- Es6 Define as funcionalidades do Javascript (6 = Ano +1)

- Babel é compilador/Tradutor de Javascript para que a escrita do ES6 seja entendida por todos os navegadores.

Exemplo: Classes não são entendidas nos navegadores, então são convertidas para funções...

# Variaveis e Constantes

var podem ser reatribuídos

const tem valores fixos e não podem ser reatribuídos. **Nota: variáveis constates podem ter suas propriedades mutados.**

var variable1 = 3;

variable1 = 60;

const constant1 = 3;

constant1 = 22; //Erro durante a execução

# Escopo da Variável

**let é escopo do bloco, quando é redeclarado novamente, o valor é valido dentro do bloco atual**

let x = 1;

if (x === 1) {

let x = 2;

console.log(x); // Output: 2

}

console.log(x); // Output: 1

var é escopo da função, quando é redeclarado, o valor é valido em toda função

var x = 1;

if (x === 1) {

var x = 2;

console.log(x); // Output: 2

}

console.log(x); // Output: 2

# Strings

## Concatenação de Strings / Template Literals

console.log("meu nome é "+usuario.nome+" ");

console.log(`meu nome é ${usuario.nome}`); //uso da craseSh

# Classe

class Teste {

constructor(){ //Construtor

this.name = "test"; //Propridade

}

setName(name) { //Método

this.name = name

}

static staticValue() { //Metodo estático

return "Static value";

}

}

const object = new Teste();

Teste. staticValue();

# Herança

class Animal { //Super Classe

constructor(color) {

this.color = color;

}

}

class Horse extends Animal { //Classe

constructor(color) {

super(color); //Chamar o metodo da classe mãe

}

}

const myAnimal = new Horse("Red");

alert(myAnimal.color);

# Arrays

## Manipulação – MAP

var vetorNum = [1, 4, 2, 8, 21];

var newVetor = vetorNum.map(function(item, index){

return item \* 2;

});

console.log(newVetor); //[2, 8, 4, 16, 42]

## Filter

var vetorNum = [1, 4, 2, 8, 21];

var newVetor = vetorNum.filter(function(item){

return item%2 == 0;

});

console.log(newVetor); //[4, 2, 8]

## Find

var vetorNum = [1, 4, 2, 8, 21];

const isFound = vetorNum.find(function(item){

return item === 2

});

console.log(isFound) //2 (achou), se não undefined

## Soma

var vetorNum = [1, 4, 2, 8, 21];

const sum = vetorNum.reduce(function(previousValue, currentValue){

return previousValue + currentValue;

})

console.log(sum) //36

# Funções

Podendo ser função tradicional ou arrow function. Os parâmetros podem ter valores padrões.

function setTestNumber(number = 9){

console.log(number)

}

setTestNumber(22) //2

setTestNumber() //9

# Arrow Function

É Recomendada uso nas funções anônimas

const funcao = () => {

return 'Teste';

};

funcao()

var vetorNum = [1, 4, 2, 8, 21];

var newVetor = vetorNum.map(function(item, index){

return item \* 2;

});

console.log(newVetor); //[2, 8, 4, 16, 42]

Ficará

var vetorNum = [1, 4, 2, 8, 21];

var newVetor = vetorNum.map((item, index) => {

return item \* 2;

});

console.log(newVetor); //[2, 8, 4, 16, 42]

Quando a função somente retorna (sem corpo)

var newVetor = vetorNum.map((item, index) => item \* 2);

**Quando arrow function retorna um Objeto**

var newVetor = vetorNum.map((item, index) => ({

nome: 'Array',

valor: item

}));

# Desestruturação

var a = 10, b = 15, c = 30;

const [a, b, c, d] = [1, 2, 3, 4]; //Criar variavel a,b,c,d e colocar valores 1,2,3,4

console.log(a);

var usuario = {

nome: "Mota",

idade: 23,

endereco: {

cidade: "Santos",

Estado: "SP"

}

}

/\*

Criar variavel nome = usuario.nome

idade = usuario.idade

cidade = usario.endereco.cidade

\*/

var {nome, idade, endereco: {cidade}} = usuario

console.log(nome, idade, cidade);

var usuario = {

nome: "Mota",

idade: 23,

endereco: {

cidade: "Santos",

Estado: "SP"

}

}

function getNome ({ nome }) { //Receber como parametro objeto que tem propriedade usuario e extrair usuario.nome

return nome

};

console.log(getNome(usuario)); //Mota

# Rest/Spread

Rest e Spread são operadores adicionados por Dependência/plug-ins por enquanto, Rest tem função de colocar todos os outros parâmetros em objeto/array. Spead tem função de colocar abrir o array

Dependência: yarn add @babel/plugin-proposal-object-rest-spread

## Rest

var usuario = {

nome: "Mota",

idade: 23,

endereco: {

cidade: "Santos",

Estado: "SP"

}

}

const {nome, ...resto} = usuario //criar variavel nome = usuario.nome e um objeto que guarda o resto das propriedades

console.log(resto)

var test = [1,2,3,4];

const [a, b, ...resto] = test; //Criar variavel a = 1, b = 2, c = [3,4]

console.log()

function getValues(...params) {

return params; //Retorna array de parametros

}

## Spread

var array1 = [1,2,3];

var array2 = [4,5,6];

var arrayTotal = [...array1, ...array2]; //vai abrir array 1 e 2 para colocar os seus conteúdo em um único array

console.log(arrayTotal); //[1,2,3,4,5,6]

var usuario = {

nome: "Mota",

idade: 23,

endereco: {

cidade: "Santos",

Estado: "SP"

}

}

const usuario2 = {...usuario, nome: "Mota2"}; //Criar novo objeto copiando as propriedade do objeto usuário e só modificando a propriedade nome

console.log(usuario2)