IOS – Instituto de Oportunidade Social

# JS 10 - Classes e Funções

# Objetivos

- Compreender o conceito de classes e sua sintaxe;
- Conhecer os métodos de uma classe;
- Aprender o uso do this no contexto de funções.

IOS – Instituto de Oportunidade Social

# Classes

# Classes

**Classes** foram introduzidas no JS no **ECMAScript 2015**, mais conhecido como **ES6**, e elas são simplificações da linguagem para utilizarmos **herança** baseadas nos **protótipos**. Uma classe JavaScript não é um objeto e sim um **template** para objetos JavaScript. A sintaxe para classes não introduz um novo modelo de herança de **POO** em JS.

Classes em JavaScript provêm uma maneira mais simples e clara de **criar objetos** e lidar com herança. O bloco de instruções (corpo) da declaração de classes é executados em **modo estrito**.

# Classes

A sintaxe de uma classe em JavaScript é:

```
class NomeClasse {
        constructor() { ... }
}
```

Classe com duas propriedades (atributos/estados da classe)

```
class Carro {
        constructor(nome, ano) {
                this.nome = nome;
                this.ano = ano;
        }
}
```

IOS – Instituto de Oportunidade Social

# Métodos de uma Classe

# Métodos de uma Classe

Os **métodos** de uma classe são criados com a **mesma sintaxe** de um **método de objetos**. Além do constructor, você pode criar diversos protótipos de métodos na classe:

```
class NomeClasse {
        constructor() { ... }
        metodo_1() { ... }
        metodo_2() { ... }
        metodo_3() { ... }
}
```

# Métodos de uma Classe

Exemplo de Classe com métodos:

```
class NovoCarro {
        constructor(nome, ano) {
                this.nome = nome;
                this.ano = ano;
        }
        idadeCarro(ano) {
                return ano - this.ano;  //cálculo no retorno
        }
 }
```

# Métodos de uma Classe

Trabalhando com datas:

```
let dataHoje = new Date();
let ano = dataHoje.getFullYear();

// Intanciando o objeto à classe
let meuNovoCarro = new NovoCarro('Ford', 2014);

console.log(meuNovoCarro.idadeCarro(ano));
```

A Classe **Date** retorna a **data atual** com dia, mês, ano, hora, minutos, segundos, e o método **GetFullYear** retorna o ano de uma data.

IOS – Instituto de Oportunidade Social

# Lexical This

# Lexical This

**Lexical this** no contexto de Funções

Podemos utilizar a palavra-chave **this** no **contexto de funções**. O valor this é determinado pela forma como a função é chamada. Ex:

- Em um **método**, o this faz referência ao **próprio objeto**.
- **Sozinho**, o this faz referência ao **objeto global**.
- Em uma **função**, o this faz referência ao **objeto global**.
- Em uma **função**, no **modo estrito**, o this é **undefined**.
- Em um **evento**, o this faz referência ao **elemento** que disparou.
- Em **método** como **call()** e **apply()** podem fazer referência do this para **qualquer objeto**.

# Lexical This

**Lexical this** em uma função:

```
function PessoaFunc(firstName, lastName, dob) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.dob = new Date(dob);
        this.getBirthYear = function () {
                return this.dob.getFullYear();
        };
        this.getFullName = function () {
                return `${this.firstName} ${this.lastName}`;
        };
}
```

# Lexical This

Instanciando os objetos de Funções:

```
const pessoa3 = new PessoaFunc('John', 'Doe', '1980-04-03');
const pessoa4 = new PessoaFunc('Marry', 'Smith', '1970-06-13');

console.log(pessoa3);
console.log(pessoa4.dob.getFullYear());
```

IOS – Instituto de Oportunidade Social

# Prototype de objetos

# Prototype de objetos

O JS permite criar **protótipos de objetos**, que basicamente é **permitir criar atributos** ou **métodos** em objetos já criados anteriormente. E esses novos protótipos são **herdados automaticamente**.

# Prototype de objetos

Adicionando prototypes:

```
PessoaFunc.prototype.getBirthDayMonth = function () {
        let dados = [this.dob.getDate(), this.dob.getMonth() + 1];
        return dados;
};

PessoaFunc.prototype.getFirstName = function () {
        return `${this.firstName}`;
};
console.log(pessoa3.getBirthDayMonth());
console.log(pessoa4.getFirstName());
```

IOS – Instituto de Oportunidade Social

# Vamos Praticar

# Vamos Praticar

Apostila de JS

04.JavaScript

Páginas 129 a 138

OBS: Acompanhar o passo a passo com o instrutor

IOS – Instituto de Oportunidade Social

# Para aprender mais

# Para aprender mais

Procure sempre aprender e estudar mais. Seguem alguns links para você estudar e aprender mais:

Classes:
https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Classes
https://www.w3schools.com/js/js_classes.asp

this
https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Operators/this
https://www.w3schools.com/js/js_this.asp

# Para aprender mais

Modo estrito:
https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Strict_mode
https://www.w3schools.com/js/js_strict.asp

Prototype:
https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Objects/Object_prototypes
https://www.w3schools.com/js/js_object_prototypes.asp

IOS – Instituto de Oportunidade Social

# Exercícios

# Exercícios

Criar uma **classe pessoa** com as propriedades **nome** e **dataNascimento** e o método **getIdade()** e um **array de pessoas** com 10 posições, instanciar a classe pessoa em cada índice do array, adicionar o método **getNiver()** por **prototype** que retorna true se já fez aniversário e false se ainda não fez no ano corrente, exibir o texto “Fulano tem X anos e **já/ainda não** fez aniversário esse ano” para cada pessoa do array (Utilizar **If** e **For**).

**getMonth()** retorna o mês de uma data (0 a 11)
**getDate()** retorna o dia do mês de uma data (1 a 31).