04

Concatenação

Transcrição

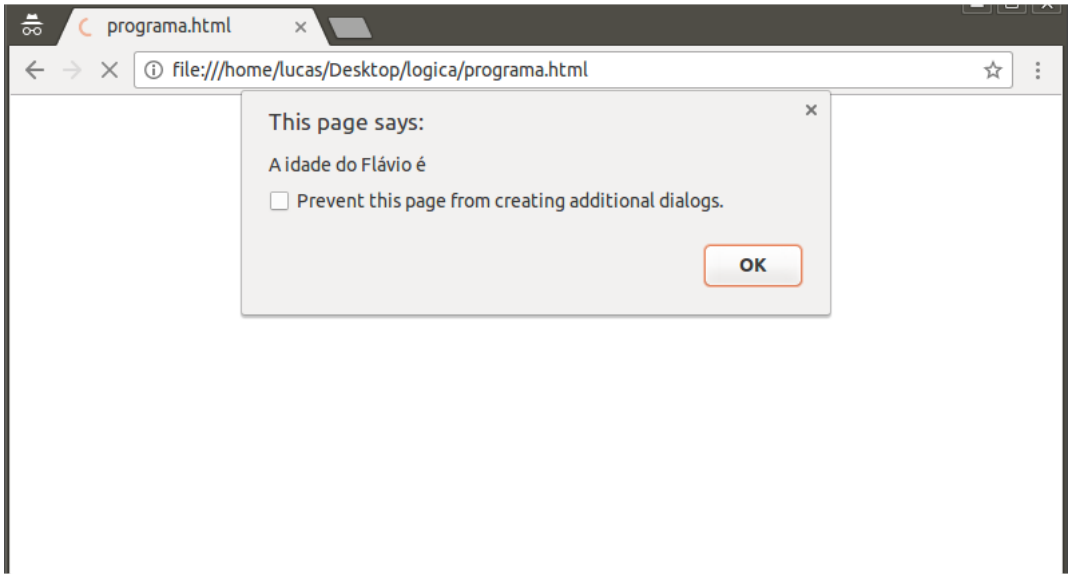
E se quisermos exibir a idade? Vamos simplesmente adicionar a idade no texto:

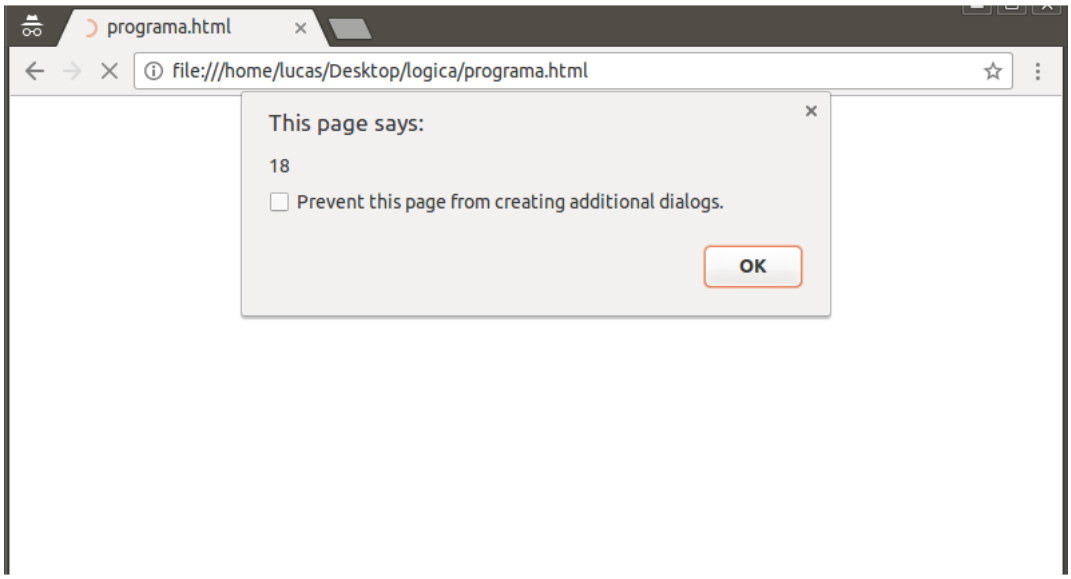
```
alert("A idade do Flávio é 18");
```

Ou podemos adicionar mais um alerta apenas com a idade:

```
alert("A idade do Flávio é");
alert("18");
```

Dessa forma ao salvar e recarregar a página dois alertas são exibidos. Faça o teste!



