06

Resumo

Transcrição

Fizemos bastante durante essa lição. Vamos revisar como está seu arquivo agora. Organize-o, criando o arquivo `mostra_idades2.html`. Logo no começo, temos a definição das nossas funções. Começamos pela `pulaLinha`:

```
<script>

    function pulaLinha() {
        document.write("<br>");
    }
```

Logo abaixo vamos ter nossa segunda função, a `mostra`, que por sua vez faz uso da `pulaLinha`. Diferente da anterior, ela recebe um **parâmetro**, que será a frase a ser apresentada no navegador:

```
function mostra(frase) {
    document.write(frase);
    pulaLinha();
}
```

Lembre-se: É importante que sua **indentação** esteja correta para facilitar a leitura do programa.

Após as duas funções declaradas, vamos utilizá-las no nosso código para imprimir quantos anos tem cada um dos envolvidos:

```
    var ano = 2012;
    mostra("Eu nasci em : " + (ano - 25));
    mostra("Adriano nasceu em : " + (ano - 26));
    mostra("Paulo nasceu em : " + (ano - 32));
</script>
```

Vamos a alguns exercícios, começando por uns baseados nesse código.

O `alert` joga uma mensagem dentro de um popup. Utilizá-lo extensivamente pode acabar com a paciência do usuário, que precisará clicar em *OK* a cada nova mensagem. O `document.write` é menos intrusivo, mas você já reparou que as mensagens são jogadas diretamente na página, sem nem mesmo um espaçamento entre as linhas. Isso porque o próprio documento `html` é alterado. Se você quiser pular uma linha através do `document.write`, precisará utilizar tags html, como o `<br>`, fazendo, por exemplo.

```
document.write("olá mundo!<br>");
```

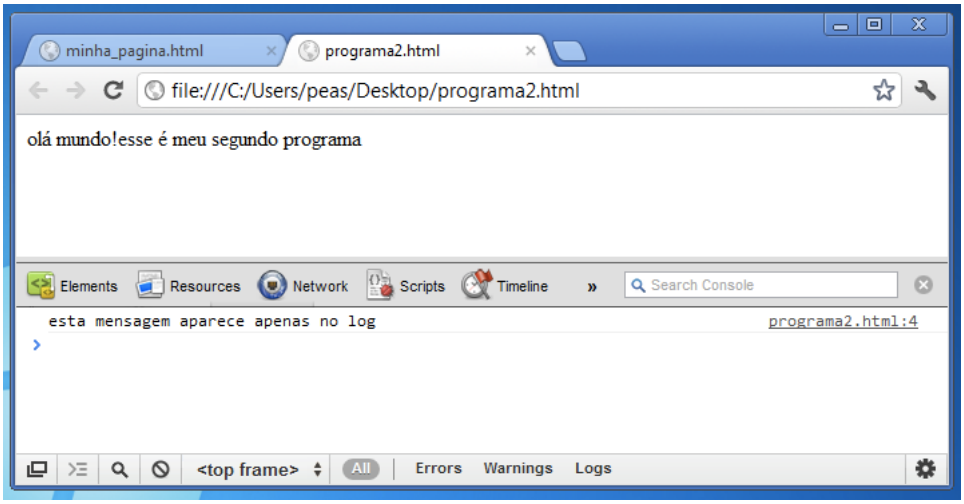
Mesmo colocando o `document.write` em uma função, muitas vezes queremos que algumas mensagens não apareçam para o usuário, porém gostaríamos de poder vê-las durante o desenvolvimento de nosso programa. Isto é, uma mensagem que de alguma forma fosse visível apenas para você, programador. Isso é muito útil para descobrir erros (o que chamamos de **bug**), aprender novos truques e testar recursos. Guardamos esses dados em **logs**. É comum usar o neologismo **logar**, assim mesmo, em português. Para logar informações com JavaScript, há a função `console.log`. Faça um teste:

```
<script>
document.write("olá mundo!");
document.write("esse é meu segundo programa");
console.log("esta mensagem aparece apenas no log");
</script>
```

Qual é o resultado?

A mensagem passada ao `console.log` não apareceu! Quando utilizamos essa função, o navegador guarda todas as mensagens em um local especial, longe da vista do usuário comum. Para ver o resultado precisamos habilitar a visualização do console, exatamente como fizemos na lição anterior para verificar erros.

No Chrome, você faz isso clicando no pequeno ícone de ferramentas/menus, escolhe a opção *Ferramentas* (*Tools*, se estiver em inglês) e depois *Console JavaScript*. É o mesmo console que você usou para ver as mensagens de erro do seu código:



Há a tecla de atalho *CTRL+SHIFT+J* no Windows e no Linux para abrir essa aba (*Command+Option+J* no Mac) e depois clicar no Console, caso outra opção esteja selecionada. Ele realmente é importante e você estará visitando-o com frequência.

Você pode combinar as três funções da melhor forma que encontrar: `alert` para destacar uma mensagem, `document.write` para adicionar informações dentro da própria página e o `console.log` para mostrar dados apenas a nós, programadores.

Também veremos, no decorrer do nosso aprendizado, outras formas e técnicas que nos auxiliam a descobrir problemas no nosso código. Maneiras de remover os **bugs**, isto é, como **debugar** o nosso código.

É também comum querermos colocar uma frase dentro do código que sirva apenas como referência para os programadores. Em JavaScript podemos fazer isso usando `//` . Tudo que vier após o `//` vai ser ignorado pelo navegador. Repare:

```
// esta função mostra uma frase no navegador e pula uma linha
function mostra(frase) {
    document.write(frase);
    pulaLinha();
}

// agora vamos colocar no navegador o ano em que nasci:
var ano = 2012;
mostra("Eu nasci em : " + (ano - 25));
```

Os comentários podem ajudá-lo a organizar melhor o código. De qualquer maneira, é sempre mais importante ter um código bem escrito, com nomes de variáveis expressivas que façam bastante sentido, do que ter de usar muitas linhas de comentários.

Há também a opção de colocar comentários entre /* e */. Dessa forma, tudo que estiver entre esses dois identificadores será ignorado pelo navegador, inclusive se houver quebras de linha.