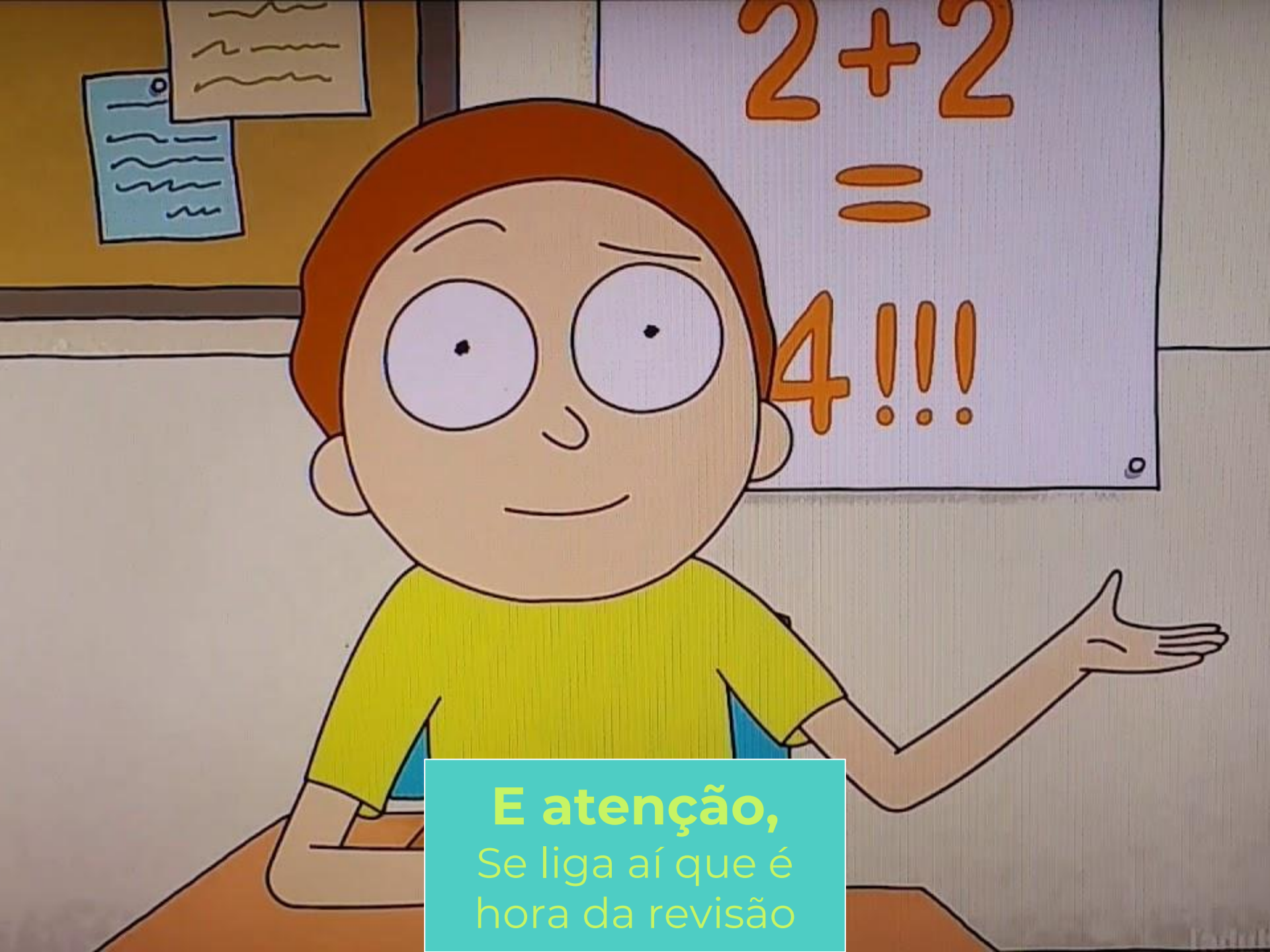


**Olá!**



**E atenção,**  
Se liga aí que é  
hora da revisão

## Lembrando...

1. Como utilizar JS: script interno e externo
2. Variáveis e tipos de dados
3. Operadores
  - a. Aritmeticos
  - b. Atribuição
  - c. Comparação
  - d. Lógicos
4. Estruturas condicionais
5. Laços
6. Funções

# JavaScript

## Aula 09

---

# 1. Objetos

Vamos  
aprender o  
conceito de  
objetos em  
JavaScript.

