Aula 4 - Páginas Web e JavaScript

aTip Learn - Lógica de Programação com JavaScript

Objetivos da Aula

- Compreender como uma página da Web é estruturada
- Entender a integração entre HTML e JavaScript
- Utilizar o JavaScript para responder a eventos do usuário na página
- Utilizar o JavaScript para manipular o conteúdo da página

Páginas da Web

Quanto utilizamos um navegador, como o Google Chrome ou o Mozilla Firefox, para acessar um site, estamos acessando uma página da Web. Essas páginas geralmente são compostas por três elementos principais:

1. **HTML**: é o que define a estrutura da página, onde definimos os elementos que compõem a página, como títulos, parágrafos, imagens, links, etc.
2. **CSS**: é o que dá o estilo da página, onde definimos como os elementos da página serão exibidos, como cores, tamanhos, fontes, etc.
3. **JavaScript**: é o que cria a interatividade da página, onde definimos como a página irá responder a eventos do usuário, como cliques, movimentos do mouse, etc.

HTML

HTML é a sigla para *HyperText Markup Language*, uma linguagem de marcação utilizada para definir a estrutura de uma página da Web. Um documento HTML é um arquivo de texto, que geralmente possui a extensão `.html`, e é composto por elementos que definem o conteúdo da página.

Esses elementos são definidos por *tags*, que são palavras-chave envoltas pelos símbolos `<` e `>`. Por exemplo, a tag `<h1>` define um título de primeiro nível, enquanto a tag `<p>` define um parágrafo. A maioria dos elementos é composto por uma tag de abertura e uma tag de fechamento, que é a mesma tag da abertura, mas com uma barra `/` antes do nome da tag. Por exemplo, um parágrafo é definido por `<p>` e `</p>`.

```
...
<h1>Esse é um título de primeiro nível</h1>
<p>Esse é um parágrafo de texto.</p>
...
```

Outras tags por sua vez não possuem conteúdo, e são chamadas de *tags vazias*. Por exemplo, a tag `<img>` define uma imagem, e não possui tag de fechamento.

```
...
<img src="imagem.jpg" alt="Descrição da imagem">
...
```

Perceba que a tag `img` possui dois atributos, `src` e `alt`, que são utilizados para definir o caminho para o arquivo contendo a imagem, e uma descrição alternativa para a imagem (utilizada por leitores de tela ou para o caso da imagem não ser carregada), respectivamente.

Para saber quais são as tags disponíveis em HTML, e seus respectivos atributos, você pode consultar a documentação oficial da linguagem, disponível em https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element.

Estrutura Básica de um Documento HTML Todo documento HTML deve possuir uma estrutura básica, que é composta por um elemento `<html>`, que contém um elemento `<head>` e um elemento `<body>`. O elemento `<head>` é utilizado para definir informações sobre o documento, como o título da página, a codificação de caracteres utilizada, links para arquivos de estilo entre outras informações. O elemento `<body>` é utilizado para definir o conteúdo da página.

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="estilo.css">
    <title>Título da Página</title>
  </head>
  <body>
      <h1>Esse é um título de primeiro nível</h1>
      <p>Esse é um parágrafo de texto.</p>
  </body>
</html>
```

Além disso, todo documento HTML deve começar com a declaração `<!DOCTYPE html>`, que define a versão do HTML utilizada no documento. A definição `<!DOCTYPE html>` é utilizada para indicar que o documento utiliza o HTML5, a versão mais recente da linguagem.

CSS

CSS é a sigla para *Cascading Style Sheets*, uma linguagem utilizada para definir o estilo de uma página da Web. Um documento CSS é um arquivo de texto, que geralmente possui a extensão `.css`, e é composto por regras de estilo que definem como os elementos da página serão exibidos.

O primeiro passo para utilizar um arquivo CSS em um documento HTML é vincular o arquivo CSS ao documento HTML. Isso é feito utilizando a tag `<link>`, que é colocada dentro do elemento `<head>` do documento HTML.

```
...
<head>
  ...
  <link rel="stylesheet" href="estilo.css">
  ...
</head>
...
```

O arquivo CSS é composto por um conjunto de regras de estilo, formadas por um seletor e um bloco de declarações de propriedades e valores. O seletor é utilizado para definir a quais elementos a regra de estilo se aplica, enquanto as declarações de propriedades e valores definem como os elementos selecionados serão exibidos. O exemplo abaixo aplicará a cor vermelha ao texto de todos os elementos `<h1>`.

