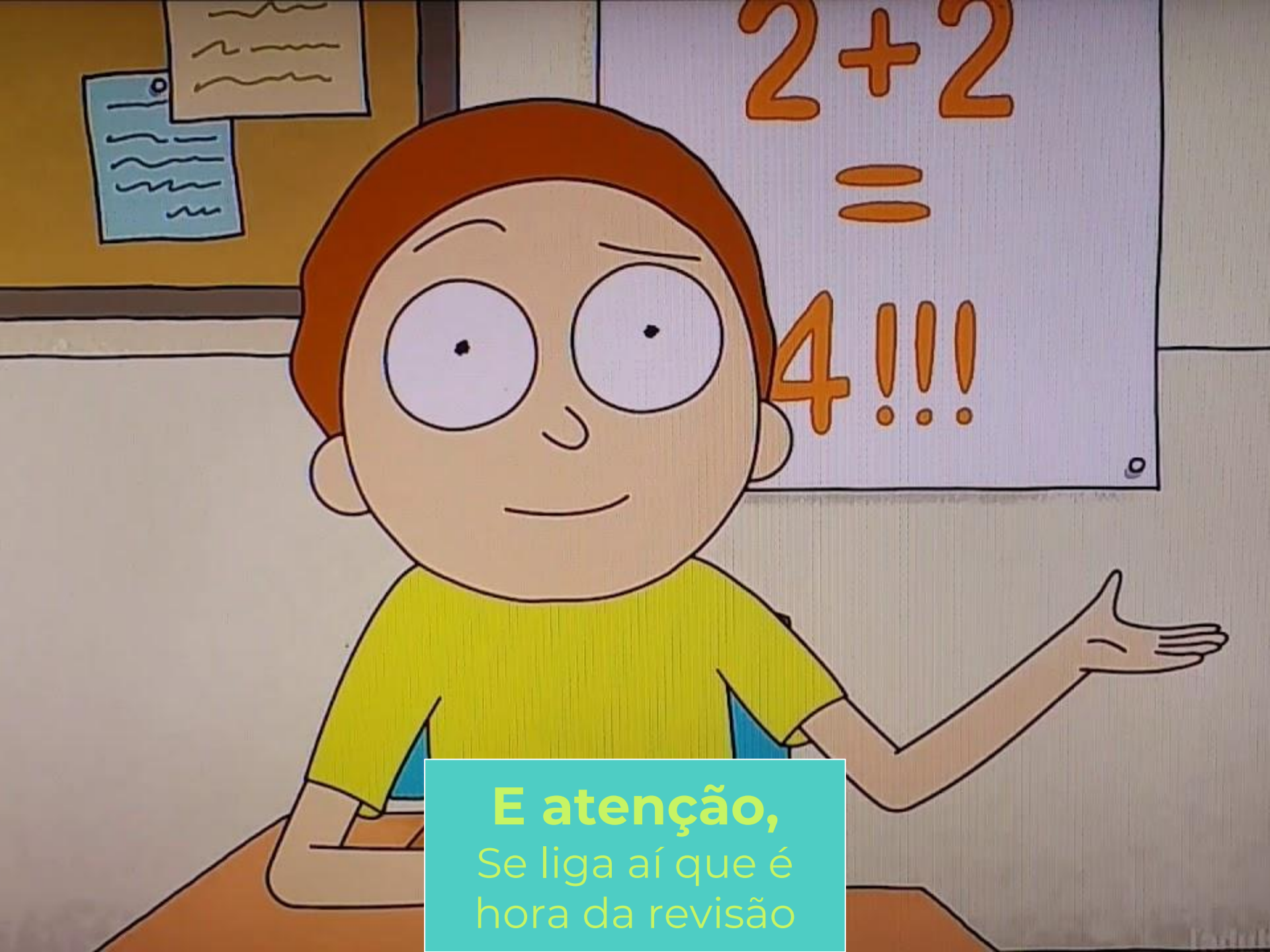


**Olá!**



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

# Lembrando...

1. Objetos
  - a. Propriedades, métodos
2. DOM
  - a. métodos

# JavaScript

## Aula 10

---

# 1. JSON

Vamos  
aprender a  
trabalhar com  
esse formato de  
transporte de  
dados.

