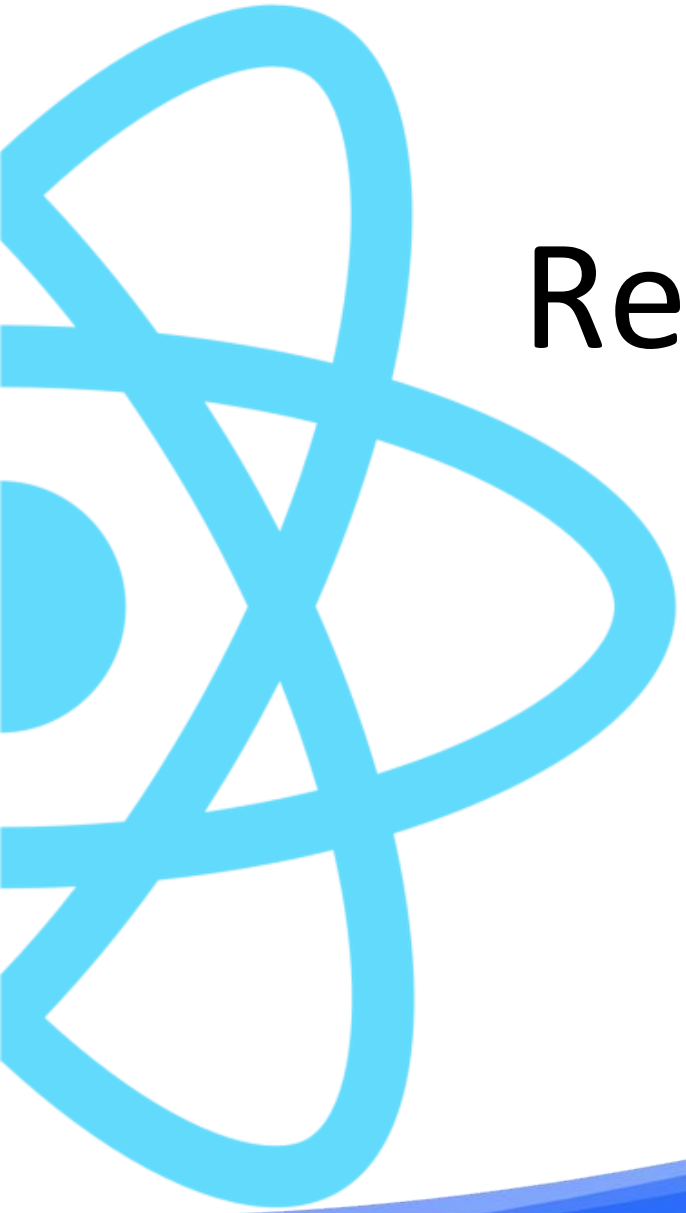


# Residência de Software 2020

## React JS

# Como dar instruções?

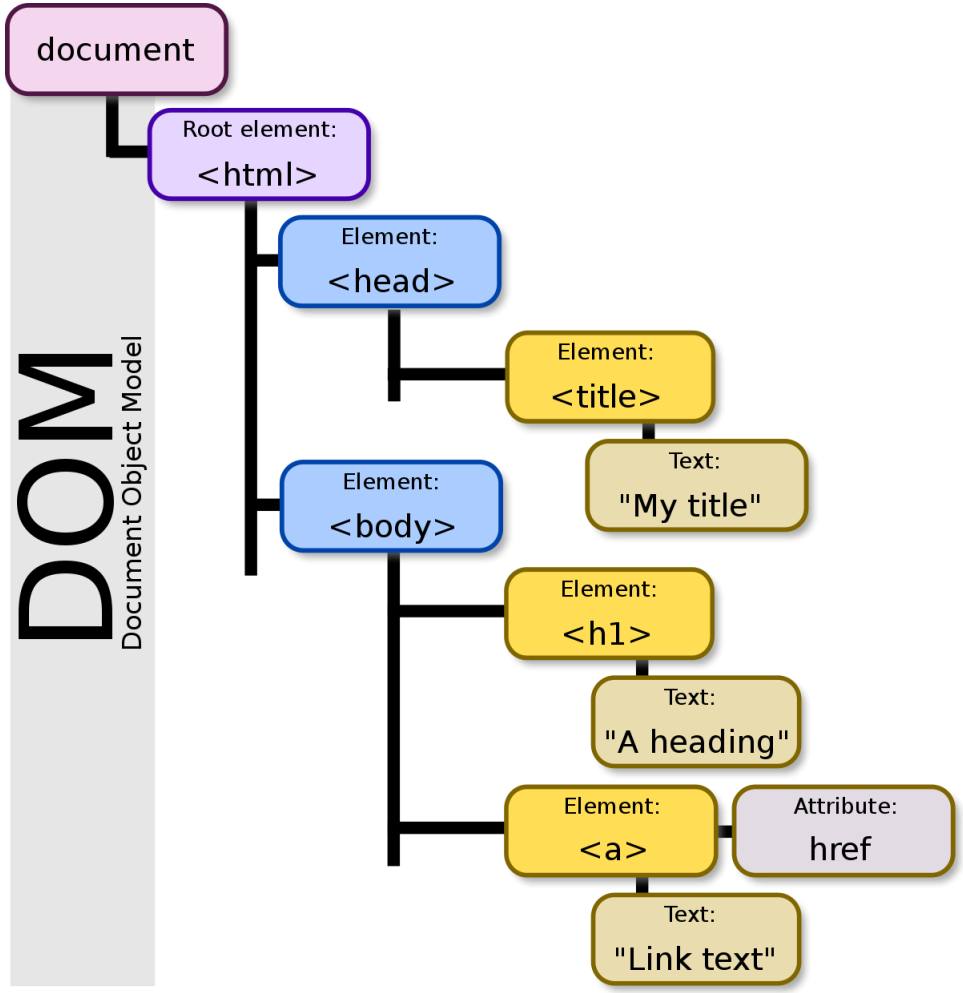

JS
ES6

# O que vamos ver hoje

- O que é DOM?
- Como Manipular a DOM?
- O que são APIs
- Javascript
  - Template String
  - Funções anônimas
  - Arrays e Objetos
    - Spread operator
    - Desestruturação
    - Métodos auxiliares de arrays (map e filter)

# O que é DOM

- Document Object Model ou simplesmente DOM;
- Utilizado pelo navegador Web para representar a sua página Web;
- Quando altera-se o DOM altera-se a página Web;
- O Javascript é capaz de alterar o DOM;
- É mais fácil trabalhar com DOM do que diretamente com código HTML ou CSS;