```css
h1 {
  color: red;
}
```

Existem diversos tipos de seletores CSS, que permitem selecionar elementos de diferentes formas. Abaixo alguns exemplos de seletores:

```css
/* Seletor de elemento (ex. seleciona todos os elementos do tipo h1) */
h1 {
  color: red;
}

/* Seletor de classe (ex. seleciona todos os elementos que possuem o atributo class="texto-vermelho") *
.texto-vermelho {
  color: red;
}

/* Seletor de ID (ex. seleciona o elemento que possui o atributo id="titulo") */
```

```css
#titulo {
  color: red;
}
```

Os seletores podem ser combinados para criar regras de estilo mais específicas. Por exemplo, a regra abaixo aplica a cor vermelha ao texto de todos os elementos `<h1>` que possuem a classe `texto-vermelho`.

```css
h1.texto-vermelho {
  color: red;
}
```

Para conhecer mais sobre os seletores CSS e as propriedades disponíveis, você pode consultar a documentação oficial da linguagem, disponível em https://developer.mozilla.org/pt-BR/docs/Web/CSS/Reference.

JavaScript em Páginas da Web

O JavaScript é utilizado em conjunto com o HTML e CSS para adicionar interatividade às páginas da Web.

Até agora criamos e executamos pequenos programas JavaScript por meio do NodeJS. No entanto, quando nosso código JavaScript é executado a partir de uma página da Web, no navegador, ele passa a ter acesso a um conjunto de recursos que permitem que ele interaja com a página, respondendo a eventos do usuário e manipulando o conteúdo da página.

Esses recursos são disponibilizados por meio de uma API (Application Programming Interface) chamada DOM (Document Object Model), que é uma representação da estrutura da página em forma de árvore, onde cada elemento da página é representado por um objeto. Como vimos anteriormente, a página HTML possui uma estrutura onde elementos podem fazer parte de outros elementos, essa mesma estrutura é representada pelo DOM.

Integração entre HTML e JavaScript

Existem várias formas de integrar JavaScript com HTML, a mais simples é inserindo o código JavaScript diretamente no documento HTML, utilizando a tag `<script>`. O código JavaScript pode ser inserido no corpo da página, ou seja no elemento `<body>`, ou no cabeçalho da página, no elemento `<head>`.

```html
<!DOCTYPE html>
<html>
  <head>
    ...
    <script>
      alert('Olá do head!');
    </script>
    ...
  </head>
  <body>
    ...
    <script>
      alert('Olá do body!');
    </script>
    ...
  </body>
</html>
```

No exemplo acima mostraremos duas caixas de mensagem, uma exibida quando o navegador carregar o cabeçalho da página, e outra exibida quando o navegador carregar o corpo da página.

Outra forma de integrar JavaScript com HTML é utilizando arquivos externos. Para isso, criamos um arquivo com extensão `.js` contendo o código JavaScript, e utilizamos a mesma tag `script`, porém, com um atributo

`src` que aponta para o arquivo contendo o código JavaScript.

`exemplo.html`

```
<!DOCTYPE html>
<html>
  <head>
    ...
    <script src="script.js"></script>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

`script.js`

```
alert('Olá do arquivo externo!');
```

Nos dois exemplos acima, utilizamos a função `alert` para exibir uma caixa de mensagem com um texto. A função `alert` é uma função global disponibilizada pelo navegador, que exibe uma caixa de mensagem com o texto passado como argumento. A função `alert` e diversas outras, fazem parte do objeto `window`, que representa a janela do navegador. Todas as propriedades e métodos do objeto `window` podem ser acessados diretamente quando se está utilizando JavaScript em uma página da Web.

Para conhecer as outras funções disponíveis no objeto `window`, você pode consultar a documentação oficial da linguagem, disponível em https://developer.mozilla.org/pt-BR/docs/Web/API/Window.

Eventos

Eventos são ações que ocorrem na página, esses eventos podem ser disparados por interações do usuário, como cliques, movimentos do mouse, pressionamento de teclas, etc. Ou podem partir de fontes internas do navegador e da página, como quando uma página termina de ser carregada. Utilizando JavaScript, podemos definir funções que serão executadas em resposta a esses eventos.

Podemos responder a um determinado evento, como o clique em um botão, adicionando um *event listener* ao elemento que dispara o evento. O *event listener* é uma função que será executada sempre que o evento ocorrer.

```
...
<button onClick="mostrarAlerta()">Clique em mim</button>
...
<script>
  function mostrarAlerta() {
    alert('Você clicou no botão!');
  }
</script>
...
```

No exemplo acima, adicionamos um *event listener* ao botão, esse *event listener* é um trecho de código JavaScript que chama a função `mostrarAlerta` sempre que o botão for clicado. Diferente dos exemplos anteriores, onde a função `alert` era chamada diretamente, agora a função `alert` é chamada a partir de uma função `mostrarAlerta` e por isso a caixa de mensagem não é exibida imediatamente, mas sim quando o botão for clicado.

