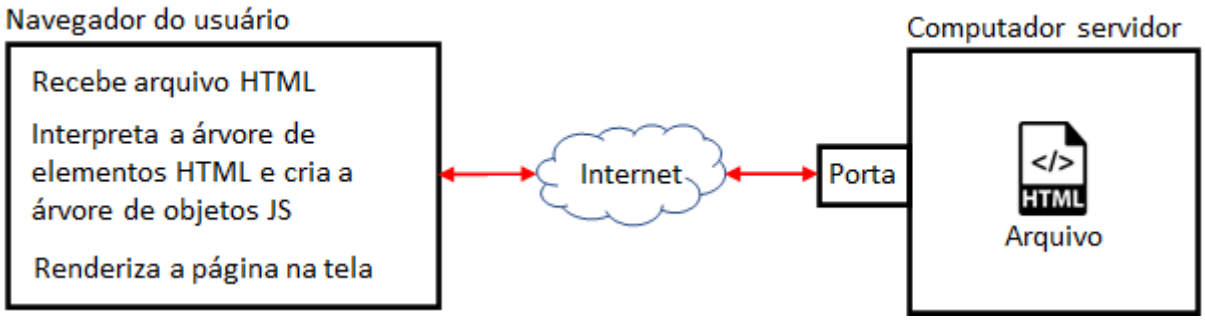
Objetivos:

I. DOM – Document Object Model;
II. Métodos e propriedades do DOM;
III. Métodos para obter nós da árvore;
IV. Métodos para manipular os atributos do nó;
V. Métodos para criar nó e adicionar na árvore;
VI. Método para remover nó;
VII. Eventos;
VIII. Incorporando JavaScript no HTML.

I. DOM – Document Object Model

O processo de carregamento de uma página tem início com a requisição de um arquivo HTML numa máquina servidora. O recurso pode estar na própria máquina (localhost), mas normalmente está em outra máquina e precisa ser acessado usando o protocolo de internet HTTP - Hypertext Transfer Protocol).

O navegador, ao receber o arquivo HTML, faz a interpretação das marcações HTML e cria um objeto JavaScript (JS) para cada marcação HTML e, posteriormente, utiliza os objetos JavaScript para renderizar a página na janela do navegador.



O Document Object Model (DOM) é a representação de dados dos objetos JS que compõem a estrutura e o conteúdo do documento no navegador, ou seja, no navegador o arquivo HTML é chamado de documento após ser interpretado.

O DOM é um padrão do W3C (World Wide Web Consortium) – entidade que regulamenta os padrões de internet.

O HTML DOM é um padrão de modelo de objeto e uma interface de programação que define:

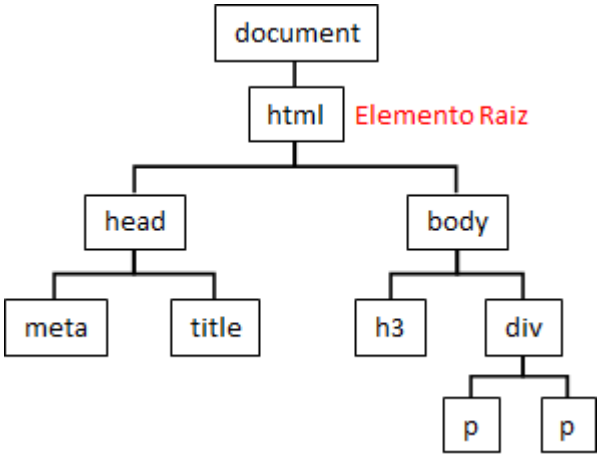
- Os elementos HTML como objetos JS;
- As propriedades dos elementos HTML;
- Os métodos de acesso dos elementos HTML;
- Os eventos dos elementos HTML.

Observação: não confundir HTML (responsável por definir a estrutura da página) e manipulação do DOM (escrito em código JS).

Para mais detalhes https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction.

