06

Dê olá ao mundo

Transcrição

Para verificar se o que escrevemos é mesmo um programa, vamos fazer o seguinte:

```
<h1>Meu primeiro teste!</h1>
<br>
Seria isso um programa? Descubra visitando a Alura <a>aqui</a>!
```

Perceba que utilizamos uma nova tag, a `<a>`, que é conhecida como *tag* âncora. Similar ao que ocorre com a `<h1>`, ela também possui um conteúdo, no caso, o texto "aqui". Junto dela também podemos indicar um endereço Web:

```
<a href="http://www.alura.com.br">aqui</a>
```

O que fizemos acima é utilizar uma *tag* HTML de âncora para criar links na Web. Repare que também fizemos uso do `href` que é o que chamamos de atributo. Como isso funciona? Vamos voltar ao navegador e atualizar nossa página.

Perceba que o texto "aqui" é exibido de maneira diferente. Ao clicar no texto somos direcionados para a página inicial da Alura.

O problema do código que escrevemos é que apesar de interessante, o HTML é estático, isto é, não é dinâmico. Por exemplo: se quisermos que o HTML não pule apenas uma linhas, mas dez, ou quinze, ou vinte, nós vamos ter que abrir o arquivo HTML e ficar inserindo diversas *tags* do tipo `<br>`. Algo como:

```
<h1>Meu primeiro teste!</h1>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
Seria isso um programa? Descubra visitando a Alura <a href="http://www.alura.com.br">aqui</a>.
```

Assim, sempre que quisermos aumentar ou diminuir o número de quebras de linha teremos que remover ou adicionar as *tags* `<br>`. Se o HTML é estático e não dinâmico não será possível exibir um contador na tela e tão pouco capturar

informações do usuário. Para que realmente seja possível programar, vamos ter que usar outra linguagem. Além do HTML, o navegador também entende uma linguagem que é verdadeiramente uma linguagem de programação, o **JavaScript**.

> Neste treinamento vamos aprender lógica de programação utilizando a linguagem JavaScript.

Agora, nosso objetivo será exibir um alerta para o usuário, um *pop-up*, que apareça na tela e diga: "isso aqui é uma linguagem dinâmica". Como fazer isso? Primeiro, definimos a mensagem em JavaScript. Assim, é necessário criar um texto e no JavaScript os textos devem vir entre aspas:

```
"Isso sim é um programa"
```

O JavaScript nos permite exibir um *pop-up* na tela e esse *pop-up* é de alerta. Então, é necessário usar uma instrução JavaScript que pede a exibição de um alerta. Para isso é utilizada a instrução `alert` . Porém, existe uma peculiaridade: o texto deve ficar entre parênteses e no final deve ser colocado um ponto e vírgula. Veremos mais detalhes no futuro:

```
alert("Isso sim é um programa");
```

Na teoria, isso já é suficiente para escrevermos nosso primeiro código em JavaScript.

Ao abrir o arquivo, o resultado é o seguinte:

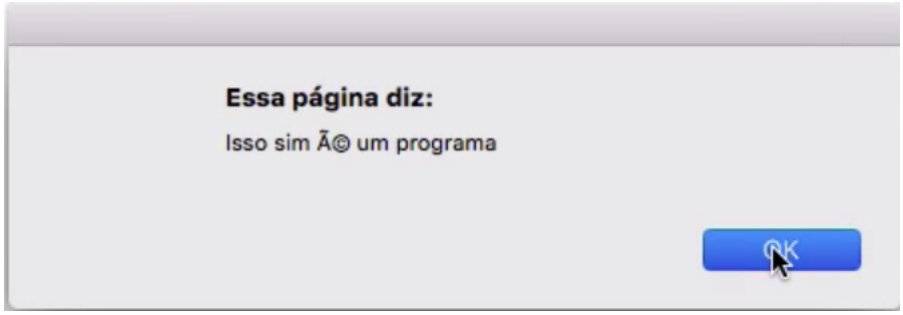
O *pop-up* não aparece!

O texto que gostaríamos que fosse exibido no alerta para o usuário foi exibido no corpo do HTML. Isso ocorre pois o navegador é poliglota e compreende mais de uma linguagem, tanto o HTML quanto o JavaScript. Então, quando escrevemos um código é preciso dar uma pista, sinalizando para o navegador que a linha do `alert` é JavaScript. Vamos indicar que a linguagem é Javascript por meio de uma *tag* chamada `<script>` :

```
<h1>Meu primeiro teste!</h1>
<br>
Seria isso um programa? Descubra visitando a Alura <a href="http://www.alura.com.br">aqui</a>.

<script>

    alert("Isso sim é um programa");

</script>
```

A *tag* é capaz de abrir e fechar e o conteúdo dela é a linha do `alert`. Agora, ao recarregar a página, podemos ver o *pop-up* sendo exibido:

É possível ter problemas com a acentuação dos caracteres. As mensagens que contém acentos podem ser exibidas de maneira estranha, como a imagem a seguir demonstra:



Para resolver o problema da acentuação basta utilizar o padrão `UTF-8`. A *tag* `<meta>` deve ficar na **primeira** linha do arquivo, observe:

```
<meta charset="UTF-8">
```

Quando utilizamos a *tag* `<UTF-8>` estamos dando uma pista para o navegador de como ele deve interpretar as cadeias de caracteres. Então, a simples adição da *tag* resolve o problema da interpretação dos acentos.

