# Linguagem de Programação para Internet

LUIZ CARLOS FELIX CARVALHO

# Regras do Jogo

Primeiro momento (1 a 12/4)

- UNIUBE+ – 5 pontos
- Trabalho – 5 pontos
- Avaliação – 20 pontos

Segundo momento (20 a 29/5)

- UNIUBE+ – 5 pontos
- Trabalho – 5 pontos
- Avaliação – 20 pontos

Terceiro momento (19 a 25/6)

- UNIUBE+ – 5 pontos
- Trabalho – 5 pontos
- Avaliação – 20 pontos
- Simulado – 10 pontos
  - 18/6

- Segunda Chamada / Substitutiva / Recuperação
  - 24 a 28/6

# Regras do Jogo

Primeiro momento

- **Trabalho – 5 pontos: 12/4**
  - 2 Trabalhos avaliativos
  - 10 pontos: 22/3
  - **10 pontos: 12/4**

Segundo momento

- Trabalho – 10 pontos
- Avaliação – 10 pontos
- 1 Trabalho Avaliativo – 5 pontos

Terceiro momento

- Projeto – 20 pontos
- 1 Trabalho Avaliativo - 5 pontos

# Árvore DOM

DOCUMENT OBJECT MODEL

MODELO DE OBJETO DE DOCUMENTO

# DOM

É uma interface de programação para documentos HTML, XML e SVG.

Fornece uma representação estruturada do documento como uma árvore.

Define métodos que permitem acesso à árvore:

- Alterar a estrutura
- Alterar estilo
- Alterar conteúdo

Fornece uma representação do documento como um grupo estruturado de nós e objetos, possuindo **várias propriedades e métodos**.

Permite o HTML ser manipulado.

**Conecta páginas web a scripts**.

# DOM

Exemplo:

- Código JavaScript acessando o DOM

```
const paragraphs = document.querySelectorAll("p");
// paragraphs[0] is the first <p> element
// paragraphs[1] is the second <p> element, etc.
alert(paragraphs[0].nodeName);
```

# DOM

Todos os métodos, propriedades, eventos estão organizados em objetos.

O mais famoso:

- Document
- Representa o próprio documento

Outro exemplo:

- HTMLTableElement
- Table – Tabela no HTML

DOM não é uma linguagem de programação

É uma interface para acessar o documento HTML

O DOM não faz parte da linguagem JavaScript

Node.js não possui a DOM API

# DOM

Mais exemplo:

HTML

```
<body onload="console.log('Welcome to my home page!');">
  …
</body>
```

HTML

```
<html lang="en">
  <head>
    <script>
      // run this function when the document is loaded
      window.onload = () => {
        // create a couple of elements in an otherwise empty HTML page
        const heading = document.createElement("h1");
        const headingText = document.createTextNode("Big Head!");
        heading.appendChild(headingText);
        document.body.appendChild(heading);
      };
    </script>
  </head>
  <body></body>
</html>
```

# DOM

Exemplos de tipo de dados:

- Document
- Element
- NodeList

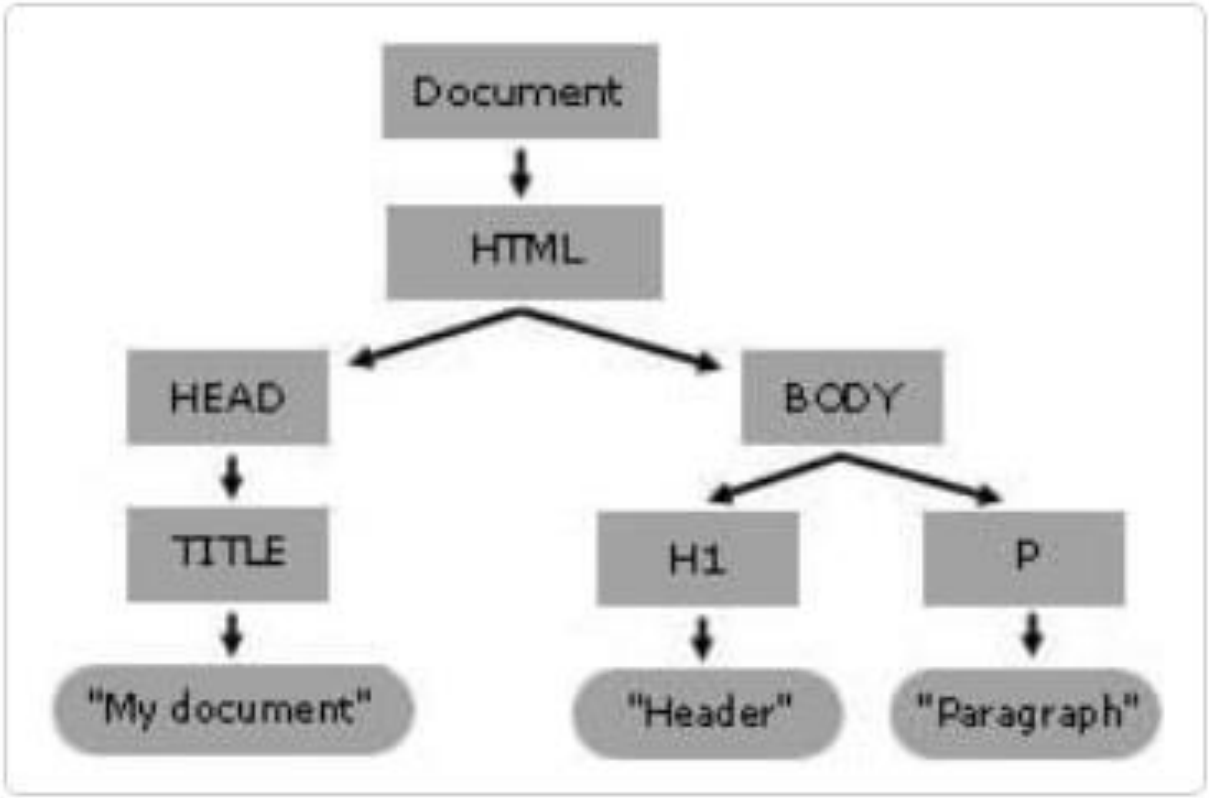
# DOM

Exemplos

# Árvore DOM

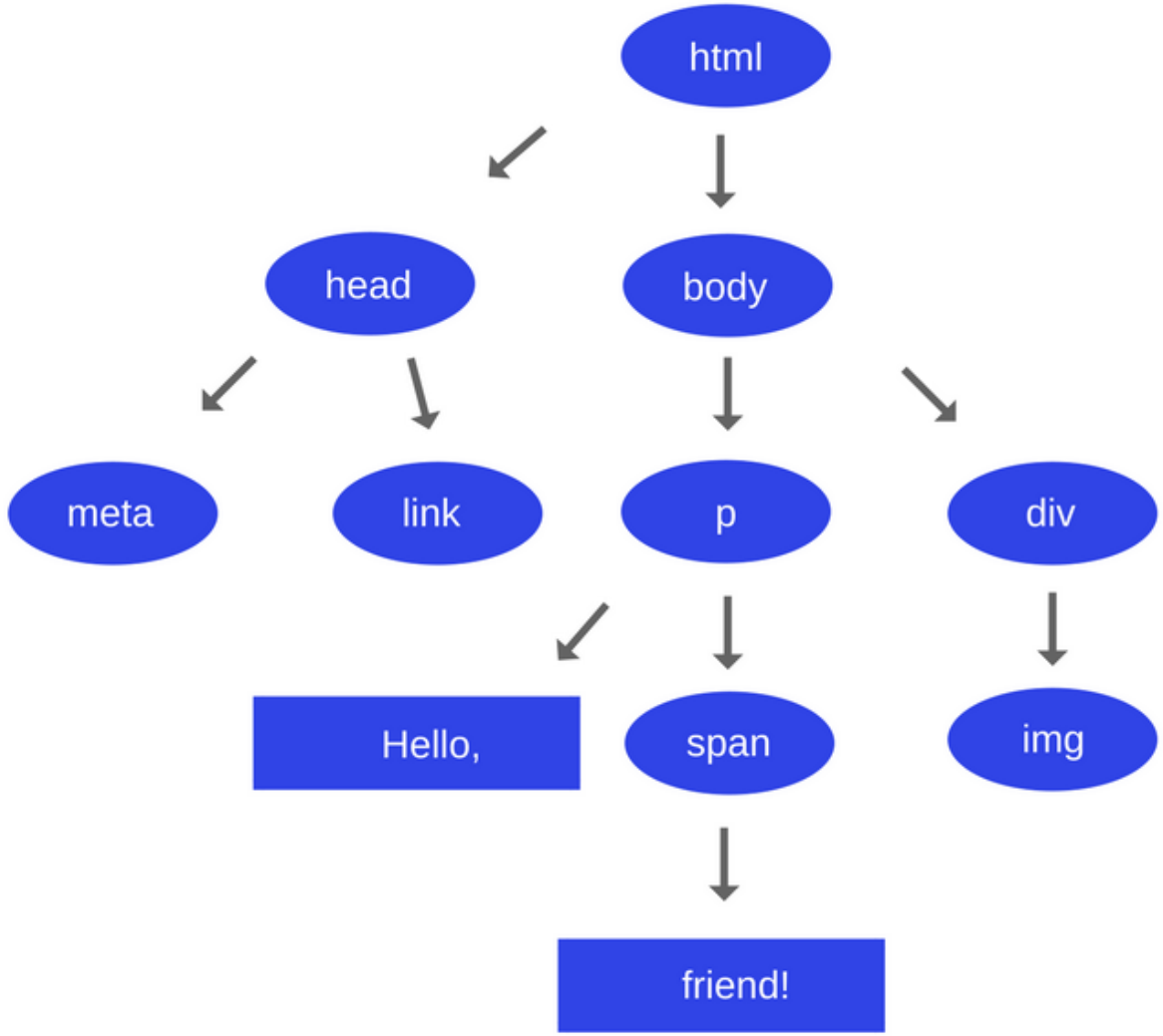
HTML

```html
<html lang="en">
  <head>
    <title>My Document</title>
  </head>
  <body>
    <h1>Header</h1>
    <p>Paragraph</p>
  </body>
</html>
```



# Árvore DOM

```
<html>
  <head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="theme.css">
  </head>
  <body>
    <p> Hello, <span> friend! </span> </p>
    <div>
      <img src="smiley.gif" alt="Smiley face" height="42" width="42">
    </div>
  </body>
</html>
```



# DOM

Interfaces do DOM:

- Attr
- CharacterData
- ChildNode Experimental
- Comment (en-US)
- CustomEvent (en-US)
- Document
- DocumentFragment
- DocumentType (en-US)
- DOMError (en-US)
- DOMException (en-US)
- DOMImplementation (en-US)
- DOMString
- DOMTimeStamp (en-US)
- DOMSettableTokenList
- DOMStringList
- DOMTokenList (en-US)
- Element
- Event
- EventTarget
- HTMLCollection
- MutationObserver
- MutationRecord (en-US)
- Node
- NodeFilter
- NodeIterator (en-US)
- NodeList
- ParentNode Experimental
- ProcessingInstruction
- Range (en-US)
- Text (en-US)
- TreeWalker (en-US)
- URL
- Window
- Worker
- XMLDocument