II. Métodos e propriedades do DOM

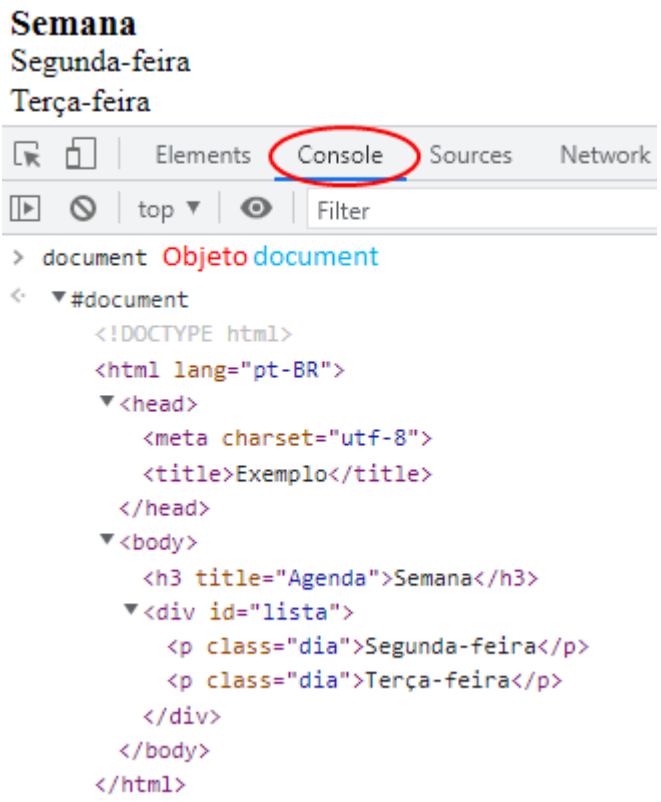
O DOM é construído como uma árvore de objetos, onde um objeto possui outros objetos. A árvore a seguir representa a estrutura do documento HTML.

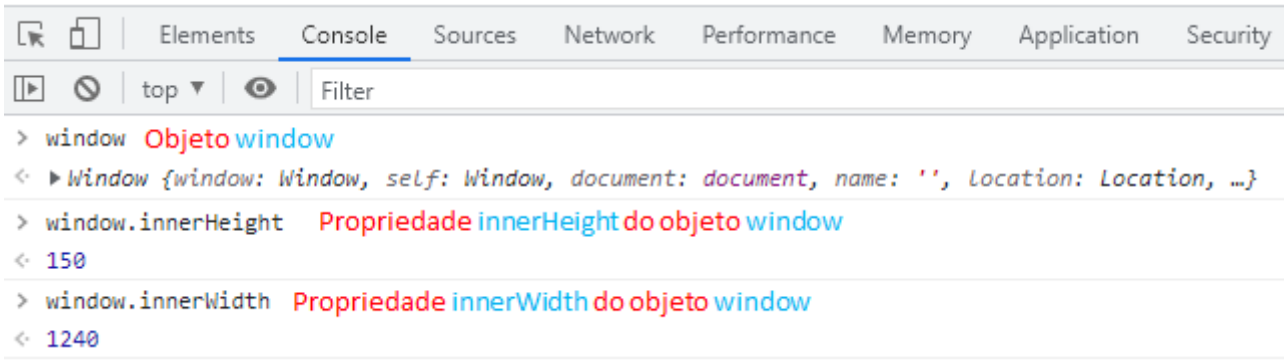



```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>Exemplo</title>
  </head>
  <body>
    <h3 title="Agenda">Semana</h3>
    <div id="lista">
      <p class="dia">Segunda-feira</p>
      <p class="dia">Terça-feira</p>
    </div>
  </body>
</html>
```

O HTML DOM é um padrão de como obter, alterar, adicionar e deletar elementos HTML na árvore de objetos. Na estrutura do DOM é importante conhecer os seguintes objetos:

- document: é o objeto que contém a página HTML. Como exemplo acesse o console do painel de "ferramentas do programador" usando as teclas Shift+Crtl+i. Na aba console digite o objeto document para ver os nós (nodes) filhos:
- node: todo objeto no document é um node. Os objetos podem ser nó de elemento (marcação), um nó de texto (conteúdo do tipo texto de uma marcação) ou nó de atributo (propriedade da marcação);
- window: é o objeto que representa o navegador. Ele não faz parte do DOM.