# Como manipular o DOM

- Qualquer coisa criada pelo navegador Web no modelo da página Web poderá ser acessado através do objeto Javascript document;
- Usa-se o DOM principalmente para atualizar uma página Web
- Pode-se mover itens dentro de uma página ou criar efeitos CSS sem precisar recarregar a página.

- `documentElement`
- `getElementById`
- `createElement`
- `getElementsByTagName`

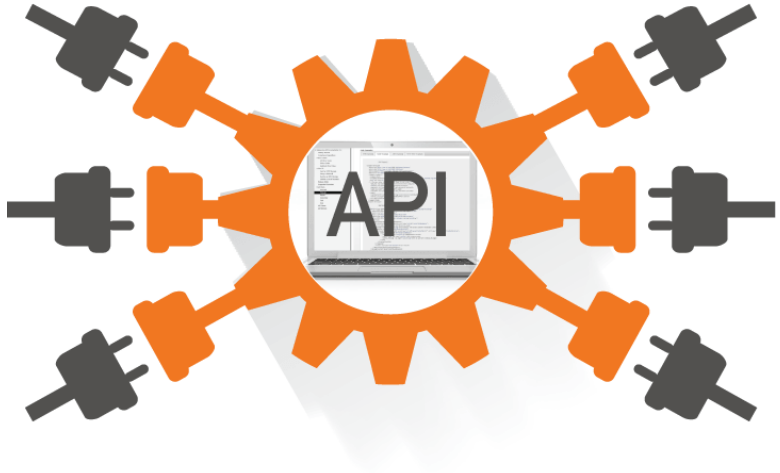
# O que são APIs

Application Programming Interfaces

Construções que permitem criar funcionalidades complexas mais facilmente.

Construções abstraem o código mais complexo;

Proporcionam o uso de sintaxes mais simples em seu lugar.



