DOCUMENT OBJECT MODEL (DOM)

Definição

Quando uma página da Web é carregada, **o navegador cria um *DOCUMENT OBJECT MODEL* (DOM)**. DOM é uma interface de programação para documentos HTML, XML e SVG . Ele fornece uma representação estruturada do documento **como uma árvore**. O DOM define métodos **que permitem acesso à árvore**, para que eles possam alterar a estrutura, estilo e conteúdo do documento. O DOM fornece uma **representação do documento como um grupo estruturado de nós e objetos, possuindo várias propriedades e métodos**.

Utilizando o DOM, o JavaScript obtém todo o poder necessário para criar HTML dinâmico, pondendo:

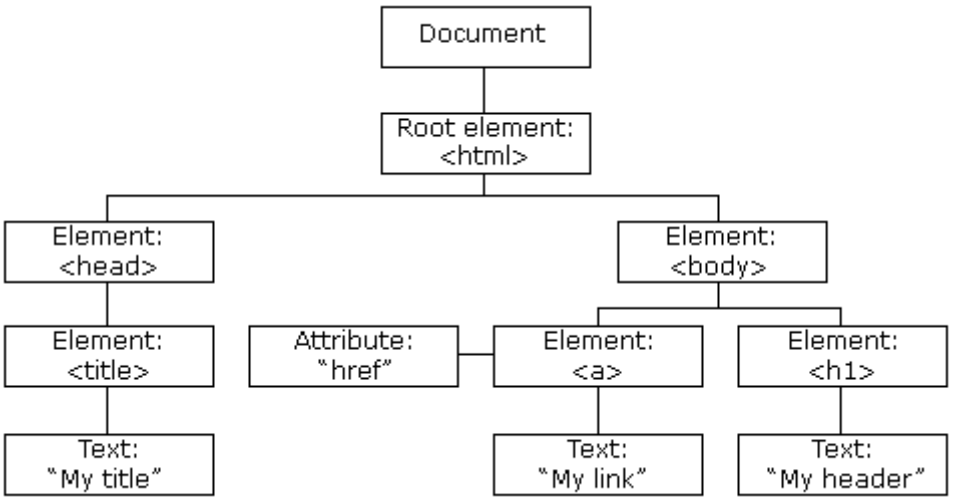
- mudar todos os elementos HTML na página
- alterar todos os atributos HTML na página
- mudar todos os estilos CSS na página
- remover elementos e atributos HTML existentes
- adicionar novos elementos e atributos HTML
- reagir a todos os **eventos** HTML existentes na página
- criar novos **eventos** HTML na página

Embora o DOM seja frequentemente acessado usando JavaScript, **não é uma parte da linguagem JavaScript**. Ele também pode ser acessado por outras linguagens.

Quais as vantagens do DOM?

Com ele você tem infinitas possibilidades, você pode criar aplicações que **atualizam os dados da página sem que seja necessário recarregar todo o código HTML**. Pode também criar aplicações que são **customizáveis pelo usuário**, mudar o layout da página sem que seja necessário atualização. Arrastar, mover, excluir elementos. Ou seja, você tem infinitas possibilidades, milhares de coisas que você pode fazer manipulando o DOM, basta você usar sua criatividade.

Na Figura a seguir apresenta-se o esquema geral de organização de uma árvore DOM HTML.

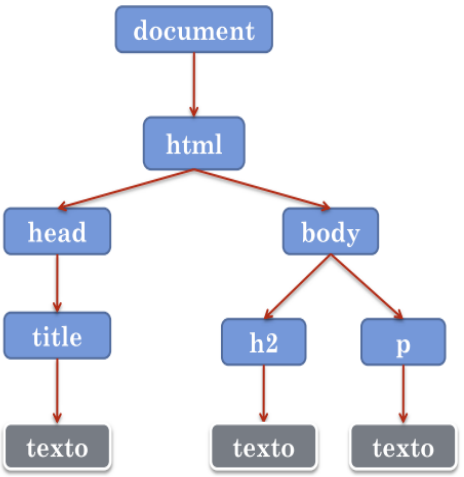


Estrutura da Árvore DOM

Exemplo

```
<!DOCTYPE html>
<html>
<head>
   <title>Testando DOM</title>
</head>
<body>
        <h2 id='titulo'>Document Object Model (DOM)</h2>
   <p >
      DOM permite o acesso e a manipulação de documentos HTML por meio de funções acessíveis ao JavaScript.
   </p>
</body>
</html>
```