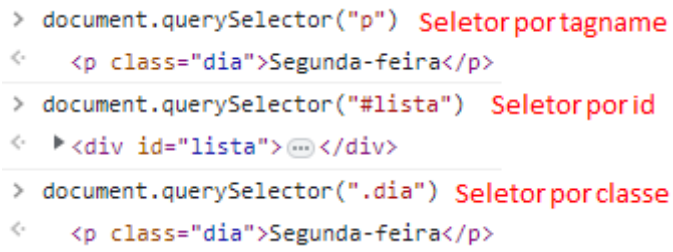




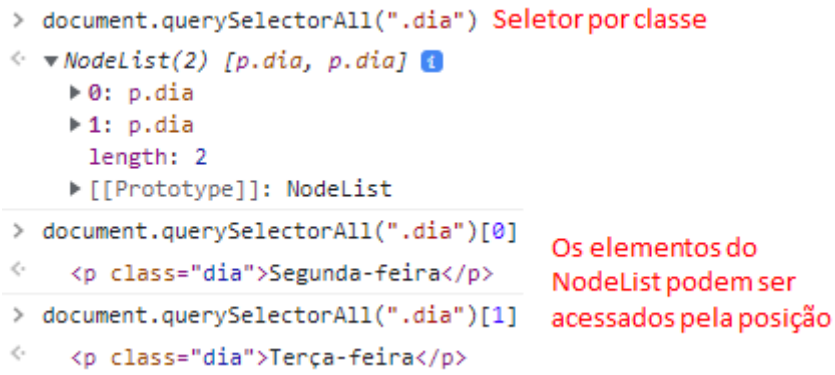
O HTML DOM provê métodos e propriedades para obter, criar, atualizar e excluir nós na árvore de documentos.

III. Métodos para obter nós da árvore

- querySelector(seletor): retorna o 1º elemento que satisfaz o seletor. Observe que mesmo existindo 2 elementos <p> foi retornado apenas o 1º elemento <p>:



- querySelectorAll(seletor): retorna um NodeList com todos os elementos que satisfazem o seletor. Observe que os elementos do objeto NodeList podem ser acessados pela posição entre colchetes:



Os elementos do DOM podem ser acessados usando os métodos a seguir, porém, eles são menos eficientes que os métodos querySelector e querySelectorAll:

- getElementsByTagName(nome da tag): retorna todos os elementos do documento que são do tipo de marcação passado como parâmetro;
- getElementById(identificador): retorna o elemento do documento que possui o atributo id igual ao valor passado como parâmetro;

- getElementsByClassName(classe): retorna todos os elementos do documento que possuem a classe passada como parâmetro.

```
> document.getElementsByTagName("p")
HTMLCollection(2) [p.dia, p.dia]
  0: p.dia
  1: p.dia
  length: 2
  [[Prototype]]: HTMLCollection
> document.getElementById("lista")
  <div id="lista">…</div>
> document.getElementsByClassName("dia")
HTMLCollection(2) [p.dia, p.dia]
  0: p.dia
  1: p.dia
  length: 2
  [[Prototype]]: HTMLCollection
```

IV. Métodos para manipular os atributos do nó

Um nó é formado pelo conteúdo e atributos. Os métodos a seguir são utilizados para acessar os atributos de um nó da árvore:

- getAttribute(nome): retorna o valor do atributo. No exemplo a seguir o método getAttribute retornou o valor do atributo `title` do elemento `<h3 title="Agenda">Semana</h1>`:

```
> document.querySelector("h3").getAttribute("title")
'Agenda'
```

- setAttribute(nome, valor): se o atributo existir, será atualizado o valor do atributo, caso contrário o atributo será criado. No exemplo a seguir o atributo `title` foi alterado para `oie` no elemento `<h3 title="Agenda">Semana</h1>`:

```
> document.querySelector("h3").setAttribute("title","oie")
undefined
> document.querySelector("h3").getAttribute("title")
'oie'
```

- removeAttribute(nome): remove o atributo do elemento. No exemplo a seguir o atributo `title` foi removido do elemento `<h3 title="Agenda">Semana</h1>`:

```
> document.querySelector("h3").removeAttribute("title")
undefined
> document.querySelector("h3").getAttribute("title")
null
```

Para mais detalhes https://developer.mozilla.org/en-US/docs/Web/API/Element/getAttribute, https://developer.mozilla.org/en-US/docs/Web/API/Element/setAttribute e https://developer.mozilla.org/en-US/docs/Web/API/Element/removeAttribute.

