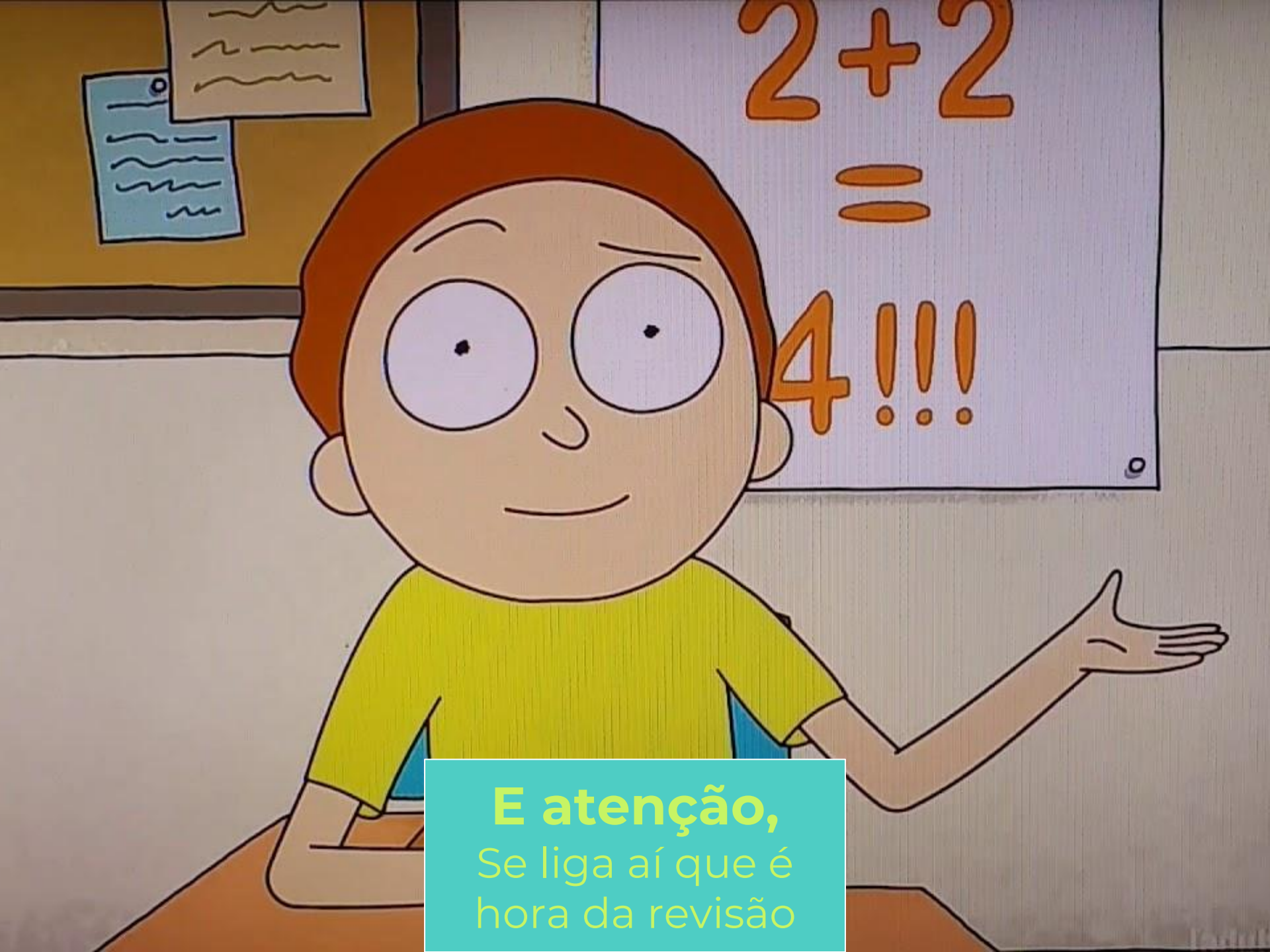
Olá!


2+2
=
4!!!
E atenção,
Se liga aí que é
hora da revisão

Lembrando...