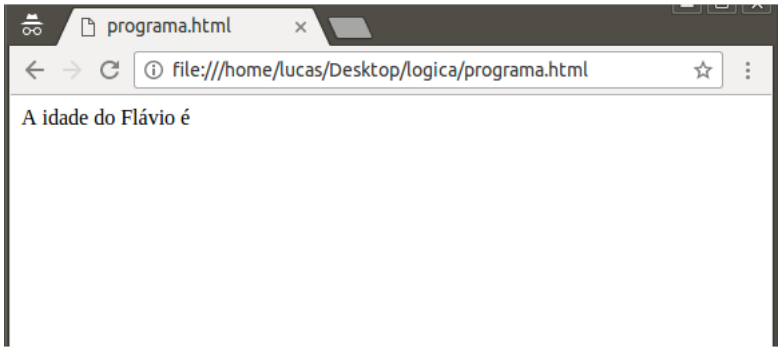


Porém, no Javascript, se utilizarmos `alerts` para exibir textos e tivermos 10 mensagens, quantas vezes o usuário terá que clicar no botão "Ok" que aparece por causa dos `alerts` ? Várias vezes! Da perspectiva do usuário, essa é uma prática ruim. Portanto, não podemos exibir todos os textos da aplicação em `alerts` .

Sabemos que tudo que está dentro da *tag* `script` o navegador entende como JavaScript e tudo que está fora dessa *tag* o navegador entende como HTML. O que queremos é a partir do mundo JavaScript escrever no mundo HTML. Para isso, utilizamos o `document.write` . *Document*, significa "documento" e uma página HTML é um documento e *write* significa escrever. Assim, o `documento.write` espera receber algo entre parênteses, por exemplo:

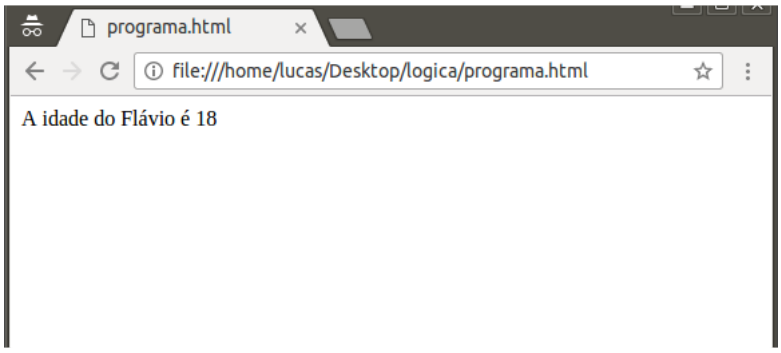
```
document.write("A idade do Flávio é ");
```

Ao salvar e recarregar a página, é possível ver a seguinte mensagem:



Vamos adicionar mais uma linha que contenha a informação da idade:

```
document.write("A idade do Flávio é ");
document.write("18");
```



Algo que pessoas que começaram a programar podem pensar é o seguinte: se escreve muito em JavaScript para escrever algo no mundo HTML. Se tivéssemos escrito diretamente no HTML, o resultado seria o mesmo:

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

A idade do Flávio é
18

<script>

</script>
```

Então, por que se dar ao trabalho de utilizar o `document.write` para do mundo JavaScript escrever no HTML? Porque JavaScript é uma linguagem dinâmica! Aprenderemos que podemos, por exemplo, imprimir algo dez vezes ou imprimir algo apenas se uma ação acontecer. Assim, temos a possibilidade de escrever um código dinâmico e fazer com que as ações também aconteçam de forma dinâmica no mundo HTML. Vimos que podemos mostrar a idade assim:

```
document.write("A idade do Flávio é ");
document.write("18");
```

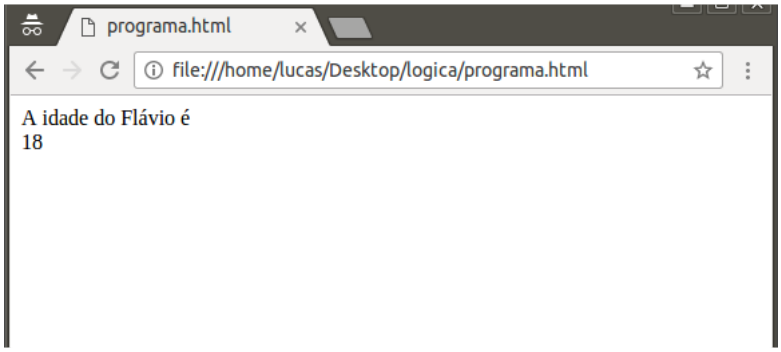
Ou escrever menos e mostrar tudo em uma única String (texto):

```
document.write("A idade do Flávio é 18");
```

Essa última opção é menos verbosa, ou seja, digitamos menos. Mas nem sempre isso valerá a pena!

Agora, nosso objetivo é pular uma linha e fazer com que o "18" fique na linha de baixo. Como fazemos para pular linha no mundo HTML? Utilizamos a *tag* `<br>`, então, vamos passar para o `document.write` o seguinte:

```
document.write("A idade do Flávio é <br> 18");
```



Como estamos escrevendo o texto no mundo HTML ele será processado e ao encontrar o `br` será feita uma quebra de linha. Outra opção seria escrever da seguinte forma:

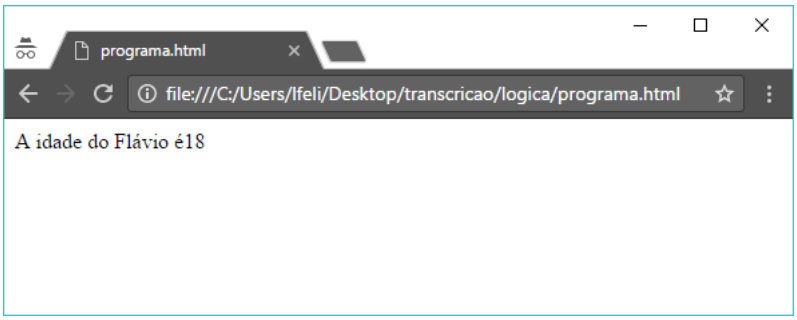
```
document.write("A idade do Flávio é");
document.write("<br>");
document.write("18");
```

Na área de programação existem várias formas de se atingir o mesmo objetivo. Algumas mais verbosas, isto é, que exigem mais escrita e outras diretas e mais simples. Quem ditará qual forma será utilizada é o problema que deve ser resolvido.

Aprendemos que o `document.write` aceita receber um texto. Mas e se escrevermos o código a seguir?

```
document.write("A idade do Flávio é ");
document.write(18);
```

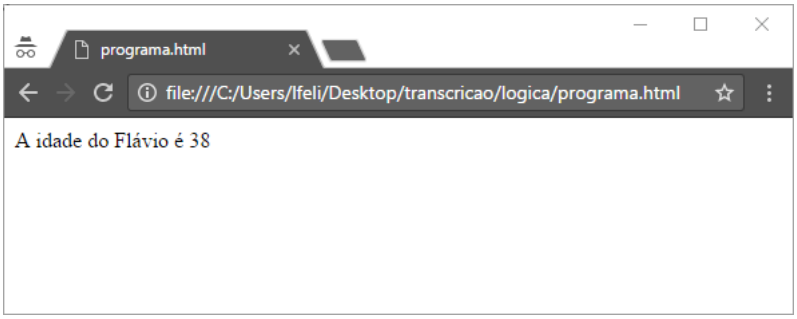
Perceba que no segundo uso do `documento.write` foram removidas as aspas do número 18. Porém, só pode ser considerado o texto que estiver entre aspas. Mesmo assim, será que o código escrito dessa maneira funciona? Vamos salvar o arquivo e recarregar a página. Temos o seguinte resultado:



O valor da idade é exibido! Então, o `document.write` aceita trabalhar com o texto - cujo termo técnico é `String` - acompanhado de um tipo número. O número pode ser utilizado para fazer operações matemáticas como a soma:

```
document.write(18 + 20);
```

O resultado dessa operação é 38:



Com o uso do "*" é possível fazer uma multiplicação:

```
document.write(18 * 20);
```

Com a "/" realizamos uma divisão:

```
document.write(18 / 20);
```

E com o "-", uma subtração:

```
document.write(18 - 20);
```

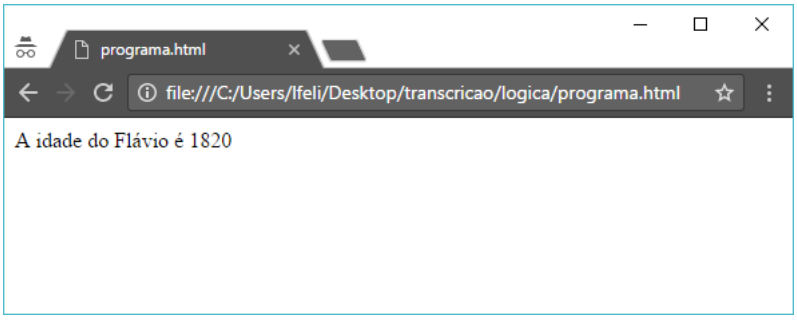
Mas, ao fazer o seguinte:

```
document.write("18");
```

A resposta é que o `"18"` representa um texto no JavaScript. E se escrevermos uma operação matemática colocando aspas nos números? Por exemplo:

```
document.write("18" + "20");
```

Os números passam a corresponder a `Strings`. Será que o resultado será 38? Vamos salvar o arquivo e recarregar a página:



O resultado foi "A idade do Flávio é 1820". Mas, o que houve? Quando existe uma operação envolvendo texto o JavaScript entende que aquilo é um texto e não um número. Quando utilizamos um operador de soma envolvendo texto é realizada uma **concatenação**, isto é, um texto é unido a outro. Por isso o resultado é "1820".

Assim, quando o JavaScript lê a "soma" dos textos ele faz a junção deles o que resulta em "1820". Em seguida o resultado é passado para o `document.write`. Portando:

Primeiro é feita a concatenação e depois o resultado é passado para o `document.write`.

O que acontece se fizermos o seguinte:

```
document.write("18" + 20);
```

Fazendo a soma de um texto com um número temos exatamente o mesmo resultado! O JavaScript é esperto e ao perceber que "18" é um texto ele converte o 20 também para texto ao em vez de lançar um erro.