V. Métodos para criar nó e adicionar na árvore

- createElement(tagname): cria um elemento do tipo especificado. No exemplo a seguir foi criado um hiperlink e colocado na variável no:

```
> no = document.createElement("a")
<- <a></a>
```

- createTextNode(string): cria um nó de texto. Em outras palavras, ele cria um conteúdo do tipo texto;

```
> texto = document.createTextNode("Fatec")
<- "Fatec"
```

- appendChild(no): coloca o conteúdo no nó. No exemplo a seguir o objeto no receberá o conteúdo da variável texto criando o elemento <a>Fatec:

```
> no.appendChild(texto)
<- "Fatec"
```

- createAttribute(nome do atributo): cria um atributo sem valor. Para atribuir o valor precisa usar a propriedade value. No exemplo a seguir criamos o atributo href="http://www.fatecjacarei.com.br/":

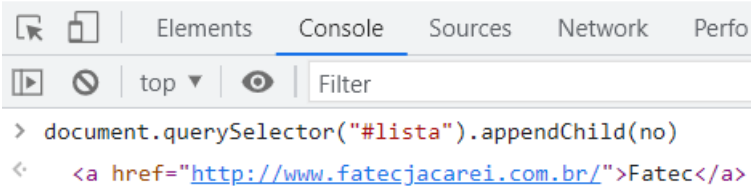
```
> atributo = document.createAttribute("href")
<- href=""
> atributo.value = "http://www.fatecjacarei.com.br/"
<- 'http://www.fatecjacarei.com.br/'
```

- setAttributeNode(atributo): adiciona o atributo no elemento. No exemplo a seguir o objeto no ficará Fatec:

```
> no.setAttributeNode(atributo)
<- null
```

- appendChild(objeto): adiciona um nó ao final da lista de filhos do nó pai especificado. No exemplo a seguir o nó pai é o nó :

Semana
Segunda-feira
Terça-feira
Fatec



Para mais detalhes https://developer.mozilla.org/en-US/docs/Web/API/Document/createElement,
https://developer.mozilla.org/en-US/docs/Web/API/Document/createTextNode,
https://developer.mozilla.org/en-US/docs/Web/API/Document/createAttribute,

https://developer.mozilla.org/en-US/docs/Web/API/Element/setAttributeNode e
https://developer.mozilla.org/pt-BR/docs/Web/API/Node/appendChild.

VI. Método para remover nó

O método removeChild remove um nó do objeto atual. No exemplo a seguir a instrução

```
no = document.querySelector("#lista")
```

foi usada para obter o seguinte elemento do documento:

```
<div id="lista">
  <p class="dia">Segunda-feira</p>
  <p class="dia">Terça-feira</p>
</div>
```

A instrução `no.firstElementChild` é usada para obter o 1º filho do objeto que está na variável no:

```
<div id="lista">
  <p class="dia">Segunda-feira</p>
  <p class="dia">Terça-feira</p>
</div>
```

A instrução `no.removeChild(no.firstElementChild)` remove o filho passado como parâmetro.

```
> no = document.querySelector("#lista")   Obtém o elemento que possui o id=lista
<- ▶<div id="lista">…</div>
> no.removeChild(no.firstElementChild)    Remove o 1º filho do elemento que possui o id=lista
<-   <p class="dia">Segunda-feira</p>
```

Para mais detalhes https://developer.mozilla.org/en-US/docs/Web/API/Node/removeChild.

VII. Eventos

Eventos DOM são utilizados para notificar o código JS de novidades ocorridas durante a navegação do usuário.

