, так как внутри функции была ОБЪЯВЛЕНА (не перезаписана) локальная переменная,

                // за её прееделами будет работать глобальная она будет только внутри функции

**ЗАМЫКАНИЕ ФУНКЦИИ – сама функция со всеми внешними переменными, которые ей доступны.**

function calc (a, b) {

    return (a + b); // возвратит сложение

}

console.log(calc(4, 3));

console.log(calc(5, 6));

function ret () {

    let num = 50;

    return num; // вернёт нам во внешний мир значение переменной

}

const anotherNum = ret(); // забираем результат из функции

console.log(anotherNum); // даст 50

****

1. Деклорейшн - создаётся сразу. Можно к ней обращаться в начале кода,

а сама функция может быть в конце. Браузер сперва считывает все такие функции.

В примерах выше, как раз она!

2. Экспрешн – только тогда, когда до неё доходит код

const logger = function() {

    console.log("Hello");

};

logger();

3. Стрелочная функция

const caalc = (a, b) => a + b;

# Методы и свойства строк и чисел

**Методы – вспомогательные функции**

**Свойства – вспомогательные значение**

**Свойство (подсчёт символов в переменной):**

const str = "test";

console.log(str.length);

**//4**

# Callback function

**Функция, которая должна быть выполнена после того, как другая завершилась.**

function learnJS(lang, callback) { //объявляем функцию с 2 аргументами

console.log(`Я учю: ${lang}`);

callback();                     //выполняем 2 функцию только после console.log

}

learnJS('Javascript', function() {      //запускаем функцию с 2 аргументами

    console.log('Я прошёл этот урок!'); //2 аргумент - функция с текстом

});

# Объекты

const options = {

    name: 'test',

    width: 1024,

    height: 1024,

    colors: {

        border: 'black',

        bg: 'red',

    }

};

console.log(options.name);

//delete options.name;

console.log(options);

for (let key in options) {

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

        for (let i in options[key]) {

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

        }

    } else {

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

    }

}

**Кол-во объектов в массиве**

const options = {

    name: 'test',

    width: 1024,

    height: 1024,

    colors: {

        border: 'black',

        bg: 'red',

    }

};

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

**Значения:** ${options[key]}

**Вывести ключи объектов:** Object.keys

**Вывести значения:** Object.values

**Ключ-значение:** Object.entries

**Подсчитать кол-во объектов в массиве** Object.keys.length

console.log(Object.keys(miniBD).length); //4

**Деструктуризация Объекта - вынос вложенных объектов из объектов (чтобы не писать некрасивый код:**

console.log(`Свойство ${i} иммеет значение ${miniBD[key][i]}`);

const miniBD = {

    type: 'short',

color: {

        line: 'red',

        topii: 'blue',

        general: 'red',

    }}

const {line, topii} = miniBD.color;

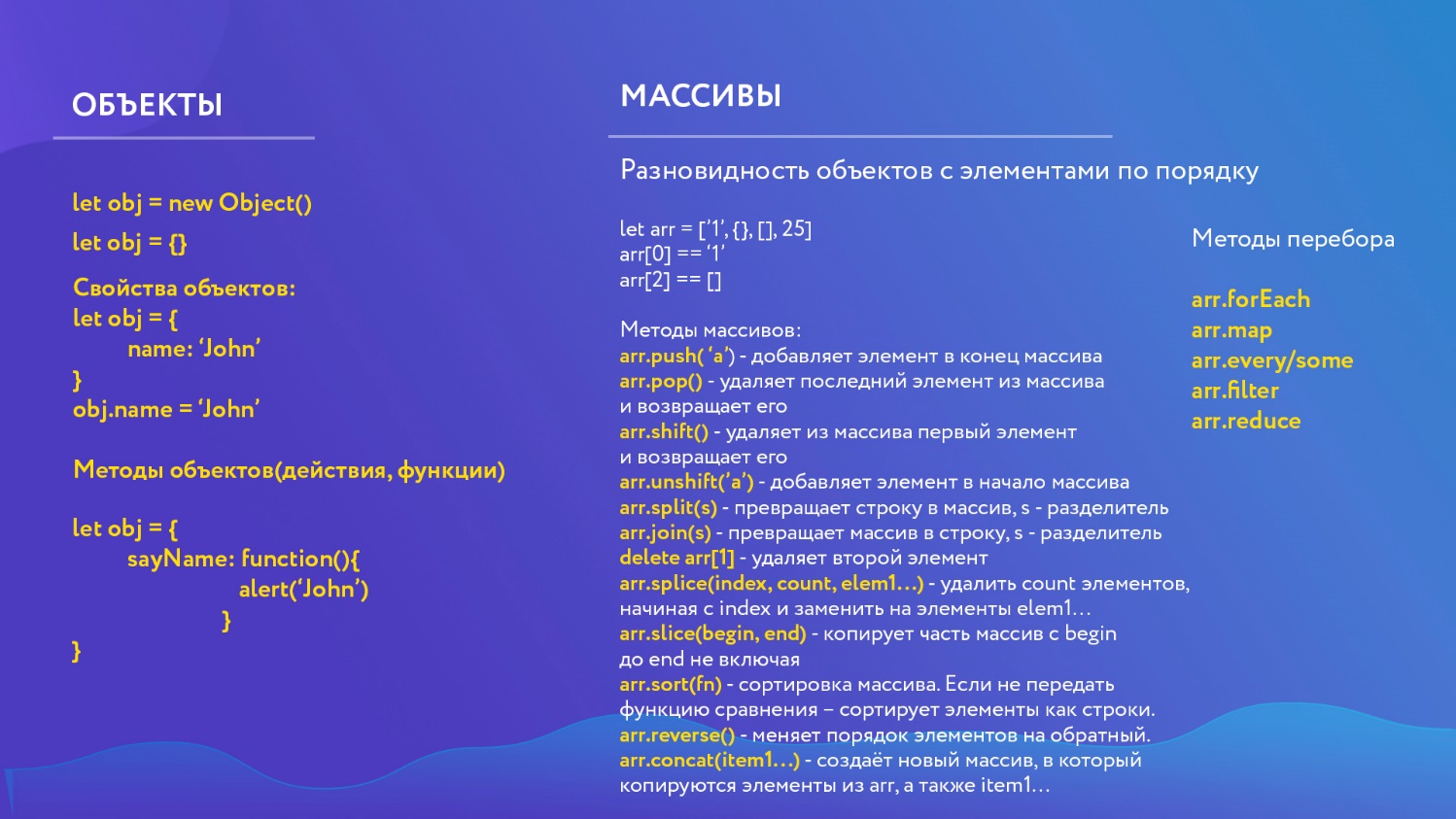
console.log(line);

**ПЕРЕБОР МАССИВОВ**

**FOR IN – перебирает индексы**

**FOR OF – перебирает значения**

**В for of и for in можно сделать continue и break. В FOREACH – нельзя.**

****

const arr = [1, 2, 3, 6, 8];

arr.forEach(function (item, i, arr) {

    //item - сами значения в массиве,

    //i - порядковый номер, arr - значения массива (все)

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

});

/\* 0: 1 внутри массива 1,2,3,6,8

1: 2 внутри массива 1,2,3,6,8

2: 3 внутри массива 1,2,3,6,8

3: 6 внутри массива 1,2,3,6,8

4: 8 внутри массива 1,2,3,6,8 \*/