Adicionar código JavaScript diretamente às propriedades dos elementos HTML, como fizemos acima utilizando a propriedade `onClick`, é uma forma simples de utilizarmos o JavaScript para responde a eventos do usuário.

No entanto, essa não é a forma mais recomendada, pois mistura o código JavaScript com o HTML e torna nosso código menos organizado e mais difícil de manter.

A forma mais recomendada de adicionar *event listeners* a elementos HTML é utilizando o método `addEventListener` do elemento onde desejamos adicionar o *event listener*. O método `addEventListener` recebe dois argumentos, o nome do evento que desejamos escutar, e a função que será executada quando o evento ocorrer.

Assim, para utilizar o método `addEventListener` precisamos primeiro obter uma referência ao elemento HTML. Vamos fazer isso adicionando um `id` ao botão, e então utilizando o método `getElementById` do objeto `document` para obter uma referência ao botão.

```
...
<button id="botao">Clique em mim</button>
...
<script>
  function mostrarAlerta() {
    alert('Você clicou no botão!');
  }
  let botao = document.getElementById('botao');
  botao.addEventListener('click', mostrarAlerta);
</script>
...
```

Nesse exemplo temos uma melhor separação de responsabilidades, o código HTML fica responsável apenas por definir o botão, enquanto o código JavaScript fica responsável por adicionar o *event listener* ao botão e definir a função que será executada quando o botão for clicado.

Manipulação do Conteúdo da Página

Além de responder a eventos, o JavaScript também pode ser utilizado para manipular o conteúdo da página. Podemo por exemplo, criar novos elementos HTML, modificar ou remover os elementos existentes, alterar o conteúdo dos elementos, etc. Vamos começar com um exemplo de um simples contador que é incrementado a cada clique em um botão.

```
...
<button id="botaoIncrementar">Incrementar</button>
<p id="contador">0</p>
...
<script>
  let botaoIncrementar = document.getElementById('botaoIncrementar');
  let contador = document.getElementById('contador');
  let valor = 0;
  function incrementarValor() {
    valor++;
    contador.textContent = valor;
  }
  botaoIncrementar.addEventListener('click', incrementarValor);
</script>
...
```

Nesse exemplo criamos 2 elementos HTML, um botão e um parágrafo, cada um deles identificado pela propriedade `id`. Em seguida, utilizamos o JavaScript para obter uma referência para cada um desses elementos, inicializamos a variável global `valor` com o valor 0, e definimos a função `incrementarValor` que incrementa o valor da variável `valor` e atualiza o conteúdo do parágrafo com o novo valor. Por fim, adicionamos um *event listener* ao botão que chama a função `incrementarValor` sempre que o botão for clicado.

Por que a variável `valor` foi declarada fora da função `incrementarValor`?

Para entender o motivo precisamos entender como nosso código se comportaria caso a variável tivesse sido declarada e inicializada dentro da função. Se fizessemos isso, cada vez que o botão fosse clicado e a função `incrementarValor` fosse chamada uma nova variável valor seria criada e inicializada com o valor 0 para depois ser incrementada. Isso faria com que o valor do contador sempre fosse 1. Utilizando a variável global `valor`, garantimos que o valor do contador não se perca entre as execuções da função, sendo incrementado a cada clique no botão.

Adicionando Novos Elementos

Vamos modificar nosso contador para transformá-lo em uma lista, a cada clique no botão um novo elemento deve ser adicionado à lista para exibir o valor do contador.

```
...
<button id="botaoIncrementar">Incrementar</button>
<ul id="lista">

</ul>
...
<script>
  let botaoIncrementar = document.getElementById('botaoIncrementar');
  let lista = document.getElementById('lista');
  let valor = 0;
  function incrementarValor() {
    valor++;
    let item = document.createElement('li');
    item.textContent = valor;
    lista.appendChild(item);
  }
  botaoIncrementar.addEventListener('click', incrementarValor);
</script>
...
```

Nesse exemplo substituímos o elemento parágrafo `p` por um elemento `ul` (unordered list) que representa uma lista não ordenada. O elemento `ul` está inicialmente vazio, não contendo nenhum item. Além disso, modificamos a função `incrementarValor` para que a cada clique no botão um novo elemento `li` (list item) seja criado por meio do método `createElement` do objeto document, nós então definimos o conteúdo do elemento `li` com o valor da variável `valor`, e adicionamos o elemento `li` à lista por meio do método `appendChild` do elemento `ul`.