Árvore DOM do Exemplo



Testar usando https://software.hixie.ch/utilities/js/live-dom-viewer/

Cada elemento do HTML é representado como um “nó” na árvore DOM

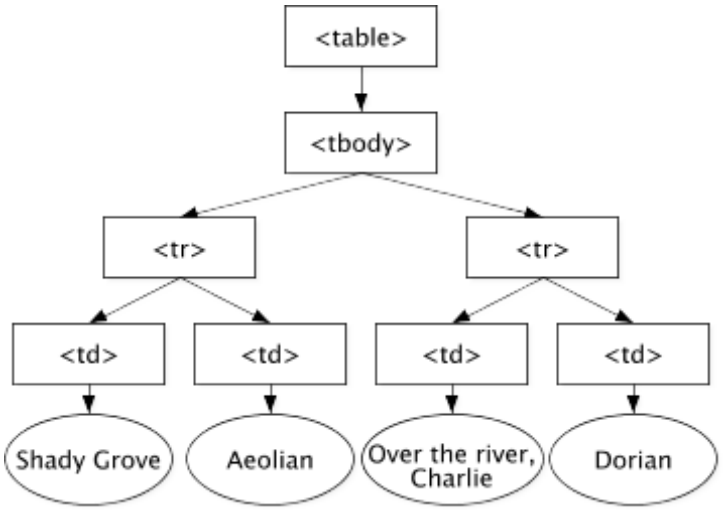
Tipos de nós:

- Elemento: <h2> (exemplo)
- Texto: “Document Object Model (DOM)” ->filho do nó elemento h2 (exemplo)

Nó de documento (document):

- É a raiz de toda a representação DOM
- Ponto de entrada para obter os outros nós

Exemplo da estrutura DOM de uma tabela



Manipulação dos elementos DOM

O objeto do ***document*** representa a página da web. Se você quiser acessar qualquer elemento em uma página HTML, você sempre começará a acessar o objeto de documento.

Localizar Elementos HTML

Método	Descrição
document.getElementById(*id*)	Procura um **elemento** pelo seu **id**
document.getElementsByTagName(*name*)	Procura **elementos** pelo seu **tag**
document.getElementsByClassName(*name*)	Procura **elementos** por sua **classe**

Alterar elementos HTML

Método	Descrição
element*.innerHTML = *new html content	Altera o conteúdo HTML de um elemento
***element*.setAttribute(*attribute, value*)**	Altera o valor de um atributo de um elemento HTML

element*.style.*property* = *new style	Altera estilo de um elemento HTML

Adicionar e excluir elementos

Método	Descrição
document.createElement(*element*)	Cria um novo elemento HTML
document.removeChild(*element*)	Remove um elemento HTML
document.appendChild(*element*)	Adiciona um elemento HTML filho
document.write(*text*)	Escreve na saída HTML

Exemplo de manipulação de elementos DOM via JavaScript

DOM_Exemplo1.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Testando DOM</title>
</head>
<body>
   <h2 id='titulo'>Document Object Model (DOM)</h2>
      <p>DOM permite o acesso e a manipulação de documentos HTML por meio de funções acessíveis ao JavaScript. </p>
  <h2>Novo Parágrafo </h2>
   <script type="text/javascript" src="js/DOM_Exemplo1.js"></script>
</body>
<footer>
   <p id='hora'> Hora</p>
</footer>
</html>
```

DOM_Exemplo1.js

```
//recupera um elemento pelo id e altera a cor
var teste = document.getElementById('titulo');
teste.innerHTML = teste.textContent + ' <u> *Alterado via DOM</u>';
teste.style.color = 'red';

//recupera um elemento pela tag e escreve no final da página
var teste2 = document.getElementsByTagName('h2');
for (i = 0; i < teste2.length; i++) {
   document.write(teste2[i].innerHTML + '<br/>');
}

//usa eventos de timer para mostrar a hora na página
var myVar = setInterval(myTimer, 1000);

//usa eventos de timer para mostrar a hora na página
function myTimer() {
   var d = new Date();
   document.getElementById("hora").innerHTML = d.toLocaleTimeString();
   hr = document.getElementById("hora");
   if (hr.style.color == 'red') {
      //altera o style do elemento html representado na variável teste
      hr.style.color = '#424280';
   } else {
      hr.style.color = 'red';
   }
}
```

Relacionamento entre nós de uma árvore DOM

Em uma árvore:

- Nó topo é chamado de principal (root)
- Todo nó possui um pai (exceto o root)
- Um nó pode ter qualquer número de filhos
- Irmãos (***siblings***) são nós com o mesmo pai
- Um nó sem filhos é chamado de folha

