**JavaScript Moderno (ES6)**

**O que é JS ES6**

‘Nova’ versão de JS (trouxe diversas funcionalidades);

Conhecimentos base para aprender frameworks modernos (React, Vue e etc);

Agiliza desenvolvimento com JS puro, deixando síntaxe mais ‘puras’ e bonitas, parecida com linguagens de alto nível.

**Let e Const**

Há duas novas formas de declarar variáveis no ES6;

A **Let** que é semelhante ao var (podemos alterar valores);

E a **Const** uma forma de declarar variáveis constantes (valor não pode ser alterado);

Grande diferencial é o escopo de blocos, variáveis não são mais vazadas para outros escopos por causa do mesmo nome.

**Arrow function**

Recurso para criar funções de forma simplificada;

Não funciona exatamente como function em todos os aspectos;

O this da arrow function é relacionado ao elemento pai de quem está executando.

const user = {

name: "Theo",

sayUserName() {//causou excesso de código, por não funcionar o this originalmente

var self = this

setTimeout(function () {

console.log(self)

console.log("Username: " + self.name)

}, 500)

},

sayUserNameArrow() {

setTimeout(() => {

console.log(this)

console.log("Username: " + this.name)

}, 700)

}

}

user.sayUserName()

user.sayUserNameArrow()//com a arrow function this é o elemento pai de fato

**Filter**

Método de array para filtrar dados, baseado em alguma condição estabelecida;

O resultado é um array com apenas os elementos que queremos, de foma performática;

No ES6 há vários métodos de array importantes, esse é um deles.

const users = [

    { name: "Ana", available: true },

    { name: "Pedro", available: false },

    { name: "João", available: false },

    { name: "Marcos", available: true }

]

const availableUsers = users.filter((user) => user.available)

const notAvailableUsers = users.filter((user) => !user.available)

**Map**

Método de array que percorre todos os elementos;

Quando é utilizado, o objetivo é modificar elementos;

É um método importante no ES6;

Modifica dado original.

const products = [

    { name: "Camisa", price: 10.99, category: "Roupas" },

    { name: "Chaleira elétrica", price: 49.99, category: "Eletro" },

    { name: "Fogão", price: 400, category: "Eletro" },

    { name: "Calça jeans", price: 50.99, category: "Roupas" },

]

products.map((product) => {

    if (product.category === "Roupas") {

        product.onSale = true

    }

})

**Template literals**

Permite a concatenação de uma string de uma forma mais simples;

Variáveis podem ser colocadas entre o texto;

String fica entre: `texto`;

Variáveis são colocadas deste modo: ${variável}.

***`texto${variável}`***

const userName = 'Matheus'

const age = 30

console.log(`O nome do usuário é: ${userName} e ele tem ${age} anos.`)

**Destructuring**

Pode ser utilizado em array e objetos;

O objetivo é transformar os itens desses dados em variáveis;

Simplifica a declaração de N variáveis para apenas uma linha.

***Array:***

const fruits = ["Maçã", "Laranja", "Mamão"]

const [f1, f2, f3] = fruits

***Objeto:***

const productDetails = {

    name: "Mouse",

    price: 39.99,

    category: "Periféricos",

    color: "Cinza"

}

const { name: productName, price, category: productCategory, color } = productDetails

**Spread operator**

Pode ser utilizado em arrays e objetos;

Utilizado para constituir novos valores destes em outros arrays e objetos;

Podem ser unidos ou adicionados diversos arrays de maneira simples.

const a1 = [1, 2, 3]

const a2 = [4, 5, 6]

const a3 = [...a1, ...a2]

console.log(a3)

const a4 = [0, ...a1, 4]

console.log(a4)

const carName = { name: "Gol" }

const carBrand = { brand: "VW" }

const otherInfos = { km: 100000, price: 49000 }

const car = { ...carName, ...carBrand, ...otherInfos, wheels: 4 }

console.log(car)

**Classes**

Recursos fundamentais para programar com orientação a objetos;

Agora o JS tem recursos implementados, onde temos acesso a: **constructor**, **propriedades**, **métodos** e mais;

Antes as classes eram baseadas e, funções (constructor functions).

class Product {

    constructor(name, price) {

        this.name = name

        this.price = price

    }

    productWithDiscount(discount) {

        return this.price \* ((100 - discount) / 100)//condição de desconto - subtrai desconto por cem, múltiplica pelo preço e divide por cem

    }

}

const shirt = new Product("Camila gola v", 20)

console.log(shirt.name)

console.log(shirt.productWithDiscount(10))

console.log(shirt.productWithDiscount(50))

const tenis = new Product("Tênis verde", 120)

console.log(tenis.productWithDiscount(20))

**Herança**

Com ES6, também podem ser criadas heranças de classes;

Para se referir a classe herdada, utiliza-se a palavra extends;

As propriedades dela que são utilizadas, são enviadas via função **super**.

class productWithAttributes extends Product {//herdando da classe(Product)

    constructor(name, price, colors) {

        super(name, price)//resgatando variáveis da classe acima

        this.colors = colors//novo atributo

    }

    showColors() {

        console.log("As cores são: ")

        this.colors.forEach((color) => {//executa função para cada elemento do array

            console.log(color)

        });

    }

}

const hat = new productWithAttributes("Chapéu", 29.99, ["Preto", "Azul", "Verde"])