# JSON



**J**ava**S**cript **O**bject **N**otation (JSON) é uma estrutura de dados baseada em objetos JavaScript, de fácil entendimento e amplamente utilizada como transporte de dados entre sistemas.

```
{
  "employees": [
    { "firstName": "John", "lastName": "Doe" },
    { "firstName": "Anna", "lastName": "Smith" },
    { "firstName": "Peter", "lastName": "Jones" }
  ]
}
```

# JSON



Pela semelhança de um documento JSON com JavaScript, tal conteúdo pode ser facilmente convertido em objetos JS.

## Regras de sintaxe:

1. Dados apresentados no formato chave:valor;
2. Dados são separados por vírgulas;
3. Chaves contém objetos em seu interior;
4. Colchetes contém arrays em seu interior.

No JSON, todas as chaves recebem aspas duplas (e não só os valores, como no JavaScript).

# JSON



Lembrem-se que um arquivo JSON nada mais é do que um arquivo texto estruturado. Como texto, pode ser lido pelo JavaScript e devidamente mapeado para objetos.

```
var text = '{"employees":[' +  
  '{"firstName":"John","lastName":"Doe" },' +  
  '{"firstName":"Anna","lastName":"Smith" },' +  
  '{"firstName":"Peter","lastName":"Jones" }]]';  
  
var obj = JSON.parse(text);
```



# JSON



API (Application Programming Interface) comumente utilizam JSON para responder a requisições do tipo GET ou coletar informações do tipo POST.

Exemplo: API de conversão de moedas (<http://fixer.io/>)

# Coletando e convertendo JSON

---

**A título de curiosidade**, é possível ler tal JSON a partir da API externa ao script e trabalhar ele como um objeto JavaScript.

```
var url = 'https://api.fixer.io/latest';
var jsonObj;

fetch(url)
  .then(res => res.json())
  .then((out) => {
    jsonObj = out;
  })
  .catch(err => { throw err });
```

# 2.

## Javascript API

Vamos  
aprender a usar  
algumas API do  
JS.

# JavaScript API



O JavaScript é uma linguagem rica não só pelo que nos possibilita fazer como também pela gama de métodos pré-definidos já existentes na linguagem.

Tais métodos definem a API JavaScript, que possuem diversas formas de manipulações de dados para os diferentes tipos da linguagem.

# Number - métodos



São métodos de tratamento para variáveis numéricas.

- **toString(numero)** - transforma o número passado por parâmetro em uma String.
- **numero.toFixed(casaDecimal)** - arredonda o número pela casa decimal informada, retornando como String.
- **numero.toPrecision(casaDecimal)** - retorna a String com o número de tamanho até a casa decimal informada.
- **Number(stringNumero)** - retorna um valor numérico convertido da String passada por parâmetro.
- **parseFloat(valor)** - retorna o valor passado por parâmetro convertido para um dado tipo float (ponto flutuante).
- **parseInt(valor)** - retorna um número convertido a partir da String.

# String - métodos



São métodos de tratamento para variáveis do tipo String.

- **texto.length** - retorna o tamanho da string (texto, no exemplo).
- **texto.indexOf(textoBusca)** - retorna o índice em que ocorre a primeira aparição do texto passado por parâmetro no texto ao qual a função é chamada.
- **texto.lastIndexOf(textoBusca)** - retorna o índice em que ocorre a última aparição do texto passado por parâmetro no texto ao qual a função é chamada.
- **texto.substring(inicio, fim)** - retorna o corte da string a partir do índice de início e fim definido por parâmetro.
- **texto.substr(inicio, tamanho)** - retorna o corte da string a partir do índice de início mais a quantidade de caracteres passada no tamanho.

# String - métodos



São métodos de tratamento para variáveis do tipo String.

- **texto.replace(stringBusca, stringTroca)** - troca na String ao qual o método é chamado a String de busca passada por parâmetro pela a String de troca.
- **texto.toUpperCase** - retorna a String maiúscula.
- **texto.toLowerCase** - retorna a String minúscula.
- **texto.concat(string1, string2, string3, ...)** - concatena Strings passadas por parâmetro na String ao qual o método está sendo chamado.
- **texto.split(delimitador)** - separa a string em um array de strings, usando o delimitador passado por parâmetro para fazer a separação.