```
<html>
  <head>
    <title>Tutorial DOM </title>
  </head>
  <body>
    <h1>Lição 1 do DOM</h1>
    <p>Hello world!</p>
  </body>
</html>
```

Considerando o HTML acima pode observar que:

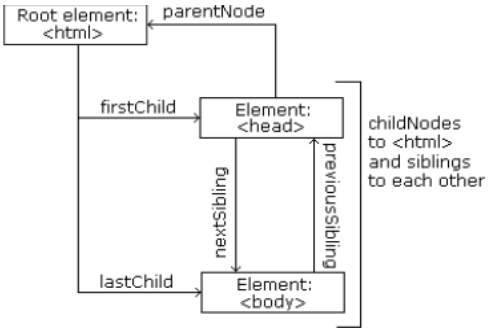
- <html> é o nó raiz
- <html> não tem pais
- <html> é o pai de <head> e <body>
- <head> é o primeiro filho de <html>
- <body> é o último filho de <html>

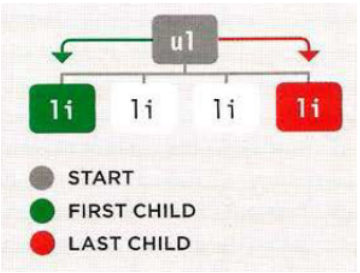
e:

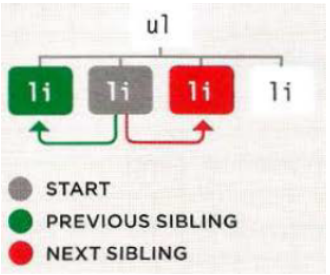
- <head> tem um filho: <title>
- <title> tem um filho (um nó de texto): "Tutorial DOM"
- <body> tem dois filhos: <h1> e <p>
- <h1> tem um filho: "Lição 1 do DOM"
- <p> tem um filho: " Hello world!"
- <h1> e <p> são irmãos

Pode-se usar as seguintes propriedades do nó para navegar entre nós com JavaScript:

- parentNode
- childNodes[nodenumber]
- firstChild
- lastChild
- nextSibling
- previousSibling







Ao utilizar as funções de navegação pelos nós da árvore DOM deve-se observar que **espaços em branco, nova linha, comentários (etc)** são considerados elementos, fazendo com que os procedimentos de navegação seja afetados. Para resolver este problema pode-se utilizar **Jquery** ou a seguinte função, que elimina esta situação

```
function clean(node) {
    for (var n = 0; n < node.childNodes.length; n++) {
        var child = node.childNodes[n];
        //types 1: element, 8:comentário, 3:texto
        if (
            child.nodeType === 8 ||
            (child.nodeType === 3 && !/\S/.test(child.nodeValue))
        ) {
            node.removeChild(child);
            n--;
        } else if (child.nodeType === 1) {
            clean(child);
        }
    }
}
```

Este função pode ser chamada para um elemento específico ou para todo um documento, como no exemplo abaixo:

```
clean(document);
clean(lista1);
```

Abaixo um exemplo demonstrando algumas funcionalidades dos métodos de navegação na árvore DOM

HTML

```
<!DOCTYPE html>
<html>
<head>
    <title id='demo'>Testando DOM</title>
    <meta charset="utf-8">
    <link type="text/css" rel='stylesheet' href="css/estilo.css">
</head>
<body>
    <h1 id='titulo'>Document Object Model (DOM)</h1>
    <h2>Lista de Alunos</h2>
    <ul id='lista1'>
        <li>Pedro</li>
        <li>Maria</li>
        <li>Julia</li>
    </ul>
    <p id='alunos'></p>
    <script type='text/javascript' src='js/ApostilaDOM.js'></script>
</body>
</html>
```

CSS

```
.verm {
    color: red;
}
.azul {
    color: blue;
}
```

JavaScript

```
lista = document.getElementById('lista1');
clean(lista); //remove os caracteres da lista
var node = document.createElement("li"); // Create a <li> node
var textnode = document.createTextNode("João"); // Create a text node
node.appendChild(textnode); // Append the text to <li>
lista.appendChild(node);

parag = document.getElementById('alunos');
parag.innerHTML = "Nr de Elementos na lista:" + lista.childElementCount
+ '<br/>';
parag.innerHTML    +=    "Primeiro    elemento    na    lista    :"    +
lista.childNodes[0].innerHTML + '<br/>';
parag.innerHTML    +=    "Último    elemento    na    lista    :"    +
lista.lastChild.innerHTML + '<br/>';

for (i = 0; i < lista.childElementCount; i++) {
    if (i % 2 == 0) {
        //lista.childNodes[i].style.color = 'red';
        lista.childNodes[i].setAttribute("class", "verm");
    } else {
        //lista.childNodes[i].style.color = 'blue';
        lista.childNodes[i].setAttribute("class", "azul");
    }
}
```