

Vendo o mapa



Foto de Aadiya Arora:
<https://www.pexels.com/pt-br/foto/ilustracao-do-mapa-mundial-592753/>

Módulo 2 - Aula 9 - Hard



Todos os direitos reservados
©2022 Resilia Educação



Review

Revisão



Hoisting



Hoisting

Hoisting (içamento) é uma técnica utilizada pelo **interpretador JavaScript** para que todas as **declarações de funções e variáveis** sejam “**elevadas**” para o **começo do código**.
Por conta disso, **funções e variáveis** podem ser **utilizadas antes** das suas **declarações**.

Hoisting - exemplo

```
console.log(soma(2,2));
```

```
function soma (num1, num2) {  
    return num1 + num2;  
}
```

Hoisting - exemplo

```
let empresa = "Resilia";  
console.log(empresa + " " + curso);  
let curso = "Programadores Cariocas";
```



Closures



Closures

Closures são funções aninhadas. Ou seja, **funções criadas dentro de funções** que **lembram o escopo** em que elas foram **criadas**.



Closures - exemplo

```
function externa() {  
    function interna() {  
        //...  
    }  
    const result = interna();  
}
```



Ficou abstrato?



Closures - exemplo

```
function mae() {  
    this.nome = "Função mãe";  
    function filha() {  
        console.log("Rodando função filha");  
        console.log(`Nome do escopo da mãe: ${this.nome}`);  
    }  
    filha();  
}  
  
mae();  
try {  
    filha();  
} catch (error) {  
    console.log("Essa função não existe no escopo global!")  
}
```





Paradigmas de programação



A detailed close-up photograph of an engine's belt drive system. A black timing belt is stretched across several pulleys. On the left, a large pulley is marked with '1C1485H'. Above it, a smaller pulley is marked with '1CWHIC'. The background shows various engine components, hoses, and metal parts, all slightly out of focus.

Paradigma funcional



Programação funcional

O paradigma de programação funcional utiliza como recurso base funções puras. Nele, os procedimentos e operações de um programa são realizados através da composição de funções.

Neste paradigma são evitadas a mutabilidade dos dados e efeitos colaterais.

JavaScript - programação funcional

Apesar de não ser uma linguagem nativamente funcional, o padrão ES5 “pegou emprestado” alguns métodos muito comuns e utilizados em linguagens funcionais.



Arrays



A world map is displayed on a light brown wooden background. The map is dark grey, and numerous small blue dots are scattered across it, primarily concentrated in Europe, Asia, and Africa, representing data points. The text "Array map" is centered over the map in a bold, yellow font.

Array map



Array - map

A função **map** de array realiza um mapeamento/transformação de todos os **elementos do array**, utilizando para isso uma **callback** passada como **argumento**. O **retorno** dessa função é um **array** com os **elementos mapeados**.



Array - map: exemplo sem map

```
//Decrementa em 10 saldos que estejam abaixo de 0
const saldos = [120.54, 0, 5.29, -72, 13, -25.3];

const saldosDecrementados [];

for (let i=0; i<saldos.length; i++) {
  if (saldos[i] < 0) {
    saldosDecrementados .push(saldos[i]-10);
  } else {
    saldosDecrementados .push(saldos[i]);
  }
}
```

Array - map: exemplo com map

```
//Decrementa em 10 saldos que estejam abaixo de 0
const saldos = [120.54, 0, 5.29, -72, 13, -25.3];

const saldosDecrementados = saldos.map( saldo => saldo
< 0? saldo-10 : saldo );
```

Array - map: imutabilidade

Como o **map** é um **método funcional** ele também respeita o **princípio da imutabilidade**, ou seja, ao aplicarmos o map em um array, **um novo é retornado com as alterações**, mas o **original continua o mesmo**.

```
const valores = [100, 200, 300]
const comDesconto = valores.map(valor=>{
  return valor*0.9 // aplicando 10% de desconto
})
console.log(valores) // [100, 200, 300]
console.log(comDesconto) // [90, 180, 270]
```





ATIVIDADE

Mapeando



Atividade: mapeando

Utilizando o link fornecido no roteiro de aula, realize as seguintes transformações no array já declarado utilizando o método `map`:

- ⇒ Crie um novo array chamado `nomes`, contendo apenas os nomes das pessoas usuárias
- ⇒ Crie um novo array chamado `nomeAbrev`, contendo as abreviações (primeiras letras) do nome e sobrenome, separadas por “.”
- ⇒ Crie um novo array chamado `amantesGatos`, contendo os nomes abreviados das pessoas que tem gatos
- ⇒ Crie um novo array chamado `maiores`, contendo os nomes das pessoas que tem 18 anos ou mais
- ⇒ Caso necessário, utilize a documentação && Não se esqueça de testar seus códigos!



Parâmetros rest





Spread



A person with long dark hair in a braid, wearing a white long-sleeved shirt and glasses, is shown from the side, aiming a wooden bow. The bow is held with both hands, and an arrow is nocked. The background is an outdoor archery range with trees and a wooden target stand. The text "Arrow functions" is overlaid in the center in a bold yellow font.

Arrow functions



A vertical chain of metal links is centered in the frame, extending from the top to the bottom. The background is a soft, out-of-focus landscape with a light sky and distant trees. The text "Utilizando bind" is overlaid in the center in a bold, yellow font.

Utilizando bind



M.V.C.





Models



Todos os direitos reservados
©2022 Resilia Educação

Controllers



Views

A hand holds a camera lens in the foreground, with the lens's viewfinder showing a reflection of a lake and mountains. The background is a blurred landscape of a lake and mountains under a blue sky with clouds.

Views

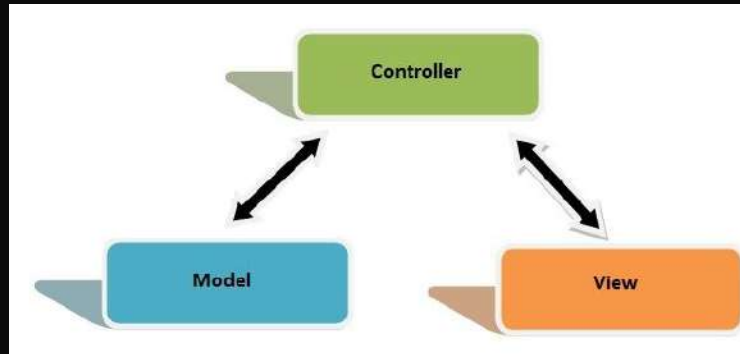
As **views** são responsáveis por **gerar a interface** que será exibida para as pessoas usuárias.

No caso de uma **aplicação web**, **views** produzem **HTML** a partir dos **dados fornecidos** para que o **controller** possa **modificar** o **estado** (interface) da página.

M.V.C. - funcionamento

A lógica e separação das responsabilidades na arquitetura MVC são:

- ⇒ **Model:** Abstração das entidades, validação dos dados e possível armazenamento;
- ⇒ **View:** Criação e manipulação dos elementos de interface referentes a entidade;
- ⇒ **Controller:** Orquestração e ponto entre models e views e controle do estado da aplicação.





MOMENTO ATIVIDADE EM PARES

