

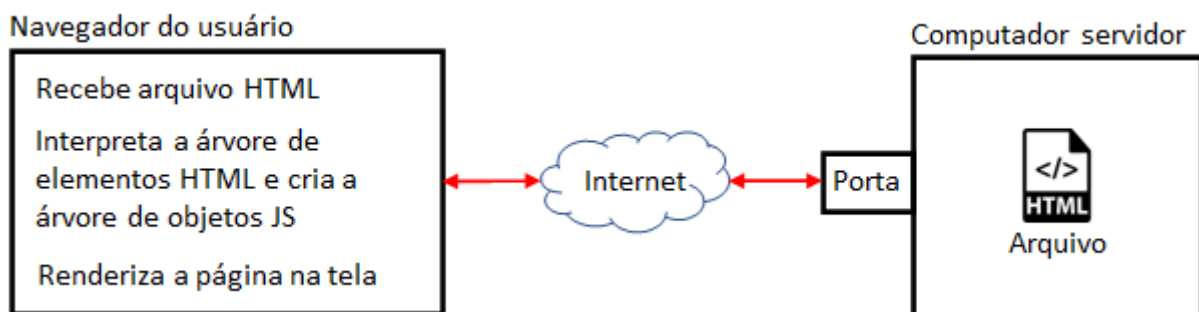
Objetivos:

- 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 - **H**ypertext **T**ransfer **P**rotocol).

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:

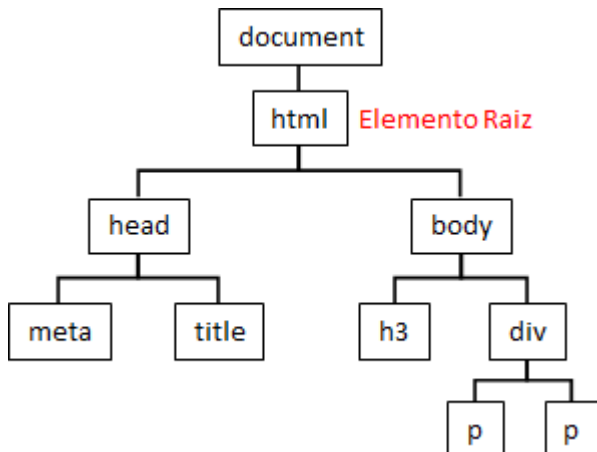
- Os elementos HTML como objetos JS;
- As propriedades dos elementos HTML;
- Os métodos de acesso dos elementos HTML;
- Os eventos dos elementos HTML.

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>
  
```

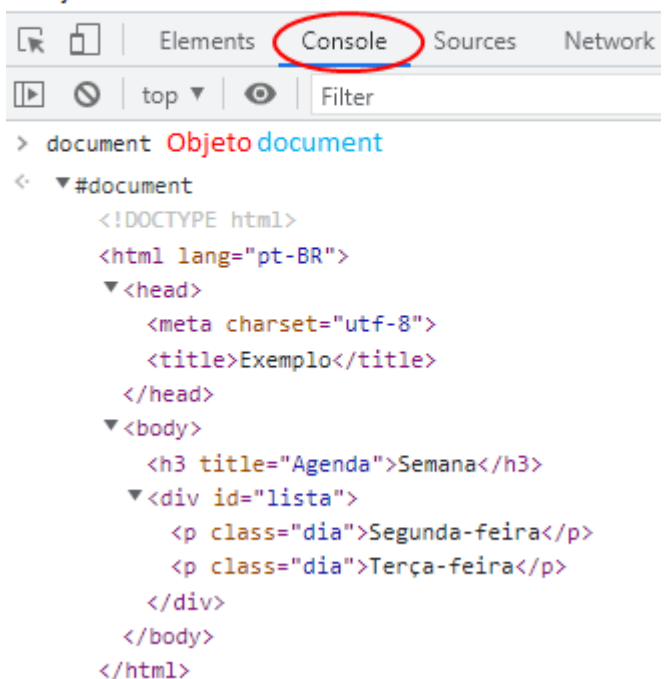
O HTML DOM é um padrão de como obter, alterar, adicionar e deletar elementos HTML na árvore de objetos. Na estrutura do DOM é importante conhecer os seguintes objetos:

- **document**: é o objeto que contém a página HTML. Como exemplo acesse o **console** do painel de “ferramentas do programador” usando as teclas **Shift+Ctrl+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.

Semana

Segunda-feira

Terça-feira



```

> window Objeto window
< ▶ Window {window: Window, self: Window, document: document, name: '', Location: Location, ...}

> window.innerHeight Propriedade innerHeight do objeto window
< 150

> window.innerWidth Propriedade innerWidth do objeto window
< 1240
  
```

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>`:

```

> document.querySelector("p") Seletor por tagname
< <p class="dia">Segunda-feira</p>