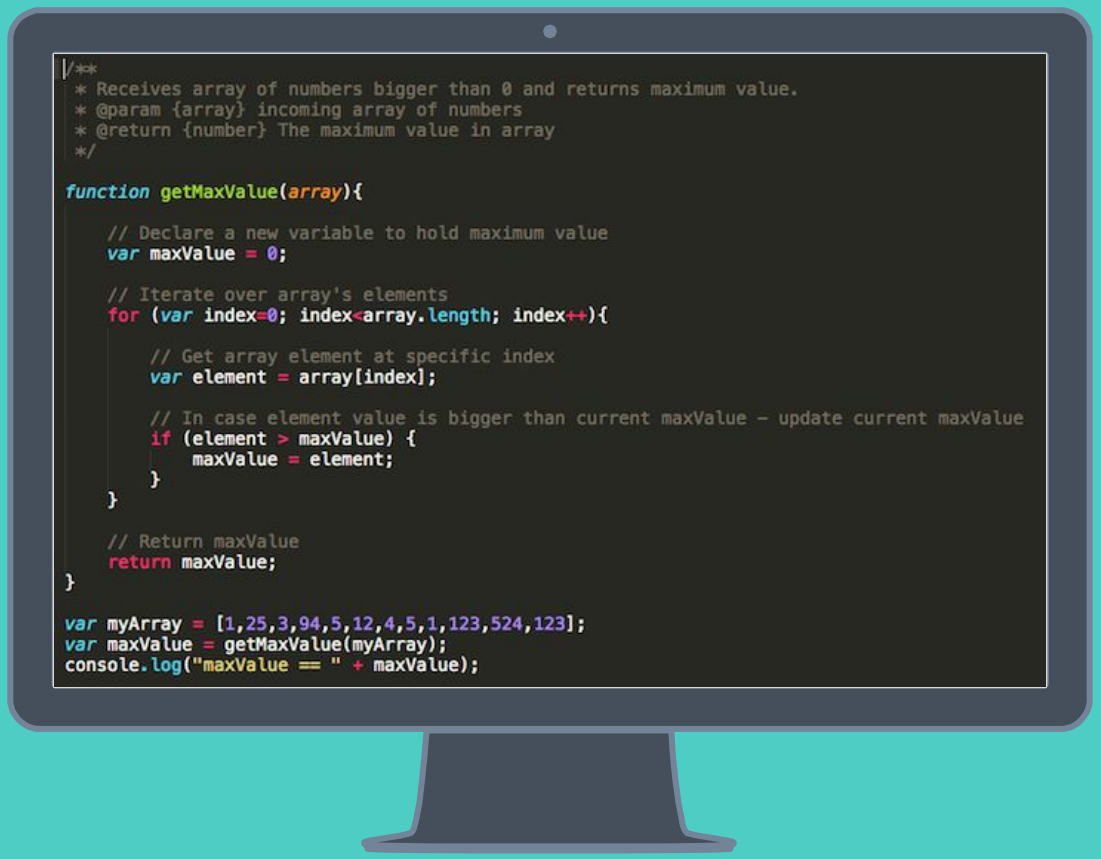
1. Gradientes
   a. Linear
   b. Radial
2. Transições
3. Transformações 2D e 3D
4. Animações

JavaScript
Aula 08

1.

Definições

Começaremo a entender o que é e pra que serve o JavaScript.



O que é JavaScript?

Linguagem de programação client-side com poder de controlar o HTML e CSS.

Lição 1

JavaScript não é Java!

Como usar JavaScript?

Assim como o CSS, existem diferentes formas de escrever o JavaScript para relacioná-lo a uma página HTML:

- Script interno
- Script externo

Script Interno

Scripts internos são adicionados a uma tag <script>, onde o type JavaScript é passado. O código JS estará contido dentro dessa tag e poderá ser referenciado no HTML.

```
<html>
  <head>
      <script language="javascript" type="text/javascript">
          function clickAlert(){
              alert("Clicado");
          };
      </script>
  </head>
  <body>
      <button onclick="clickAlert();">click</button>
  </body>
</html>
```

Script Externo

Scripts externos são mais comumente utilizados. São criados arquivos externos ao HTML, com extensão ".js", que são referenciados no HTML utilizando o <link>, assim como o CSS.

```
<html>
    <head>
        <script type="text/javascript" src="script.js" ></script>
    </head>
    <body>
    </body>
</html>
```

Estrutura

Blocos JavaScript são delimitados por chaves e, ao final de cada comando a ser interpretado, é recomendado o uso de ponto e vírgula.

```
function myFunction(p1, p2) {
    return p1 * p2;
}
```

Executando JavaScript para testes

Para testar noções básicas de JavaScript (sem integração com HTML ainda) podemos utilizar os consoles do browser, que aceitam execução desta linguagem.