# Objetos

---

Objetos são containers de valores e ações, mas, em outras palavras, objetos são como carros!



# Objetos



Carros têm as mesmas ações (liga, desliga, acelera, freia), mas podem possuir características (nome, cor, modelo, peso, etc) diferentes. Pensando em carros como objetos, eles tem métodos (comuns a todos) e propriedades (variando de carro para carro)

| Properties                      | Methods                  |
|---------------------------------|--------------------------|
| <code>car.name = Fiat</code>    | <code>car.start()</code> |
| <code>car.model = 500</code>    | <code>car.drive()</code> |
| <code>car.weight = 850kg</code> | <code>car.brake()</code> |
| <code>car.color = white</code>  | <code>car.stop()</code>  |

# Objetos



Objetos são variáveis que podem conter diversos valores, escritos como um conjunto de **chave : valor**

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```



# Declarando Objetos

---

Existem duas formas de declarar os objetos:

```
var person = {  
    firstName: "John",  
    lastName: "Doe",  
    age: 50,  
    eyeColor: "blue"  
};
```

```
var person = new Object();  
person.firstName = "John";  
person.lastName = "Doe";  
person.age = 50;  
person.eyeColor = "blue";
```

# Objetos - Propriedades



Propriedades podem receber quaisquer tipos de valores, inclusive funções (métodos) do objeto!

| Property  | Value  |
|-----------|--|
| firstName | John   |
| lastName  | Doe  |
| age       | 50   |
| eyeColor  | blue   |
| fullName  | <code>function() {return this.firstName + " " + this.lastName;}</code> |

\_\_\_\_\_

```
objectName.property           // person.age

objectName["property"]        // person["age"]

objectName[expression]        // x = "age"; person[x]
```

# Acessando propriedades

---

O for / in também pode ser utilizado para acessar cada propriedade do objeto da seguinte maneira:

```
var txt = "";
var person = {fname:"John", lname:"Doe", age:25};
var x;
for (x in person) {
    txt += person[x] + " ";
}
```

txt ao final  
da execução: **John Doe 25**

# Adicionando / deletando propiedades

---

```
person.nationality = "English";
```

```
delete person.age;    // or delete person["age"];
```

# Objetos - Métodos



Métodos são funções pertencentes aos objetos, atribuídos a uma propriedade.

# Acessando métodos



Métodos podem ser acessados da seguinte forma:

*objectName.methodName ()* \*

\* Caso não se coloque o () no final, ao invés de executar o método a definição do mesmo (função) será retornada.

# Adicionando / deletando métodos

---

```
person.name = function () {  
    return this.firstName + " " + this.lastName;  
};
```

```
delete person.name;
```



# Objetos - Construtores



Para facilitar na criação e organização de objetos, construtores podem ser definidos.

```
function Person(first, last, age, eye) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eye;  
}  
  
var myFather = new Person("John", "Doe", 50, "blue");  
var myMother = new Person("Sally", "Rally", 48, "green");
```

# Objetos - this



Quando utilizamos "this", queremos dizer que estamos acessando uma propriedade pertencente ao objeto.

É a mesma ideia de utilizar "Object.property", com a diferença que estamos trabalhando propriedades do objeto dentro do objeto, e por isso não usamos uma variável instanciada com ele, mas sim o "this".