Vamos observar a linha abaixo:

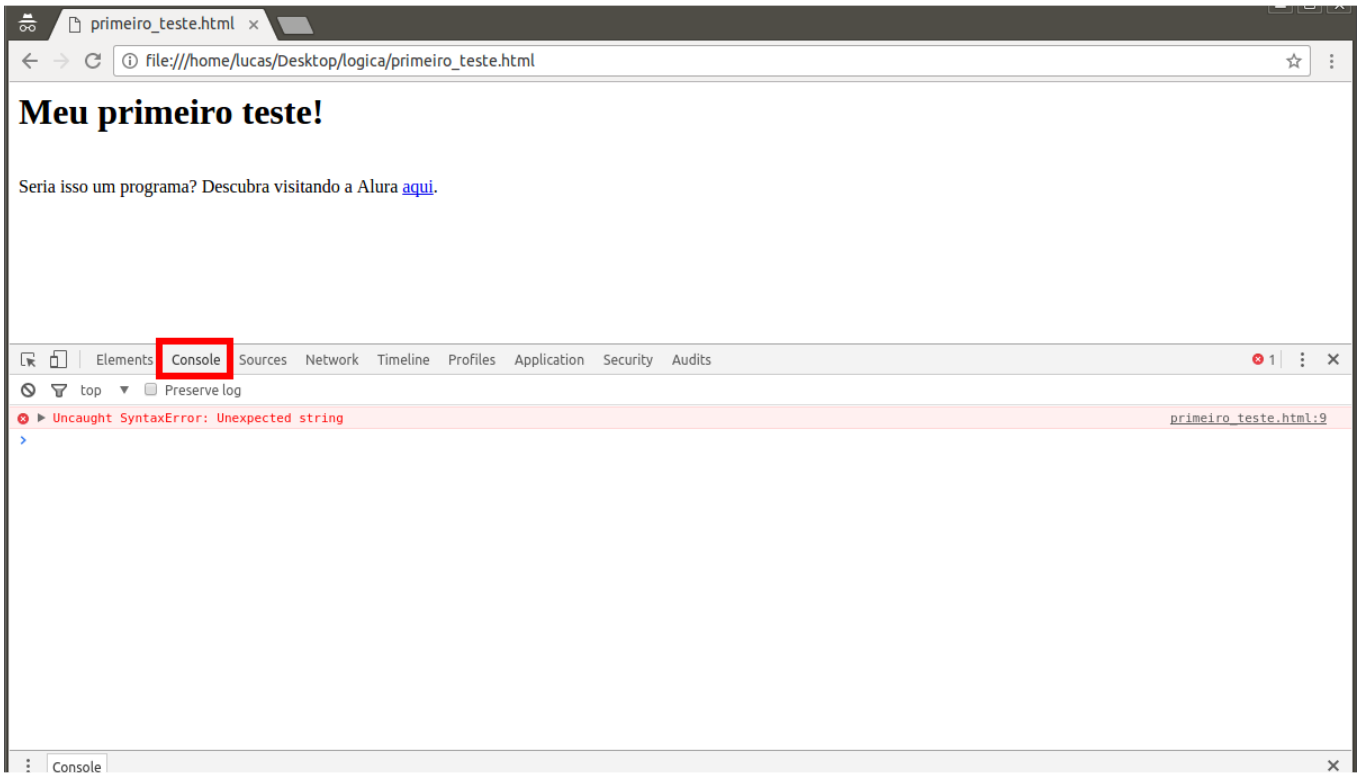
```
alert("Isso sim é um programa");
```

Será que nela precisamos de tudo que está escrito? Os parênteses são mesmo necessários? Vamos fazer o teste! Retiramos os parênteses e recarregamos a página:

```
alert "Isso sim é um programa";
```

Quando carregamos a página nenhum *pop-up* é exibido. Assim, parece que houve um problema! Então, é bom aprendermos logo que todo navegador tem dentro de si uma espécie de *plugin*. O *plugin* funciona como um depurador que nos diz o local do código no qual houve um problema. Você o acessa apertando "F12" ou combinando as teclas "Ctrl + Shift + C".

Na aba console, destacada na imagem a seguir, podemos ver, do lado direito, o nome do arquivo (`primeiro_teste.html`) seguido da linha (9). Note que aparece um aviso de erro de sintaxe: `Unexpected String` . String, em programação, é um nome comum para dados de tipo texto. Assim, o erro indica que temos um texto não esperado.



Note que a linha 9 é exatamente a que alteramos:

```
alert "Isso sim é um programa";
```

Toda linguagem de programação que se preze possui alguma maneira de informar onde ocorrem os erros de sintaxe no código. Dessa forma, é possível saber a linha exata na qual estamos fazendo algo errado. Para corrigir basta voltar os parênteses e salvar o arquivo:

```
alert("Isso sim é um programa");
```

Assim, ao retornar no navegador e recarregar a página é possível ver o alerta sendo exibido novamente.

Então, browser, editor de texto e depurador nos auxiliam a saber onde estão os erros em nosso código, serão ferramentas fundamentais durante todo o treinamento.