# Date



É um tipo de objeto pré-definido do JavaScript que possui métodos específicos para manipulação de datas.

```
new Date()
```

```
new Date(milliseconds)
```

```
new Date(dateString)
```

```
new Date(year, month, day, hours, minutes, seconds, milliseconds)
```



# Date - formatos



O JavaScript aceita vários tipos de formatos de datas em sua criação, como por exemplo:

Type	Example
ISO Date	"2015-03-25" (The International Standard)
Short Date	"03/25/2015"
Long Date	"Mar 25 2015" or "25 Mar 2015"
Full Date	"Wednesday March 25 2015"

# Date - métodos

---

- **toString()** - método que converte a data em String.
- **toUTCString()** - retorna a String da data no formato UTC.
- **toDateString()** - retorna uma String de data mais legível.

# Date - métodos



Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

# Date - métodos



Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)



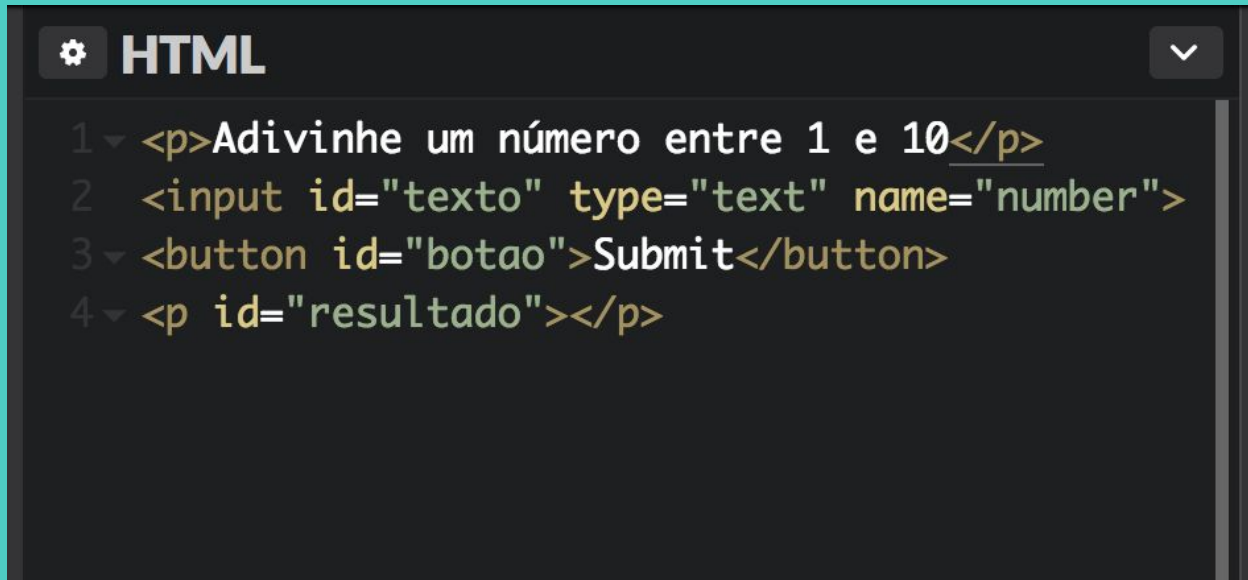
Desafio

1. Jogo da adivinhação:
  - a. Crie um número randômico de 1 à 10.
  - b. Faça um formulário HTML como o abaixo.

**Adivinhe um número entre 1 e 10**

- c. Ao clicar no botão, a seguinte lógica deve ser operada:
  - i. Se não houver texto, aponte o erro "Digite um valor." na cor vermelha.
  - ii. Se houver texto, mas não for número, aponte o erro "Digite um número entre 1 e 10." na cor vermelha
  - iii. Se for um texto numérico:
    1. Compare com o número randômico, se der erro, aponte a mensagem "Você errou, tente outro número!" na cor vermelha.
    2. Caso contrário, informe a mensagem "Acertou!" em verde.

1. Use a seguinte estrutura HTML

A screenshot of a code editor window with a dark background. The title bar at the top says "HTML" with a gear icon on the left and a dropdown arrow on the right. The code is written in a light-colored font and is numbered 1 through 4 on the left side of the editor. The code consists of four lines: a paragraph tag, an input tag, a button tag, and another paragraph tag.

```
1 <p>Adivinhe um número entre 1 e 10</p>  
2 <input id="texto" type="text" name="number">  
3 <button id="botao">Submit</button>  
4 <p id="resultado"></p>
```

2. Não use CSS.



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)