# Exercício 1

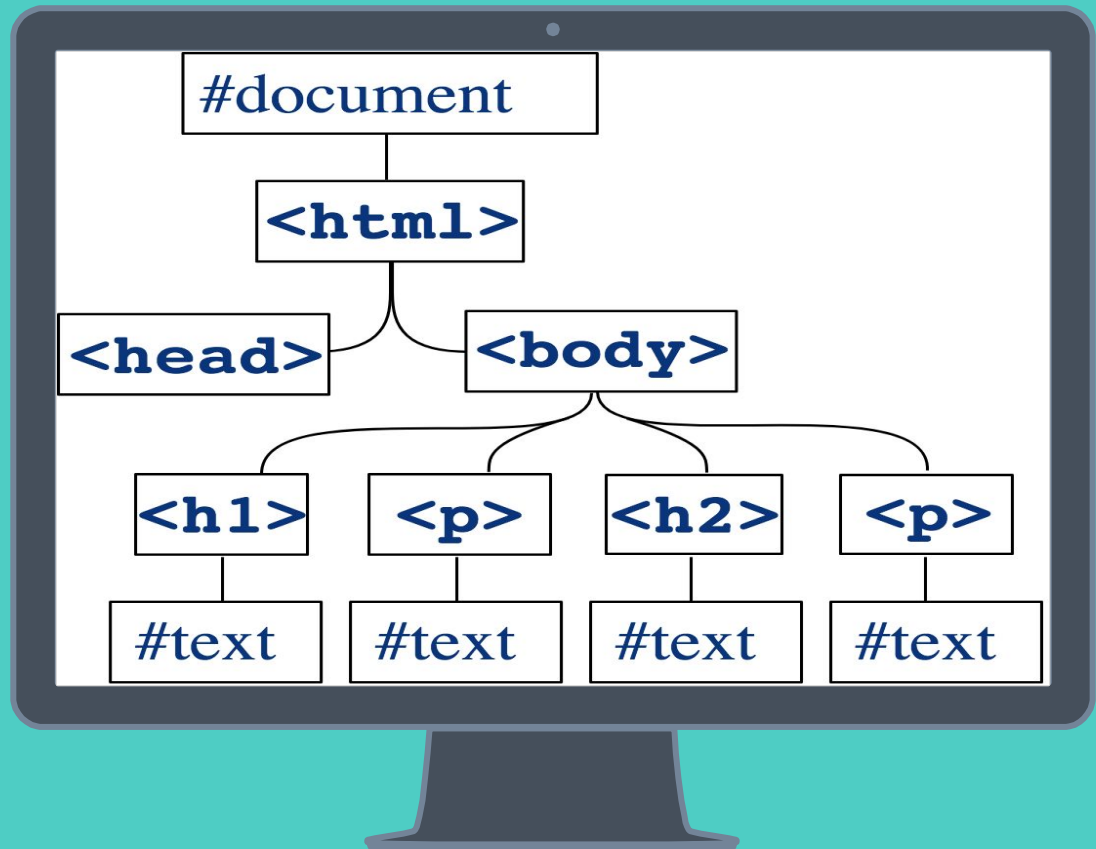
1. Crie um objeto chamado "boleto", que deve possuir as propriedades "valor", "diaVencimento" e "mesVencimento".
2. Crie um segundo objeto chamado "pessoa", que possui as propriedades "nome" e "valorDisponivel". Além disso, esse objeto deve possuir um método "pagarBoleto", que recebe como parâmetro um "boleto", e altera o "valorDisponível" da pessoa para o restante que ela possuirá após pagar o boleto. Além disso, deverá retornar a mensagem "Boleto de valor <valor do boleto> foi pago por <nome da pessoa> e agora restam <valor restante da pessoa>. #TaPago"\*.

\* <valor do boleto>, <nome da pessoa> e <valor restante da pessoa> devem ser substituídos pelos valores dos objetos referidos.

