Programação Visual e Autoria Web

Document Object Model

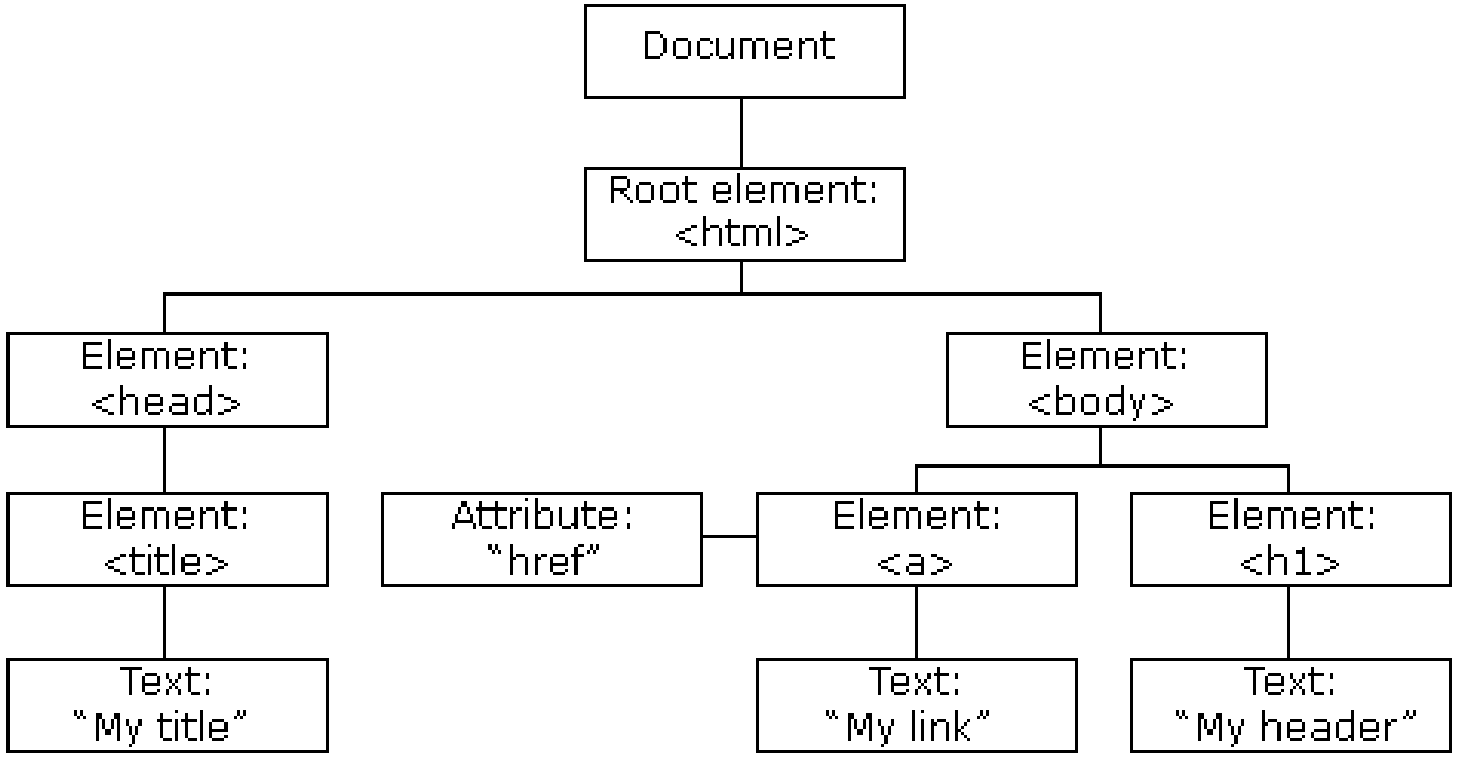
Universidade Federal do Rio Grande do Norte

Introdução

- Ao carregar uma página HTML os navegadores guardam em sua memória uma estrutura de dados que organiza todos os elementos do documento.
- Tal estrutura é conhecida como DOM, ou Document Object Model, e seu organização se dá sob uma estrutura de árvore onde a raiz é o elemento HTML.
- A partir do JavaScript podemos ter acesso ao DOM para manipula-lo e realizar operações de acesso e manipulação dos elementos
  - document

Introdução

```
<!DOCTYPE html>
<html>
    <head>
        <title>My title</title>
    </head>
    <body>
        <a href="#">My link</a>
        <h1>My header</h1>
    </body>
</html>
```



FONTE: https://www.w3schools.com/whatis/whatis_htmldom.asp

Introdução

- A partir do DOM, o JavaScript pode:
  - alterar elementos HTML na página
  - alterar atributos HTML da página
  - alterar estilos CSS na página
  - remover elementos e atributos HTML existentes
  - adicionar novos elementos e atributos HTML
  - reagir aos eventos HTML existentes na página
  - criar novos eventos HTML na página

Manipulando o DOM

- A maneira mais fácil de selecionar elementos do DOM é através do atributo id.
- Como esse atributo é único, podemos selecionar de maneira específica o elemento que queremos.
- Atenção para um detalhe: o browser vai executar o arquivo HTML linha por linha do topo até o fim. Isso significa que seu script poderá ser executado antes do browser inserir todos os elementos HTML no DOM.
  - Por enquanto, vamos colocar nossos scripts apenas ao final do documento HTML.