O método `appendChild` adiciona um novo elemento filho ao elemento onde ele foi chamado, no final da lista de filhos desse elemento. Além do método `appendChild` você também pode utilizar o método `insertBefore` para adicionar um novo elemento antes do elemento especificado. O método `insertBefore` recebe dois argumentos, o elemento que será adicionado e o elemento que será o próximo irmão do elemento adicionado.

```
...
<button id="botaoIncrementar">Incrementar</button>
<ul id="lista">
  <li id="fimDaLista">Fim da Lista</li>
</ul>
...
<script>
  let botaoIncrementar = document.getElementById('botaoIncrementar');
  let lista = document.getElementById('lista');
  let fimDaLista = document.getElementById('fimDaLista');
  let valor = 0;
```

```
  function incrementarValor() {
    valor++;
    let item = document.createElement('li');
    item.textContent = valor;
    lista.insertBefore(item, fimDaLista);
  }
  botaoIncrementar.addEventListener('click', incrementarValor);
</script>
...
```

Removendo Elementos

Além de adicionar novos elementos, o JavaScript também pode ser utilizado para remover elementos existentes. Vamos modificar nosso contador para que a cada clique no botão um novo elemento seja adicionado à lista, e a cada clique em um item da lista esse item seja removido.

```
...
<button id="botaoIncrementar">Incrementar</button>
<ul id="lista"></ul>
...
<script>
  let botaoIncrementar = document.getElementById('botaoIncrementar');
  let lista = document.getElementById('lista');
  let valor = 0;

  function removerElemento(event) {
    lista.removeChild(event.target);
  }

  function incrementarValor() {
    valor++;
    let item = document.createElement('li');
    item.textContent = valor;
    lista.appendChild(item);
    item.addEventListener('click', removerElemento);
  }

  botaoIncrementar.addEventListener('click', incrementarValor);
</script>
...
```

Nesse exemplo adicionamos um *event listener* a cada item da lista que chama a função `removerElemento` sempre que o item for clicado. A função `removerElemento` recebe um argumento `event`, que é um objeto que contém informações sobre o evento que ocorreu, toda função utilizada como um *event listener* tem acesso a esse argumento `event` que é passado pelo navegador.

O objeto `event` possui diversas propriedades e métodos que nos permitem acessar informações sobre o evento, como o elemento que disparou o evento, as coordenadas do evento, etc. No exemplo acima utilizamos a propriedade `target` do objeto `event` para acessar o elemento que disparou o evento, que nesse caso é o próprio item da lista que foi clicado, utilizamos então o método `removeChild` da nossa lista para remover o item apontado pela propriedade `target`.

Para mais informações sobre o objeto `event` você pode consultar a documentação oficial da linguagem, disponível em https://developer.mozilla.org/pt-BR/docs/Web/API/Event.

Conclusão

Nessa aula tivemos uma introdução à estrutura de uma página da Web, e como o JavaScript pode ser utilizado para adicionar interatividade a essa página. Vimos como responder a eventos do usuário, como cliques em botões, e como manipular o conteúdo da página, adicionando e removendo elementos.

É importante ressaltar que o que vimos é apenas uma pequena parte dos recursos que a linguagem JavaScript oferece para trabalhar com páginas da Web. Existem muitos outros recursos disponíveis, que seriam impossíveis de citar em um curso introdutório como esse. No entanto, esperamos que o que foi apresentado seja suficiente para que você possa começar a explorar por conta própria e aprofundar seus conhecimentos. Lembre-se de utilizar a documentação oficial da linguagem sempre que tiver dúvidas sobre um método ou propriedade, e de praticar bastante para fixar o conhecimento adquirido.

Desafio

O exemplo abaixo contém a estrutura básica de um formulário de login com os campos de email e senha. Utilizando JavaScript, adicione um *event listener* ao botão de login para verificar se os campos foram preenchidos (não estão vazios) e caso estejam vazios exiba uma mensagem de erro. Caso os campos estejam preenchidos, exiba uma mensagem de sucesso.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Formulário de Login</title>
  </head>
  <body>
    <form>
      <h1>Login</h1>
      <div>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email">
      </div>
      <div>
        <label for="senha">Senha:</label>
        <input type="password" id="senha" name="senha">
      </div>
      <button id="botaoLogin">Login</button>
    </form>
    <p id="mensagem"></p>
    <script>
      // Adicione seu código aqui
    </script>
  </body>
</html>
```

Dicas:

- Você precisará obter referências para os campos de email e senha, e para o botão de login.
- Crie uma função que verifique se os campos de login e senha estão em branco, utilize a propriedade `value` dos campos para acessar o conteúdo.
- Utilize o método `alert` para exibir as mensagens de erro ou de sucesso.

Desafio Extra

Você é capaz de modificar sua solução para exibir uma mensagem de erro ao lado de cada campo ao invés de exibir uma mensagem com `alert`?