Os nomes dos eventos começam com o prefixo on. A seguir tem-se alguns eventos que podem ser adicionados nas marcações como atributos. Lembre-se que um atributo é formado por nome="valor" e no caso dos eventos, o valor será código JS:

- onload: disparado quando o documento termina de ser carregada pelo navegador. O atributo onload precisa estar na marcação <body>. No exemplo a seguir o atributo onload recebeu a instrução console.log('Carregou'):

```
<body onload="console.log('Carregou')">
  <h3 title="Agenda">Semana</h3>
  <div id="lista">
    <p class="dia">Segunda-feira</p>
    <p class="dia">Terça-feira</p>
  </div>
</body>
```

Observação: como usamos aspas duplas para delimitar o conteúdo do atributo onload="", daí precisamos colocar aspas simples para delimitar a string 'Carregou'. Porém, poderíamos usar de forma trocada:

```
onload='console.log("Carregou")'
```

- onclick: disparado quando a marcação recebe um clique do mouse. No exemplo a exemplo o código console.log('clicou') será executado ao clicar na marcação:

```
<p class="dia" onclick="console.log('Clicou')">Terça-feira</p>
```

- onmouseover e onmouseout: disparados, respectivamente, quando o cursor entra e sai da área do elemento na página:

```
<h3
  title="Agenda"
  onmouseover="console.log('Entrou')"
  onmouseout="console.log('Saiu')"
>
  Semana
</h3>
```

- onmousemove: disparado quando o cursor move sobre o elemento na página:

```
<h3 title="Agenda" onmousemove="console.log('Movendo')">
  Semana
</h3>
```

- onchange: disparado ao deixar um campo de entrada que tenha sofrido alteração:

```
<body>
  <label>Entrada</label>
  <input type="text" onchange="console.log('Alterou')" />
</body>
```

Podemos fazer validação quando o usuário deixar o campo de entrada. No exemplo a seguir o texto fornecido pelo usuário será convertido para maiúsculo. O objeto this possui o elemento <input> no código JS e o objeto event possui o evento atual:

```
<body>
  <label>Entrada</label>
  <input type="text" onchange="this.value = event.target.value.toUpperCase()" />
</body>
```

- onkeydown e onkeyup: disparados, respectivamente, ao pressionar e ao liberar uma tecla do teclado. A propriedade key do objeto event possui a tecla pressionada:

```
<input
  type="text"
  onkeydown="console.log('Pressionada:',event.key)"
  onkeyup="console.log('Solta:',event.key)"
/>
```

Para mais detalhes https://developer.mozilla.org/pt-BR/docs/Web/Events.

VIII. Incorporando JavaScript no HTML

O código JS pode ser adicionado no documento HTML de três formas:

- inline (na instrução): o código JS é o valor de uma propriedade da marcação. No exemplo a seguir o código `console.log('clicou')` é o valor do atributo `onclick`:

```
<button onclick="console.log('clicou')">ok</button>
```

- incorporado: o código JS é colocado no corpo da marcação `<script>`. No exemplo a seguir, foi criada a função `testar` no corpo da marcação `<script>`.

  Lembre-se que uma função só será executada ao ser chamada, por este motivo, colocamos a instrução `console.log("clicou")` no corpo de uma função.

  Como valor do atributo `onclick` tem-se apenas a chamada da função `testar()`:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>Exemplo</title>
    <script>
      function testar() {
        console.log("clicou");
      }
    </script>
  </head>
  <body>
    <button onclick="testar()">ok</button>
  </body>
</html>
```

  Observação: a marcação `<script>` pode ser colocada em qualquer parte do documento HTML, porém, recomenda-se manter no `<head>`.

- externo: o código JS fica em um arquivo do tipo .js e será incorporado numa marcação `<script>` do documento HTML. No exemplo a seguir o código do arquivo exemplo.js será colocado entre as marcações `<script>o código do arquivo será colocado aqui</script>` ao carregar o documento HTML no navegador:

Arquivo exemplo.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>Exemplo</title>
    <script src="exemplo.js"></script>
  </head>
  <body>
```

Arquivo exemplo.js

```
function testar() {
  console.log("clicou");
}
```

```
    <label>Entrada</label>
    <button onclick="testar()">ok</button>
  </body>
</html>
```