**Создание массива функций**

main.js

const funcs=[{

f:(x)=> x\*x;

color: ‘#f0f’,

width:2,

}

];

function render( ){

graph.clear( );

printOXY( );

funcs.forEach(func =>

func && printFunction(func.f, func.color,func.width));

}

UI.js

function UI({addFunction, delFunction}){

let num=0;

document.getElementById(‘addFunction’).addEventListener(‘click’,addClickHandler);

// в index.html добавить кнопку с названием добавить функцию с идентификатором addFunction, туда же добавить блок funcInputs

index.html

<button id=”addFunction”>+</button>

<div id=”funcInputs”/>

UI.js

function addClickHandler( ){

const input = document.createElement(‘input’);

input.setAttribute(‘placeholder’, ‘Функция №’ + num);

input.dataset.num = num;

input.addEventListener(‘keyup’, keyupHandler);

const button = document.createElement(‘button’);

button.innerHTML = ‘Удалить’;

button.addEventListener(‘click’,( )=>{

delFunction(input.dataset.num=0;

funcInputs.removeChild(input);

funcInputs.removeChild(button);

});

const funcInputs = document.getElementById(‘funcInputs’);

funcInputs.appendChild(input);

funcInputs.appendChild(button);

num++;

}

function keyupHandler( ){

try{

let f;

eval(`f=function(x){ return ${this.value} ;}`)

addFunction(f, this.dataset.num - 0);

}catch(e){

console.log(‘Ошибка ввода’, e);}

}

}

// конструкция trycatch может выполнить потенциально небезопасный код, не прерывая программу

main.js

const ui = new UI({addFunction, delFunction});

function addFunction(f, num){

funcs[num] = {

f,

color: black,

width: 3

};

render( );

}

function delFunction(num){

funcs[num] = null;

render( );

}

**GitHub.com**

Существует множество репозиториев, в котором хранится код со всеми изменениями. Каждое записанное изменение называется commit. Локальная копия создается на компьютере. Для того чтобы локальные изменения выложить, нужно их заpushить. Для того чтобы взять изменения с удалённого репозитория, их нужно заpullить. При заpushивании и заpullивании если имеются изменения, то система пытается решить конфликты самостоятельно, если не получается создает merge и предлагает решить пользователю самому. В репозитории commit’ы осуществляются в определённую ветку, в репозитории не ограниченое количество веток. Каждая ветка спланирована от другой ветки. master раньше, сейчас main – ветка изначальная.

Д/З: Касательную отмечать пунктиром(через производную… getDerivate(f, x0), то что в документе тоже сделать.