# O que são APIs

- Não fazem parte da linguagem em si;
- Mas são escritas sobre o *core* da linguagem JavaScript
- Fornecem superpoderes para serem utilizados em seu código.
- São duas categorias:

    - **APIs de navegadores:**
    - **APIs de terceiros:**

# O que são APIs

- **APIs de navegadores:**
  - Fazem parte do seu navegador web, expõem dados do navegador, do ambiente e do computador. Ex: Dell, busca de drivers, versão do Windows, versão do navegador.
  - API Web Áudio fornece meios simples para manipular áudio:
    - Pegar uma faixa de áudio, alterar o volume dela, aplicar efeitos, etc;
    - Mas na verdade o navegador utiliza códigos complexos de baixo nível (ex: C++) para realizar o processamento de áudio de fato, deixando a complexidade abstraída pela API;
- **APIs de terceiros:**
  - Necessário recuperar seu código e suas informações de outro local da web;
  - A API do Pagar.me oferece acesso a funcionalidades para o processamento de pagamentos online: Aprovação de cartão de crédito, envio de boleto, etc.

# Template String

- Introduz uma nova forma de se trabalhar com strings;
- Representado pelo operador ``;
- Diferente das aspas simples e duplas usuais;

```
const str1 = `teste`
console.log(str1)
```

# Template String - Multilinhas

Não é mais necessário incluir \n para indicar quebra de linha;

```
const strMultiLinha = 'linha1 \n
linha2'
console.log(strMultiLinha)

const strMultiLinha = `linha1
linha do meio
linha2`
console.log(strMultiLinha)
```

# Template String – Interpolação de strings

- Misturar string com expressões ao invés de concatenar

```
const str = `Ola ${1 + 1} !`
console.log(str)
```

- Podem ser utilizadas variáveis também:

```
const a = 10
const str = `Ola ${a + 1} !`
console.log(str)
```

# Template String - Tag

- Utilização de Tags para receber strings e values.
- A strings vai ter olá e mundo e o value terá o 10.
- Ele está entre as strings, por isso o nome interpolação.
- Com isso podemos criar uma tag em HTML sem grandes problemas.

```
function tag(strings, ...values) {
  console.log(strings);
  console.log(values);
}
tag`Olá ${10} mundo!`
```

# Template String - RAW

- Podemos acessar somente a parte string envolvida na interpolação desta forma:

```
console.log(strings.raw)
```

- E acessar as partes separadamente desta forma:

```
console.log(strings.raw[0])
```

# Funções Anônimas

- Uma função anônima é uma função sem nome.
- Pode ser chamada imediatamente após a declaração:

```
(function(){
        console.log(`Chamada imediatamente`);
})();
```

- Pode ser passado parâmetro:

```
let pessoa = {
        primeiroNome: "Marcelo",
        segundoNome: "Collares"
};

(function(){
console.log(`${pessoa.primeiroNome} ${pessoa.segundoNome}`);
})(pessoa);
```

# Funções Anônimas

- Pode ser usada como argumentos de outras funções.

```
setTimeout(function(){
        console.log(`Executar a cada 1 segundo`);
}, 1000);
```

- Geralmente não está acessível após a sua criação.
- Mas se precisarmos chamá-la em outro momento, atribuímos a uma variável;

```
let anonima = function () {
        console.log(`Função anônima`);
}
```

# Funções Anônimas X Arrow Function

Segue outra função anônima:

```
let add = function (a, b) {
        return a + b;
}
```

- O código da função acima pode ser reduzido usando a arrow function abaixo:

```
let add = () => a + b;
```

# Arrays e Objetos - Spread

- Spread Sintax ou Spread Operator;
- Permite que um objeto iterável, como um array ou string, seja expandida em locais onde zero ou mais argumentos sejam esperados;
- Para chamadas de função:

```
minhaFuncao(...objetoIteravel);
```

- Para array literais:

```
[...objetoIteravel, 4, 5, 6]
```

- Desestruturação:

```
[a, b, ...objetoIteravel] = [1, 2, 3, 4, 5];
```

# Arrays e Objetos - Spread - Exemplos

Para chamadas de função:

```
function minhaFuncao(x, y, z) { }
var args = [0, 1, 2];
minhaFuncao(...args);
```