> document.querySelector("#lista") Seletor por id
< ▶ <div id="lista">...</div>

> document.querySelector(".dia") Seletor por classe
< <p class="dia">Segunda-feira</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:

```

> document.querySelectorAll(".dia") Seletor por classe
< ▼ NodeList(2) [p.dia, p.dia] ⓘ
  ▶ 0: p.dia
  ▶ 1: p.dia
  length: 2
  ▶ [[Prototype]]: NodeList

> document.querySelectorAll(".dia")[0]
< <p class="dia">Segunda-feira</p>

> document.querySelectorAll(".dia")[1]
< <p class="dia">Terça-feira</p>
  
```

Os elementos do NodeList podem ser acessados pela posição

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</h3>`:

```
> 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</h3>`:

```
> 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</h3>`:

```
> 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
```

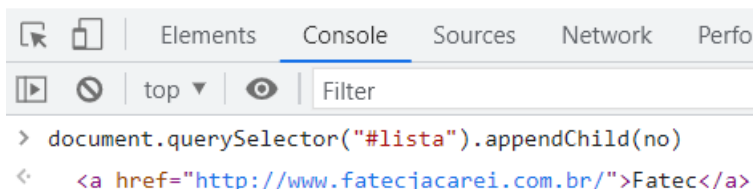
- appendChild(objeto): adiciona um nó ao final da lista de filhos do nó pai especificado. No exemplo a seguir o nó pai é o nó `<div id="lista">`:

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.firstChild` 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.firstChild)` 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.firstChild)  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('clique')` 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('clizou')` é o valor do atributo `onclick`:

```
<button onclick="console.log('clizou')">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("clizou")` 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("clizou");
      }
    </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	Arquivo exemplo.js
<pre><!DOCTYPE html> <html lang="pt-BR"> <head> <meta charset="utf-8" /> <title>Exemplo</title> <script src="exemplo.js"></script> </head> <body></pre>	<pre>function testar() { console.log("clizou"); }</pre>


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

Exercícios

Veja o vídeo se tiver dúvidas nos exercícios: <https://youtu.be/sgQWvGKiE1U>

Exercício 1: Chamar a função exer1 ao carregar o documento HTML no navegador.

Requisitos do exercício:

- Colocar o arquivo exercicio1.js dentro da pasta chamada js;
- O arquivo exer1.js não poderá ser alterado.

Estrutura de pastas e arquivos	Arquivo exercicio1.html	Arquivo exercicio1.js
<div> <div>RESPOSTAS</div> <div> <div>js</div> <div>JS exercicio1.js</div> <div><> exercicio1.html</div> </div> </div>	<pre><!DOCTYPE html> <html lang="pt-BR"> <head> <meta charset="utf-8"> <title>Exercício 1</title> </head> <body> <label>Entrada</label> <input type="text"> <button>ok</button> </body> </html></pre>	<pre>function exer1() { console.log("Carregou"); }</pre>

Exercício 2: Chamar a função exer2 ao clicar no botão ok.

Requisitos do exercício:

- Colocar o arquivo exercicio2.js na pasta js;
- O arquivo exer2.js não poderá ser alterado.

Arquivo exercicio2.html	Arquivo exercicio2.js
-------------------------	-----------------------

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Exercício 2</title>
  </head>
  <body>
    <label>Entrada</label>
    <input type="text" id="entrada">
    <button>ok</button>
    <div id="saida"></div>
  </body>
</html>
```

```
function exer2() {
  // obtém o value do campo entrada
  entrada = document.querySelector("#entrada").value;
  // seta o corpo da div saida
  document.querySelector("#saida").innerText = entrada;
}
```

Exercício 3: Ao clicar no botão ok o conteúdo do campo **entrada** deverá ser copiado para o campo **saida**.

Requisitos do exercício:

- Criar o arquivo exercicio3.js na pasta js;
- Colocar o código numa função chamada exer3 no arquivo exercicio3.js.

Arquivo exercicio3.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Exercício 3</title>
  </head>
  <body>
    <label>Entrada</label>
    <input type="text" id="entrada">
    <button>ok</button>
    <label>Saída</label>
    <input type="text" id="saida">
  </body>
</html>
```

Dica:

- A propriedade value é usada para leitura e escrita.

Exercício 4: Ao clicar no botão ok deverá ser criado um elemento <p> com o seu nome e adicionado no corpo da marcação **saida**.

Requisitos do exercício:

- Criar o arquivo exercicio4.js na pasta js;
- Colocar o código numa função chamada exer4 no arquivo exercicio4.js.

Arquivo exercicio4.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Exercício 4</title>
  </head>
  <body>
    <label>Entrada</label>
    <input type="text" id="entrada">
    <button>ok</button>
    <div id="saida"></div>
  </body>
</html>
```

Exemplo de saída

Entrada

Ana Maria

Ana Maria

Ana Maria

Dicas:

- Use o método createElement para criar um elemento do tipo <p>;
- Use o método createTextNode para criar um texto;
- Use o método appendChild para adicionar o texto no elemento <p>;
- Use o método querySelector para obter a <div> com id=saida;
- Use o método appendChild para adicionar o elemento <p> como filho da <div> com id=saida.