# 2.

## DOM

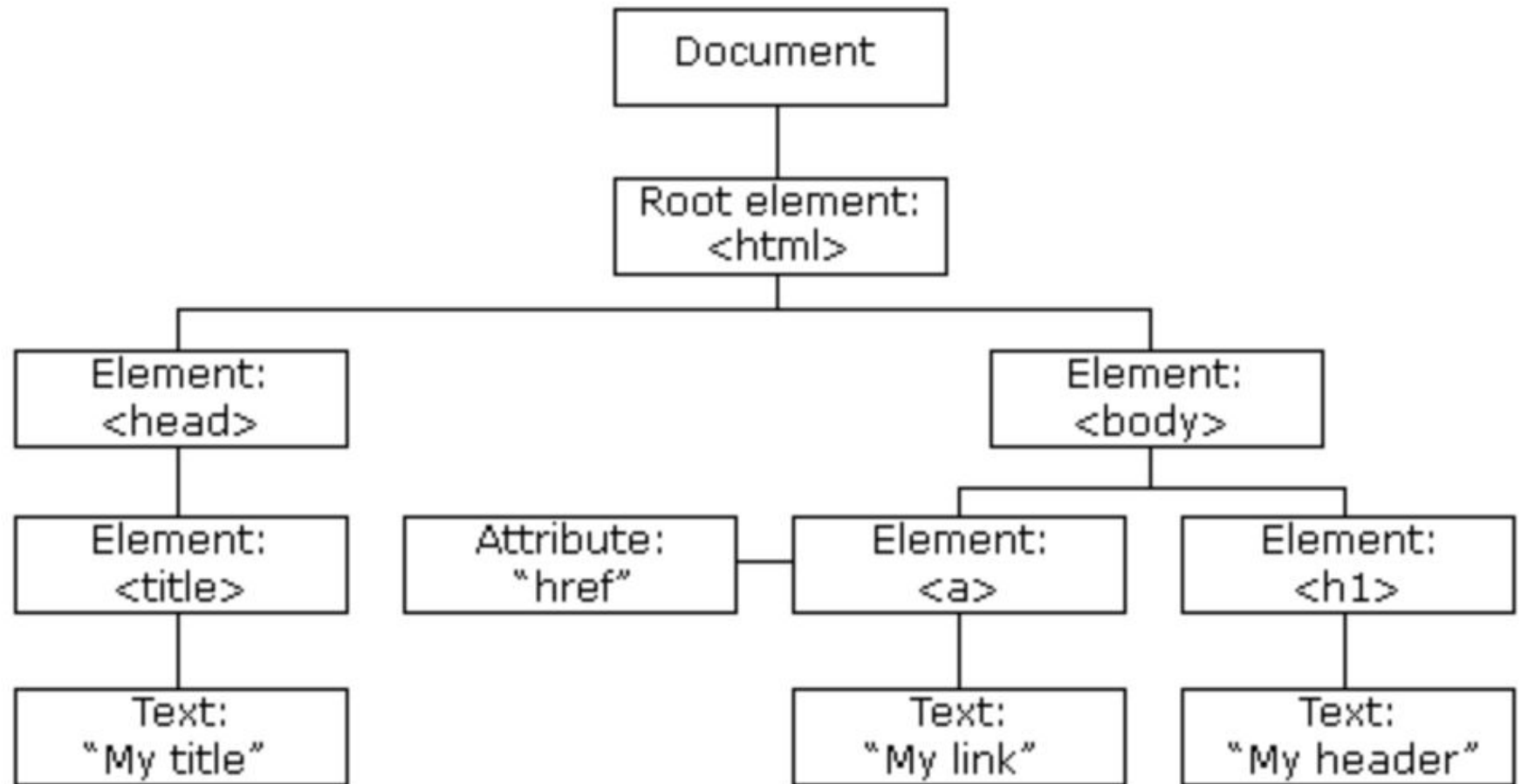
Vamos aprender sobre Document Object Model e como usar isto com JS.



## O que é DOM?

Document Object Model é um modelo criado assim que cada página do browser é aberta. Possui uma árvore de objetos padronizada e acessíveis via JavaScript.

# The HTML DOM Tree of Objects



# DOM - Interface de programação



No DOM, todos os elementos são considerados objetos. Logo, todos eles possuem métodos e atributos.



# Métodos



Para iniciarmos a nossa junção de JavaScript com HTML, vamos introduzir alguns métodos bastante utilizados.

- **getElementById()**: retorna o objeto do DOM que possui o id passado por parâmetro.
- **innerHTML**: retorna o conteúdo de um objeto HTML.

```
document.getElementById("demo").innerHTML = "Hello World!";
```

# Métodos



## Buscando elementos no DOM:

| Method  | Description                   |
|---|-------------------------------|
| <code>document.getElementById(<i>id</i>)</code>           | Find an element by element id |
| <code>document.getElementsByTagName(<i>name</i>)</code>   | Find elements by tag name     |
| <code>document.getElementsByClassName(<i>name</i>)</code> | Find elements by class name   |

