Eventos

O JavaScript permite iniciar a execução de comandos ou blocos de código a partir de eventos pré-definidos que podem ocorrer em objetos HTML de uma página ou um site. Em outras palavras, é possível executar comandos JavaScript a partir do clique do mouse em alguma parte da página, ao preencher determinado campo ou ainda ao apertar algum botão, entre outros.

Existem vários eventos tratando diversas situações que podem ocorrer na interface ou comportamento de uma página web. Os principais estão listados a seguir:

Evento	Descrição
`onBlur`	Executa o código quando o usuário retira a ação do campo em questão. Utilizado em campos de formulários.
`onChange`	Executa o código quando o usuário altera o valor do campo em questão. Utilizado em campos de formulários.
`onClick`	Executa o código quando o usuário clica sobre o objeto da marcação em questão.

`onFocus`	Executa o código quando o usuário ativa o campo em questão. Utilizado em campos de formulários.
`onLoad`	Executa o código quando a página termina de ser carregada. Geralmente utilizada na marcação `<body>`.
`onUnload`	Executa o código quando o usuário sai da página, seja ao clicar em um link ou fechar o navegador. Geralmente utilizada na marcação `<body>`.
`onMouseOver`	Executa o código quando o usuário passa com o mouse sobre o objeto da marcação em questão.
`onMouseOut`	Executa o código quando o usuário retira o mouse de cima do objeto da marcação em questão.
`onMouseDown`	Executa o código quando o usuário clica sobre o objeto da marcação em questão, mais precisamente ao apertar o botão do mouse.
`onMouseUp`	Executa o código quando o usuário clica sobre o objeto da marcação em questão, mais precisamente ao soltar o botão do mouse.
`onKeyPress`	Executa o código quando o usuário digita (pressiona e solta) uma tecla com o objeto da marcação em questão ativado.

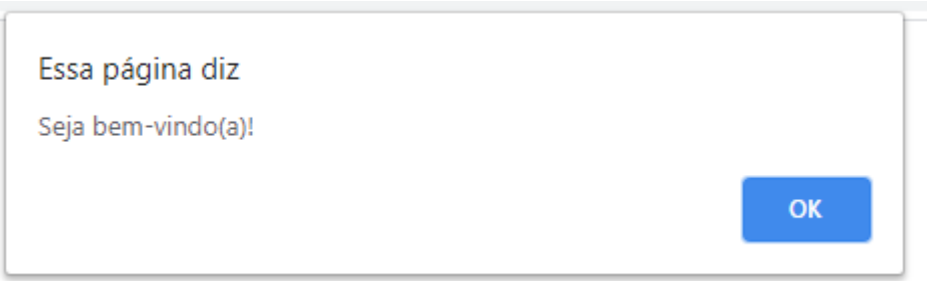
`onKeyDown`	Executa o código quando o usuário pressiona uma tecla com o objeto da marcação em questão ativado.
`onKeyUp`	Executa o código quando o usuário solta uma tecla com o objeto da marcação em questão ativado.
`onSubmit`	Executa o código quando o usuário pressiona o botão de *submit* de um formulário. Geralmente utilizado com a marcação `<FORM>`.

A utilização destes recursos deve ser feita da seguinte forma: incluir o nome do evento que deseja tratar como um atributo de uma marcação HTML de abertura, atribuindo a este o(s) comando(s) que deseja executar. Antes de visualizarmos alguns exemplos de utilização destes recursos, convido você a conhecer um comando JavaScript que apresenta uma pequena janela na tela com uma mensagem customizada para o usuário. Esse comando será útil para a apresentação dos eventos JavaScript e se chama **`alert.`** Considere inicialmente o código apresentado a seguir:

Linguagem JavaScript (Eventos)

EVENTOS01.html

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5          <title>Eventos em JavaScript</title>
6      </head>
7      <body>
8          <script>
9
10             alert("Seja bem-vindo(a)!");
11
12         </script>
13     </body>
14 </html>
```

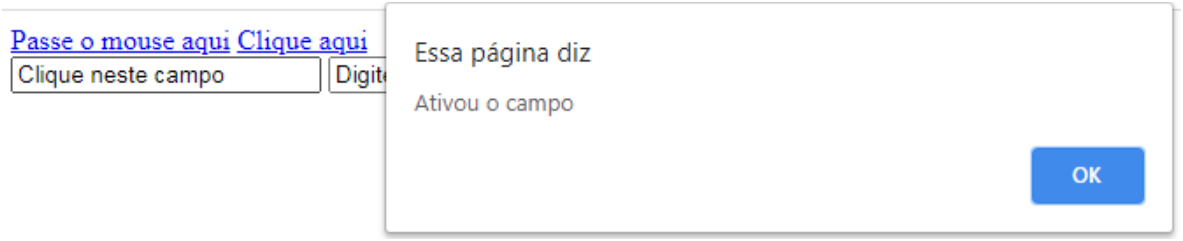


No exemplo apresentado, temos a utilização do comando `alert`, que recebe como parâmetro uma sentença de texto e a apresenta na tela para o usuário, com um botão com o texto OK para o fechamento da janela. Este comando é bastante simples e ao mesmo tempo bastante útil. Agora que você conhece o comando `alert`, vamos dar uma olhada em um exemplo com vários casos de uso de eventos JavaScript em uma página web. Para isso, considere o código apresentado a seguir:

Linguagem JavaScript (Eventos)

EVENTOS02.html

