Объявление переменных

*var* - было

*let и const* - сейчас

В случае объявлении константы составного типа, его свойства можно менять. Разница между var и let’ом в области видимости объявляемых переменных. Область видимости var’а ограничивается функцией в которой он объявлен, у let’а в блоке.

*function f(){*

*if (1) {*

*let a;// a – только в if’е*

*}else{…}*

*}*

Области видимости let и const совпадают.

*function f(color = ‘#000’){*

*// цвет по умолчанию черный*

*}*

Способы задания свойства объекта

Если название переменной совпадает с названием свойств объекта, то WIN : WIN можно сократить до WIN.

Если необходимо в другой объект передать все свойства исходного объекта

Пример:

*const a = {*

*b.7,*

*c.8*

*},*

*const b = {*

*a:11,*

*…a*

*}*

*const {id,WIN,width,height=300} = options; || {};*

//деструктурирующее присваивание

Новые методы для массивов

По правилам хорошего тона в проге должны быть только константы.

*a.reverse* – переворачивает массив

*a.forEach(function(value,index,array){…});* – метод, который позволяет перебрать все элементы массива в переданной оперирующей функции.

*a.map* – возвращает новый массив собранный по какому-либо правилу с помощью итерирующей функции.

*const b = a.map(function(valut){*

*return value\*2;*

*});*

a.reduce – вычисляет и возвращает новое значение на основании итерирующей функции(Например: посчитать сумму элементов массива)

*a.reduce(function(S,val){*

*return S+val;*

*}*

Стрелочные функции

Объявляется как функциональный итерал.

const a = (аргументы) => {…}

Различия между обычными и стрелочными заключается в следующем:

* У стрелочной функции нет своего контекста(this будет искать выше)
* Стрелочную функцию невозможно использовать в качестве конструктора
* У стрелочной функции отсутствует специальная переменная arguments(специальный массив в котором лежат все аргументы функции)

Особенности записи стрелочной функции

Если у стрелочной функции больше 1 аргумента, () нужны, если 1, то можно () опустить.

Если у стрелочной функции в её теле присутствует только return, то {} и return можно опустить.

Правило хорошего тона: в лучшем случае все функции должны быть стрелочными. Обычная если необходимо контекст.

Перетаскивание графиков мышкой

Обратная функция перевода

sx = xs / canvas.width \* WIN.WIDTH

sy = …

function Graph({id, width=…, height=…, WIN, callbacks}){

let canvas;

if (id) {…} else{…}

const {wheel, mousemove, mouseleave, mouseup, mousedown} = callbacks;

canvas.addEventListener(‘wheel’, wheel);

this.sx = x => x\*WIN.WIDTH/canvas.width;

this.sy = y => -y\*WIN.HEIGHT/canvas.height;

}

main.js

const graph = new Graph({

id:’canvas’

width:…

height:…

WIN,

callbacks:{wheel, mouseup, mousedown, mousemove, mouseleave}

});

let canMove = false;

const mouseup = () =>{

canMove=false;

}

const mouseleave = () => {

canMove=false;

}

const mousedown = () => {

canMove=true;

}

const mousemove = (event) => {

if (canMove){

WIN.LEFT-=graph.sx(event.movementX);

WIN.BOTTOM-=graph.sy(event.movementY);

render();

}

Поиск нуля функции

function getZero(f, a, b, eps=0.0001){

if (f(a)\*f(b)>0)return null;

if (Math.abs(f(a)-f(b))<=eps) return(a+b)/2;

const half = (a+b)/2;

if (f(a)\*f(b) <= 0)

return getZero(f,a,half,eps);

if (f(half)\*f(b)<=0)

return getZero(f,half,b,eps);

}

Домашнее задание: переписать код на новый стандарт, научить таскать график, искать ноль функции на выбранном отрезке(с отдельным инпутом), найти все нули функции с использованием данного алгоритма.

//рисование стрелки x

graph.line(WIN.WIDTH + WIN.LEFT, 0, WIN.WIDTH + WIN.LEFT - WIN.WIDTH/100, WIN.HEIGHT/100);

graph.line(WIN.WIDTH + WIN.LEFT - WIN.WIDTH/100, WIN.HEIGHT/100, WIN.WIDTH + WIN.LEFT - WIN.WIDTH/100, -WIN.HEIGHT/100);

graph.line(WIN.WIDTH + WIN.LEFT - WIN.WIDTH/100, -WIN.HEIGHT/100, WIN.WIDTH + WIN.LEFT, 0);

//рисование стрелки y

graph.line(0, WIN.HEIGHT + WIN.BOTTOM, WIN.WIDTH/100, WIN.HEIGHT + WIN.BOTTOM - WIN.HEIGHT/100);

graph.line(WIN.WIDTH/100, WIN.HEIGHT + WIN.BOTTOM - WIN.HEIGHT/100, -WIN.WIDTH/100, WIN.HEIGHT + WIN.BOTTOM - WIN.HEIGHT/100);

graph.line(-WIN.WIDTH/100, WIN.HEIGHT + WIN.BOTTOM - WIN.HEIGHT/100, 0, WIN.HEIGHT + WIN.BOTTOM);