## Alterando elementos do DOM:

| Method   | Description                                   |
|--|---|
| <code><i>element</i>.innerHTML = <i>new html content</i></code>          | Change the inner HTML of an element           |
| <code><i>element</i>.attribute = <i>new value</i></code>                 | Change the attribute value of an HTML element |
| <code><i>element</i>.setAttribute(<i>attribute</i>, <i>value</i>)</code> | Change the attribute value of an HTML element |
| <code><i>element</i>.style.<i>property</i> = <i>new style</i></code>     | Change the style of an HTML element           |

# Métodos



Adicionando / deletando elementos do DOM:

| Method  | Description                       |
|---|-----------------------------------|
| <code>document.createElement(<i>element</i>)</code> | Create an HTML element            |
| <code>document.removeChild(<i>element</i>)</code>   | Remove an HTML element            |
| <code>document.appendChild(<i>element</i>)</code>   | Add an HTML element               |
| <code>document.replaceChild(<i>element</i>)</code>  | Replace an HTML element           |
| <code>document.write(<i>text</i>)</code>            | Write into the HTML output stream |

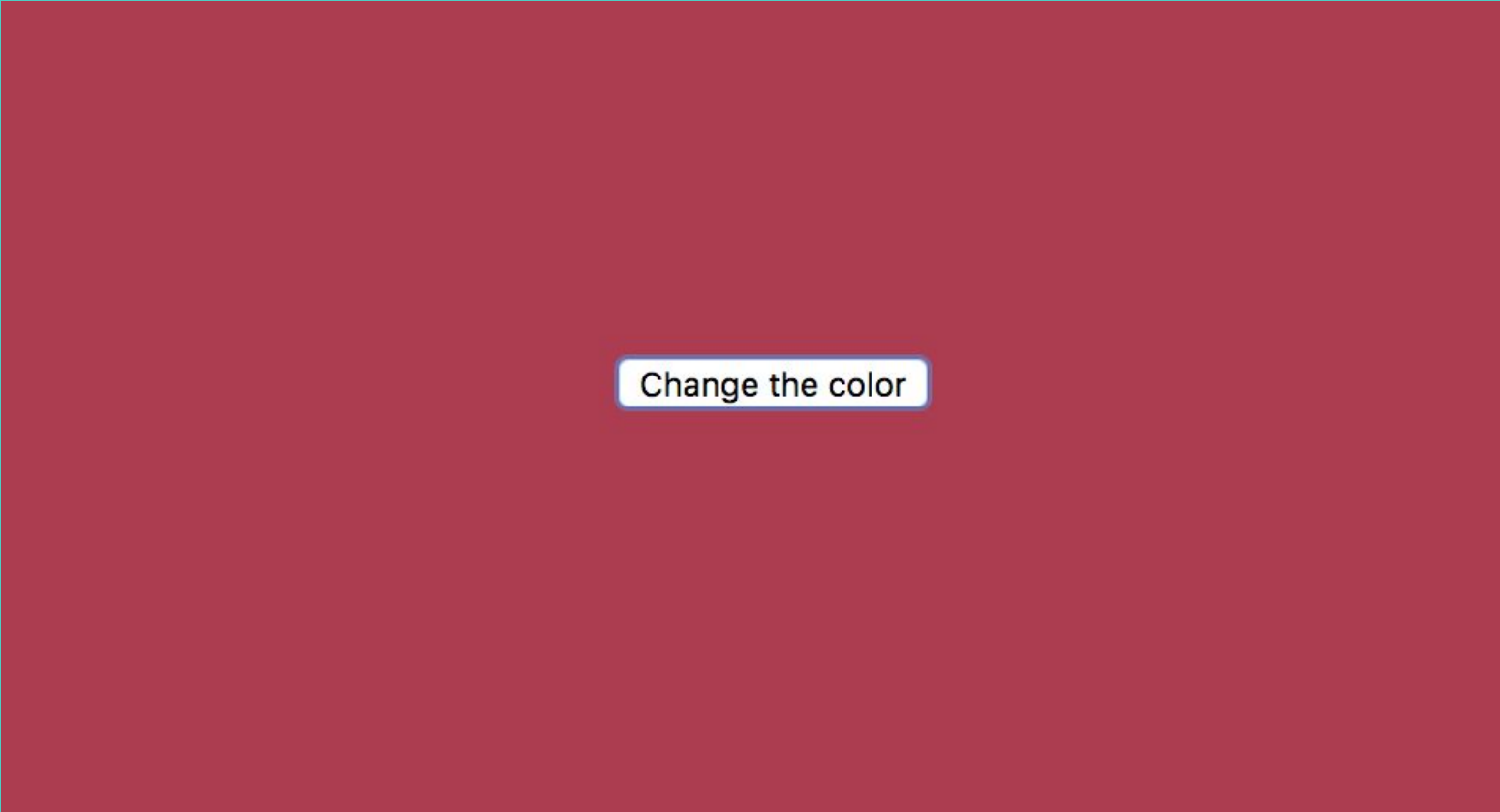
Adicionando *handlers* para eventos:

