**Компилятор сначала ищет все переменные var и все функции, только потом инициализирует переменные**

**Git**

**git init** команда слежения в гите за папкой(добавлять в папку проэкта)

**git config** – конфигурация может быть local и global(пишем после –)

**git config --global user.name "Alexandr work"** – устанавливаю имя, того кто работает с репозиторием

**git config --global user.email** [**iiii.love.ukraine@yandex.ru**](mailto:iiii.love.ukraine@yandex.ru) **–** емейл пользователя

**git status –** показывает статус репозитория, например какие файлы были изменены

**git add –A** –добавить все файлы что б за ними следили(-А значит все, можно указать конкретное имя)

**git commit –a –m”ddfd”** – делаем что б показать что изменили в этой конкретной версии(-а означает все, -м означает меседж, сообщение, в скобки пишем что изменили)

последовательность git add, git commit

**git log** – показывает какие комиты были и кем

**git remote add origin** [**https://github.com/AlexMorozan/-.git**](https://github.com/AlexMorozan/-.git)

заливаем наж уже созданый проэкт в репозиторий «origin” название

**git push** – Загрузить файлы в репозиторий

**git pull** – выгрузить файлы с репозитория

**git clone** + сылка на репозиторий – скопировать репозиторий по сылке, если работаешь на 2 компах

всегда сначала делать пул

**Function**

Обычная функция создаётся до начала выполнения кода и инициилизации переменных

function a() {dsfdsfdsfsdfs}

Функция в переменной создаёться только когда к ней доходит выполнения кода

let a = function() {djsvkldsmls}

Стрелочная функция let a = (a+b) => {

return a + b};

**Методы для строк**

Let str = “my name is alex”

srt.toUpperCase() = изменение регистра на большой

srt.toLowerCase() = изменение регистра на маленький

srt.indexOf(“alex”) = ищет индекс с которого начинаеться слово, можно так же использовать для поиска буквы или слова

srt.slice(с какого индекса вырезаем слово, до какого) – вырезаем часть строки

substring так же только без отрицательних значений

**Методы для чтсел**

Match – клас с методами

Match.round(3.5) = 3 – метод округления числа

**parseInt()**

Let a = “52px” parseInt(a)= 52 – преобразует в число

**parseFloat()** – такойже только возвращает не только целые числа

Обьекты

Const human = {

Name: “vasya”

Age: 23

Gender: “male”

Obj: {

Wight: 1368

Hight: 768}

delete – удалить поле обьекта

Пример = delete human.Name;

Object.keys(human).lenght; - показывает колво методов, типа записываем все поля в масив, и вызываем length для масива

Object.assign(первый обьект, второй обьект)= обьеденяим эти обьекты

Можно использовать что б в 1 обьект клонировать другой

**Деструктуризация обьекта**

Const {Wight,Hight} = human.Obj; вытащили значения вложеного обьекта в обьект

**Масивы**

let arr = [3,4,5,6,7,8,9,]

arr.pop() = удаляет последний элемент маства

arr.push(45) = добавляем єлемент в конец масива

arr.forEach(function(элемент, который мы перебираем, номер по порядку, сылка на масив, который мы перебираем)

{

}) = аналог форича, обычный форич используем когду нужно использовать break или continuum

Arr = arr1.slice(); копируем в масив арр масив арр1

… спрет оператор, комирует масив в масив превращая копируемый масив просто в набор данных arr1 = […arr]; через запятую можно добавлять несколько масивов и дополнять