Para facilitar, existem compiladores online de Javascript que nos permitem visualizar a execução direta do script.

https://jsconsole.com/

2.

Variáveis e tipos de dados

Iremos introduzir conceitos de variáveis e seus tipos em JS.

Variáveis

Variáveis são os armazenadores de dados, de diferentes tipos no JavaScript.

Eles precisam ser declarados através da palavra reservada "var" e possuem um identificador.

```
var x = 5;
var y = 6;
```

Tipos de dados

Variáveis JavaScript podem receber diferentes tipos de dados, que são categorizados em primitivos e não-primitivos. São eles:

- Primitivos
  - String
  - Boolean
  - Number
  - Undefined
  - Null
- Não-Primitivos
  - Objeto
  - Array
  - RegExp

Tipos de dados

```
// TIPOS PRIMITIVOS
var stringExample = "Olá";
var numberExample = 6;
var booleanExample = true;
var undefinedExample;
var nullExample = null;

//TIPOS NÃO-PRIMITIVOS
var objectExample = {type:"Fiat", model:"500", color:"white"};
var arrayExample = [1,2,3,4];
var regExpExample = /(\d+)\.\d*/;
```

3.

Operadores

Iremos aprender a realizar operações com diferentes tipos em JS.

Operadores

JS possui diversas possibilidades no tocante a operar com suas variáveis.

Os mesmos operadores podem indicar diferentes operações, a depender do tipo do dado.

Operadores aritméticos

As definições de operadores aritméticos são usados apenas em variáveis do tipo number. São eles:

- + (soma)
- - (subtração)
- * (multiplicação)
- / (divisão)
- % (módulo)
- ++ (incremento)
- -- (decremento)

Operadores de atribuição

São operadores que vão atribuir valores às variáveis, de algumas diferentes formas. São eles:

- = (atribuição)
- += (some e atribuição)
- -= (subtração e atribuição)
- *= (multiplicação e atribuição)
- /= (divisão e atribuição)
- %= (módulo e atribuição)

Operadores de strings

Com o tipo de dados string, conseguimos fazer concatenação do mesmo, com diferentes variáveis:

```
var txt1 = "Design";
var txt2 = "Culture";
var txtFinal = txt1 + " " + txt2;
```

Operadores de comparação

Conseguimos com JS fazer comparação entre variáveis. Essa comparação tem dois níveis: a nível de valor e a nível de tipo.

- == (igualdade de valor)
- === (igualdade de valor e de tipo)
- != (diferença de valor)
- !== (diferença de valor ou tipo)
- > (maior que)
- < (menor que)
- >= (maior ou igual que)
- <= (menor ou igual que)
- ? (operador ternário)

Operadores de comparação

Conseguimos com JS fazer comparação entre variáveis. Essa comparação tem dois níveis: a nível de valor e a nível de tipo.

- == (igualdade de valor)
- === (igualdade de valor e de tipo)
- != (diferença de valor)
- !== (diferença de valor ou tipo)
- > (maior que)
- < (menor que)
- >= (maior ou igual que)
- <= (menor ou igual que)
- ? (operador ternário)

Operadores de lógicos

Operadores lógicos nos permite compor e aprimorar diferentes comparações.

- && ("e" lógico)
- || ("ou" lógico)
- ! ("negação" lógica)

4.

Estruturas condicionais e laços

Iremos aprender a realizar operações com diferentes tipos em JS.

Estruturas condicionais

O JS consegue construir seus blocos de forma mais complexa, agregando seus valores e operadores. Além disso, fluxos condicionais também podem ser escritos para que o código mude de comportamento a depender da condição ser satisfeita ou não.