| Method  | Description                                   |
|---|---|
| <code>document.getElementById(<i>id</i>).onclick = function(){<i>code</i>}</code> | Adding event handler code to an onclick event |



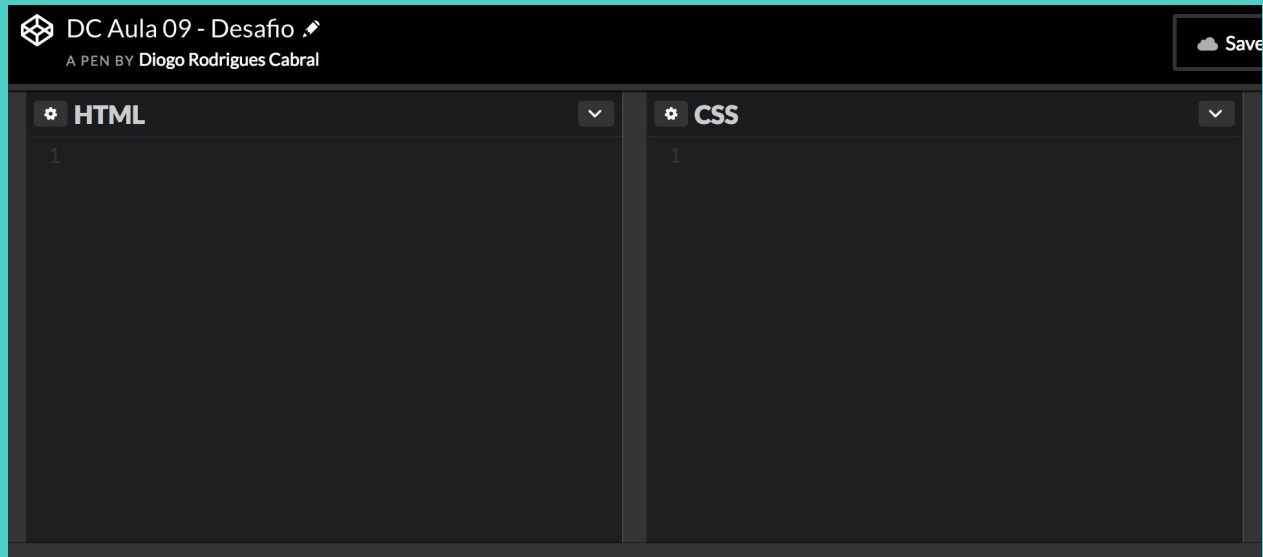
Desafio

1. Utilizando o CodePen, crie uma página que conterá:
  - a. Um botão centralizado
  - b. A cada vez que esse botão for clicado, a cor de fundo deverá mudar



Change the color

1. Porém... NÃO PODEM UTILIZAR CÓDIGO HTML OU CSS! É UM EXERCÍCIO APENAS DE JAVASCRIPT!



## 1. Para facilitar:

- a. As cores randômicas de background podem ser calculadas com `rgb`.
  - i. Lembrem-se que o JavaScript aceita valores textuais, neste caso.
  - ii. Procurem por `Math.random` e `Math.floor` para gerar valores randômicos num intervalo de valores.





Dúvidas?





[github.com/drcabral](https://github.com/drcabral)



[twitter.com/DrCabrales](https://twitter.com/DrCabrales)



[diogo.cabral.dev@gmail.com](mailto:diogo.cabral.dev@gmail.com)