Para array literais:

```
var genericos = ['oi', 'ola'];
var comprimentos = ['beleza', ...genericos, 'legal', 'joia’];
```

Desestruturação:

```
var primeiro, segundo, demais;
var todos = [primeiro, segundo, ...demais];
```

# Arrays e Objetos - Desestruturação

- Usada para facilitar a extração de valores de objetos em JavaScript
- Usa-se a sintaxe: {Variáveis} = Objeto;

SEM DESESTRUTURAÇÃO

```
const pessoa = {
    nome : "Marcelo",
    profissao : "Professor",
    email : "mmcollares@gmail.com"
};

var nome = pessoa.nome;
var profissao = pessoa.profissao;
var email = pessoa.email;

console.log(nome);
console.log(profissao);
console.log(email);
```

COM DESESTRUTURAÇÃO

```
const pessoa = {
    nome : "Marcelo",
    profissao : "Professor",
    email : "mmcollares@gmail.com"
};

const {nome, profissao, email} = pessoa;

console.log(nome);
console.log(profissao);
console.log(email);
```

# Arrays e Objetos - Desestruturação

- Se eu quiser pegar apenas um valor?

```
const pessoa = {
    nome : "Marcelo",
    profissao : "Professor",
    email : "mmcollares@gmail.com"
};

const {email} = pessoa;

console.log(email);
```

# Arrays e Objetos - Desestruturação

- Se eu quiser atribuir a uma variável com outro nome?
- Usa-se a sintaxe: {campo_objeto : nome_escolhido} = Objeto;

```
const pessoa = {
    a : "Marcelo",
    b : "Professor",
    c : "mmcollares@gmail.com"
};

const {a:nome, b:profissao, c:email} = pessoa;

console.log(nome);
console.log(profissao);
console.log(email);
```

```
var nome = pessoa.a;
var profissao = pessoa.b;
var email = pessoa.c;
```

# Arrays e Objetos - Desestruturação

- Será que eu posso passar como parâmetro de uma função?

SEM DESESTRUTURAÇÃO

```
function imprimePessoa(pessoa) {

    console.log(pessoa.nome);
    console.log(pessoa.idade);
    console.log(pessoa.email);
}

imprimePessoa(pessoa);
```

COM DESESTRUTURAÇÃO

```
function imprimePessoa({nome, idade, email}) {

    console.log(nome);
    console.log(idade);
    console.log(email);
}

imprimePessoa(pessoa);
```

# Arrays e Objetos – Metodos Auxiliares - MAP

Mapeia um objeto conforme uma regra e alterar valores;

```
let pessoas = [
    {id: 1, nome: "Arthur", sobreNome: "Ranquine", alimento: "Churrasco" },
    {id: 2, nome: "Tatiane", sobreNome: "Costa", alimento: "Escondidinho" },
    {id: 3, nome: "Cristopher", sobreNome: "Costa", alimento: "Escalopinho" },
    {id: 4, nome: "Arthur", sobreNome: "Vinagre", alimento: "Macarronada" },
    {id: 5, nome: "Mariana", sobreNome: "Rodrigues", alimento: "Nhoque" }
    ];

pessoas.map( ( pessoa ) => {
    if(pessoa.sobreNome === "Costa"){
        return pessoa.alimento = "Churrasco";
    }
    return undefined;
});

console.log(pessoas);
```

# Arrays e Objetos – Metodos Auxiliares - FILTER

Filtra os dados de um objeto conforme uma regra;

```
let pessoas = [
    {id: 1, nome: "Arthur", sobreNome: "Ranquine", alimento: "Churrasco" },
    {id: 2, nome: "Tatiane", sobreNome: "Costa", alimento: "Escondidinho" },
    {id: 3, nome: "Cristopher", sobreNome: "Costa", alimento: "Escalopinho" },
    {id: 4, nome: "Arthur", sobreNome: "Vinagre", alimento: "Macarronada" },
    {id: 5, nome: "Mariana", sobreNome: "Rodrigues", alimento: "Nhoque" }
    ];

let filtro = pessoas.filter( ( pessoa ) => {
        return pessoa.alimento === "Churrasco";
        });

console.log(filtro);
```