- If - else if - else
- Switch - case - default

If - else if - else

São blocos condicionais, ao qual você pode aplicar diversas condições e uma determinada lógica padrão caso nenhuma das condições anteriores seja validada

```
function myFunction() {
    var greeting;
    var time = new Date().getHours();
    if (time < 10) {
        greeting = "Good morning";
    } else if (time < 20) {
        greeting = "Good day";
    } else {
        greeting = "Good evening";
    }
document.getElementById("demo").innerHTML = greeting;
}
```

Switch - case - default

Também são blocos condicionais, ao qual você pode aplicar diversas condições (em uma estrutura de código diferente do if-else) uma determinada lógica padrão caso nenhuma das condições anteriores seja validada

```
var text;
switch (new Date().getDay()) {
    case 6:
        text = "Today is Saturday";
        break;
    case 0:
        text = "Today is Sunday";
        break;
    default:
        text = "Looking forward to the Weekend";
}
document.getElementById("demo").innerHTML = text;
```

Laços

A partir de um conjunto de dados (como um array, por exemplo), podemos executar um bloco de código que vai percorrer todos os valores desse conjunto, e executar alguma operação. Podemos fazer isso com os laços, e em JS existem os seguintes tipos:

- For
- For / in
- While
- Do / while

For

O laço "for" irá executar a ação um número determinado de vezes, podendo passar por cada elemento da lista. É comumente utilizado o tamanho total da lista como condição de parada do laço (mas não é uma regra que se faça desta forma).

```
var text = "";
var i;
for (i = 0; i < 5; i++) {
    text += "The number is " + i + "<br>";
}
```

For

```
var cars = ["BMW", "Volvo", "Saab", "Ford"];
var text = "";
for (var i = 0; i <= cars.length -1; i++) {
    text += cars[i] + " ";
}
```

For / in

O laço "for / in" irá executar a ação um número determinado de vezes, passando por cada elemento da lista. Sempre é utilizado percorrendo elementos da lista.

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

While

O laço "while" irá executar a ação enquanto a condição for verdadeira. Uma vez que a condição passa a ser falsa, o fluxo de execução sairá do laço.

```
var text = "";
var i = 0;
while (i < 10) {
    text += "The number is " + i + ";";
    i++;
}
```

Do / While

O laço "do / while" irá executar a ação ao menos uma vez, e, as demais, enquanto a condição for verdadeira. Uma vez que a condição passa a ser falsa, o fluxo de execução sairá do laço.

```
var text = ""
var i = 10;

do {
    text += "The number is " + i + ";";
    i++;
}
while (i < 10);
```

Exercício 1

1. Usando um editor de texto, crie um script que deverá ter implementado a seguinte lógica:
   a. Crie uma variável que receberá uma lista de números de 1 a 10.
   b. Percorra a lista de números e verifique, para cada número, se eles são divisíveis por 2
   c. Se for divisível por 2, concatene o valor de tal número, múltiplicado por 5, a uma variável de texto
   d. Tal variável de texto deverá conter a resposta de todos os elementos da lista que satisfaçam a condição durante a execução do código, separados por vírgula.
   e. A vírgula só deve aparecer entre valores

```
"10, 20, 30, 40, 50"
```

4.

Funções

Iremos aprender a utilizar funções em JS.

Funções

Funções são blocos JS que armazenam código em seu identificador, que podem ser chamados quando necessário, aceitando ou não parâmetros.

- Funções anônimas
- Funções declarativas

Funções declarativas

São funções que possuem um identificador próprio, declarado após a palavra reservada "function".

```
function myFunction(a, b) {
    return a * b;
}
```

Funções anônimas

São funções que não possuem identificador próprio, mas que podem ser atribuídas a variáveis comuns.

```
var x = function (a, b) {
    return a * b
};
```

Exercício 2

1. Crie uma função que receberá uma lista de números de 1 a 10 como parâmetro.
2. O retorno dessa função deverá ser a soma de todos os valores da lista.

Dúvidas?

github.com/drcabral

twitter.com/DrCabrales

diogo.cabral.dev@gmail.com