Manipulando o DOM

document.getElementById (Literal)

Retorna a primeira ocorrência de um elemento com ID dado.

```
<!DOCTYPE html>
<html>
    <head>
        <title id="titulo">My title</title>
    </head>
    <body>
    </body>
    <script src="script.js"></script>
</html>
```

```
let titulo = document.getElementById("titulo");
console.log(typeof titulo);
console.log(titulo.innerText);
```

Atividade

- Escreva o código JavaScript para verificar se um elemento de um formulário possui conteúdo com mais de 5 caracteres. Caso possua menos de 5 caracteres adicione uma borda vermelha ao elemento.
- Dicas:
  - Usar `onclick ()` no botão para invocar a função JavaScript
  - O conteúdo do `input text` pode ser obtido através da propriedade `value`.
  - É possível adicionar uma classe a um elemento do DOM utilizando a propriedade `className`

Atividade

```
<!DOCTYPE html>
<html>
    <head>
        <title id="titulo">My title</title>
        <style>
            .erro {
                border: 0.15rem solid red;
            }
        </style>
    </head>
    <body>
        <label for="nome"> Nome: </label><input type="text" id="nome“ />
        <button type="submit">Verificar</button>
    </body>
    <script src="script.js"></script>
</html>
```

Atividade

```html
<!DOCTYPE html>
<html>
    <head>
        <title id="titulo">My title</title>
        <style>
            .erro {
                border: 0.15rem solid red;
            }
        </style>
    </head>
    <body>
        <label for="nome"> Nome: </label><input type="text" id="nome" />
        <button type="submit" onclick="isFilled('nome')">Verificar</button>
    </body>
    <script src="script.js"></script>
</html>
```

```javascript
function isFilled(id) {
    let elemento = document.getElementById(id);
    if (elemento.value.length < 5) {
        elemento.className += " erro";
    } else {
        elemento.className = "";
    }
}
```

Manipulando o DOM

- É possível selecionar vários elementos de um mesmo tipo através da função getElementsByTagName.
- O retorno dessa função é uma coleção de elementos HTML (HTMLCollection)
- A interface HTMLCollection representa uma coleção genérica (objeto semelhante a um array) de elementos (na ordem do documento) e oferece métodos e propriedades para seleção na lista
  - HTMLCollection.length, retorna o número de itens na coleção
  - HTMLCollection.item(i), retorna o iésimo elemento da coleção

Iterando sobre uma coleção

```
let elementos = document.getElementsByTagName("p");


for (let i = 0; i < elementos.length; i++) {
    console.log(elementos.item(i).innerText);
}
```

Iterando sobre uma coleção

```
let elementos = document.getElementsByTagName("p");

for (let e of elementos) {
    console.log(e.innerText);
}

for (let i = 0; i < elementos.length; i++) {
    console.log(elementos.item(i).innerText);
}
```

Atividade

- Escreva um código JavaScript que conte quantos parágrafos existem no documento e quantos parágrafos existem dentro do elemento `article`. Coloque o resultado como conteúdo da `div#resultgeral` ou `div#resultarticle`.

Atividade

```
<!DOCTYPE html>
<html>
    <head>
        <title id="titulo">My title</title>
    </head>
    <body>
        <h1>Início</h1>
        <p>Texto 1</p>
        <p>Texto 2</p>
        <p>Texto 3</p>
        <article>
            <p>Texto 4</p>
        </article>
        <div id="resultgeral"></div>
        <div id="resultarticle"></div>
    </body>
    <script src="script.js"></script>
</html>
```

Atividade

```html
<!DOCTYPE html>
<html>
    <head>
        <title id="titulo">My title</title>
    </head>
    <body>
        <h1>Início</h1>
        <p>Texto 1</p>
        <p>Texto 2</p>
        <p>Texto 3</p>
        <article>
            <p>Texto 4</p>
        </article>
        <div id="resultgeral"></div>
        <div id="resultarticle"></div>
    </body>
    <script src="script.js"></script>
</html>
```

```js
let paragrafos = document.getElementsByTagName("p");
document.getElementById("resultgeral").innerHTML =
    "O número de paragrafos é " + paragrafos.length;

let article = document.getElementsByTagName("article");
let paragrafosArticle = article[0].getElementsByTagName("p");

document.getElementById("resultarticle").innerHTML =
    "O número de paragrafos no artigo é " + paragrafosArticle.length;
```