Exercício 5: Alterar o código do Exercício 4 para o elemento <p>, a ser criado, receba o valor do campo de entrada.

Requisitos do exercício:

- Criar o arquivo exercicio5.js na pasta js;
- Colocar o código numa função chamada exer5 no arquivo exercicio5.js.

Dica:

- Use o método querySelector e a propriedade value para obter o valor do campo de entrada.

Exemplo de saída

Entrada

Ana

Pedro

Lúcia

Exercício 6: Alterar o código do Exercício 5 para o elemento <p> ter o atributo title com o valor “Nome fornecido”.

Requisitos do exercício:

- Criar o arquivo exercicio6.js na pasta js;
- Colocar o código numa função chamada exer6 no arquivo exercicio6.js.

Dicas:

Exemplo de saída

Entrada

Ana

Pedro

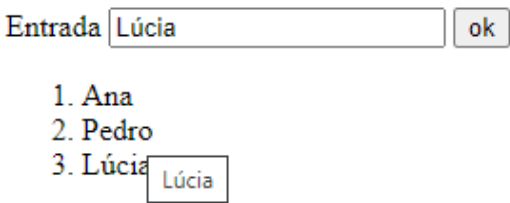
Lúcia

- Use o método `createAttribute` para criar o atributo `title`;
- Use a propriedade `value` para colocar o texto "Nome fornecido" no atributo;
- Use o método `setAttributeNode` para colocar o atributo no elemento `<p>`.

Exercício 7: Alterar o código do Exercício 6 para colocar os nomes no elemento ``.

Requisitos do exercício:

- Criar o arquivo `exercicio7.js` na pasta `js`;
- Colocar o código numa função chamada `exer7` no arquivo `exercicio7.js`;
- O atributo `title` deverá ter o mesmo nome do campo do texto. No exemplo a seguir o cursor está sobre o elemento `<li title="Lúcia">Lúcia`.

Arquivo <code>exercicio7.html</code>	Exemplo de saída
<pre> <!DOCTYPE html> <html lang="pt-BR"> <head> <meta charset="utf-8"> <title>Exercício 7</title> </head> <body> <label>Entrada</label> <input type="text" id="entrada"> <button>ok</button> </body> </html> </pre>	

Exercício 8: Alterar o código do Exercício 7 para ser adicionado somente se existir algum caractere no campo de entrada.

Requisitos do exercício:

- Criar o arquivo `exercicio8.js` na pasta `js`;
- Colocar o código numa função chamada `exer8` no arquivo `exercicio8.js`.

Dica:

- Envolver o código da função por um `if` com um teste se a string possui `length` maior que zero.

Exercício 9: Alterar o código do Exercício 8 para exibir a quantidade de nomes.

Requisitos do exercício:

- Criar o arquivo `exercicio9.js` na pasta `js`;

- Colocar o código numa função chamada exer9 no arquivo exercicio9.js;
- Após adicionar o elemento na tela o campo de entrada deverá ser limpo.

Arquivo exercicio9.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Exercício 9</title>
  </head>
  <body>
    <label>Entrada</label>
    <input type="text" id="entrada">
    <button>ok</button>
    <ol id="saida"></ol>
    <div>Total: <span id="total">0</span></div>
  </body>
</html>
```

Exemplo de saída

Entrada

1. Ana
2. Pedro
3. Lúcia

Total: 3

Dicas:

- Use o método `querySelectorAll` para obter todos os elementos do tipo ``. O método retornará uma estrutura do tipo array;
- Use a propriedade `length` para obter quantidade de elementos do array;
- Use a propriedade `innerText` para setar o texto o elemento que possui `id=total`.

Exercício 10: Alterar o código do Exercício 9 para exibir uma mensagem de alerta ao clicar no nome.

Requisitos do exercício:

- Criar o arquivo `exercicio10.js` na pasta `js`;
- Colocar o código numa função chamada `exer10` no arquivo `exercicio10.js`;
- A função `exibir` deverá ser chamada ao clicar no nome. Não é permitido alterar o código da função `exibir`.

Arquivo exercicio10.html

Arquivo exercicio10.js

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Exercício 10</title>
  </head>
  <body>
    <label>Entrada</label>
    <input type="text" id="entrada">
    <button>ok</button>
    <ol id="saida"></ol>
    <div>Total: <span id="total">0</span></div>
  </body>
</html>
```

```
function exer10() {
}

function exhibir(){
  alert("Exibindo");
}
```

Dica:

- Crie um atributo assim como você fez no Exercício 6;
- O valor do atributo onclick deverá ser "exibir()".