Manipulando do DOM

- Também é possível selecionar um conjunto de elementos através da sua classe utilizando o método `getElementsByClassName (className).`
- O conjunto de elementos obtidos será uma HTMLCollection, portanto, possui a propriedade lenght e o método item(i).

```
var x = document.getElementsByClassName("erro");
```

Manipulando o DOM

- Podemos selecionar elementos do DOM utilizando seletores CSS através do método `querySelectorAll(selector)`
- O conjunto de elementos obtidos será uma HTMLCollection, portanto, possui a propriedade lenght e o método item(i).

```
var y = document.querySelectorAll("article p");
```

HTML Collections

- Existem HTML Collections padrão que podem ser acessadas diretamente pelo DOM.
  - document.anchors
  - document.forms
  - document.images
  - document.links
  - document.scripts

Adicionando elementos ao DOM

- É possível criar elementos HTML dentro do JavaScript e adiciona-los ao DOM.
  - Função `document.createElement( elementName )`
- Depois de criado do elemento é necessário adicionar atributos.
  - Função elemento.setAttribute (‘nome_atributo’, ‘valor’).
- Para adicionar texto ao elemento utilizamos a função `document.createTextNode(‘texto');`
- Por fim, é necessário adicionar o elemento ao DOM
  - Selecionar o elemento container
  - Adicionar o novo elemento com a função appendChild (novo_elemento)

Adicionando elementos ao DOM

```
let novo_elemento = document.createElement("a");
novo_elemento.setAttribute("href", "http://www.eaj.ufrn.br");
novo_elemento.setAttribute("target", "_blank");

let texto_elemento = document.createTextNode("Link para o site");
novo_elemento.appendChild(texto_elemento);

let container = document.querySelector("article");
container.appendChild(novo_elemento);
```

Removendo elementos do DOM

- Podemos remover um elemento do DOM através do método removeChild.
- Note que precisamos indicar o `parent` para realizar a remoção do elemento.

```
let remove_elemento = document.querySelector("#botao");

let pai_elemento = document.querySelector("body");
pai_elemento.removeChild(remove_elemento);
```

Eventos

- Quando um usuário interage com a página HTML ele pode gerar um evento.
- Os eventos estão associados a elementos HTML e podem utilizados para disparar funções JavaScript.
- Podemos também utilizar o DOM para alterar o evento que está associado a um elemento HTML.

Eventos

- oninput: input tem seu valor modificado
- onclick: click com o mouse
- ondblclick: dois clicks com o mouse
- onmousemove: move o mouse
- onmousedown: pressiona o botão do mouse
- onmouseup: solta o botão do mouse
- onkeypress: pressiona e solta uma tecla

Eventos

- onkeydown: pressiona uma tecla
- onkeyup: solta uma tecla
- onblur: elemento perde foco
- onfocus: elemento ganha foco
- onchange: input, select ou textarea tem seu valor alterado
- onload: página é carregada
- onunload: página é fechada
- onsubmit: formulário enviado (validações)

Bibliografia Básica

- DUCKETT, J. **Javascript e Jquery**: desenvolvimento de interfaces web interativas. Rio de Janeiro: Alta Bokks, 2016
- FLANAGAN, D. **JavaScript**: O Guia Definitivo. Bookman, 2012. ROGERS, Y.; SHARP, H.; PREECE, J. **Design de Interação: além da interação homem-computador**. Bookman, 2013. MEYER
- Web Content Accessibility Guidelines (WCAG) 2.1. Disponível em: <https://www.w3.org/TR/WCAG21/>
- https://developer.mozilla.org/en-US/docs/Web/HTML/
- https://www.w3schools.com/tags/
- https://css-tricks.com/archives/
- https://www.w3schools.com/whatis/whatis_htmldom.asp

Bibliografia Complementar

- AMARAL, L. G. **CSS Cascading Style Sheets**: guia de consulta rápida. 2. ed. São Paulo: Novatec, c2006.
- KAWANO, W. **Crie aplicativos Web com HTML, CSS, JavaScript, PHP, PostgreSQL, Bootstrap, AngularJS e Laravel**. Rio de Janeiro: Ciência Moderna, 2016.
- PUREWAL, S. **Aprendendo a Desenvolver Aplicações Web**. Novatec, 2014.
- TONSIG, S. L. **Aplicações na nuvem**: como construir com HTML5, javaScript, CSS, PHP e MYSQL. Rio de Janeiro: Ciência moderna, 2012.
- USABILIDADE.com. Disponível em <http://www.usabilidade.com/>
- TASK-Centered User Interface Design: A Practical Introduction. Disponível em <http://hcibib.org/tcuid/>