```
1   <!DOCTYPE html>
2   <html lang="pt-br">
3       <head>
4           <meta charset="utf-8">
5           <title>Eventos em JavaScript!</title>
6       </head>
7       <body onLoad="alert('Página carregada!');">
8
9           <a href='#' onMouseOver="alert('Passou o mouse');">Passe o mouse aqui</A>
10          <a href='#' onClick="alert('Realizou um clique');">Clique aqui</A>
11
12          <form>
13              <input onFocus="alert('Ativou o campo');" value='Clique neste campo'>
14              <input onKeyPress="alert('Digitou');" value='Digite neste campo'>
15          </form>
16
17      </body>
18  </html>
```



No exemplo apresentado, várias coisas diferentes podem acontecer dependendo de como você interagir com a página. Inicialmente a página será carregada e imediatamente será apresentada na tela a mensagem Página carregada!, por meio do uso de um comando `alert.` Isso irá ocorrer porque foi programado o evento `onLoad`, dentro da marcação. Observe que o atributo `onLoad` deve apresentar entre aspas as instruções JavaScript que deverão ser realizadas quando a página for carregada. Neste caso foram utilizadas aspas duplas para delimitar o começo e o fim do código que será executado.

Preste bastante atenção no comando `alert` existente entre as aspas do atributo `onLoad`. O comando `alert` poderá causar conflito de aspas caso a sentença que ele imprimirá também seja delimitada por aspas duplas, já utilizadas para delimitar o código que será executado. Por este motivo o uso de outro tipo de aspas foi sugerido, no caso, aspas simples. Esta é uma forma de resolver este problema.

Depois de fechar a mensagem que aparece quando a página é carregada, experimente passar o mouse sobre o link do texto Passe o mouse aqui para ver o que acontece. Como foi programado, a marcação define que o evento `onMouseOver` executará algum código JavaScript. Por este motivo, ao passar o mouse neste texto, o comando `alert` com a frase Passou o mouse será executado.

Já em outro link de texto é apresentado, com o texto Clique aqui, mas neste caso somente disparará o comando `alert` quando o usuário clicar sobre o link, uma vez que foi utilizado o evento `onClick.`

Alguns eventos específicos de campos de formulários também foram utilizados no exemplo do Código-Fonte. O primeiro deles está apresentado, disparando um comando `alert` com o texto Ativou o campo quando o usuário ativar o campo em questão para digitar algum valor. E o segundo, que somente será disparado quando o usuário apertar alguma tecla com o campo ativo, exibindo na tela a sentença Digitou. No exemplo apresentado neste tópico, somente comandos `alert` foram executados a partir dos eventos JavaScript, mas vale a pena reforçar que esses eventos podem disparar qualquer tipo de código JavaScript. Sinta-se livre para brincar com os demais eventos existentes, testando e conhecendo ainda mais suas capacidades. Neste tópico vimos o comando `alert,` utilizado para apresentar mensagens para o usuário.

Caixas de diálogo

O JavaScript disponibiliza alguns métodos de interação com o usuário por meio de caixas de diálogo pré-existentes, que podem ser customizadas pelo programador. Por meio destes recursos é possível exibir alertas na tela, bem como solicitar informações ao usuário. A seguir estão descritas as três formas mais comuns de exibir caixas de diálogos utilizando JavaScript.

Caixa de diálogo alert

O comando `alert`, permite exibir uma mensagem para o usuário. Este método exibe uma janela que se sobrepõe ao conteúdo apresentado pelo navegador, dando prioridade para que sua mensagem seja lida pelo usuário.

A sintaxe de execução deste comando é a seguinte:

```
alert(mensagem);
```

Caixa de diálogo confirm

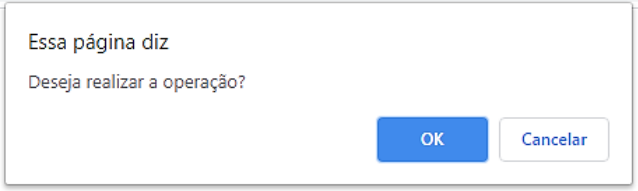
O comando `confirm` é uma extensão do comando `alert.` Assim como o primeiro, este também exibe uma mensagem para o usuário na tela, por meio de uma caixa de diálogo. A diferença é que o método confirm apresenta também para o usuário dois botões: um botão OK e um botão Cancel. Ambos os botões encerram a apresentação da caixa de diálogo. No entanto, é possível detectar no código JavaScript se o usuário utilizou a opção OK ou Cancel e, com base nisso, executar um determinado trecho de código JavaScript ou não.

```
confirm(conteúdo);
```

Linguagem JavaScript (Eventos)

CaixaDialogo.html

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5          <title>Caixa de Diálogo em JavaScript!</title>
6      </head>
7      <body>
8          <script>
9              resposta = confirm('Deseja realizar a operação?');
10
11             if(resposta == true)
12             {
13             alert('Pressionou Ok');
14              }
15             else
16             {
17             alert('Pressionou Cancel');
18             }
19         </script>
20     </body>
21 </html>
```



Caixa de diálogo prompt

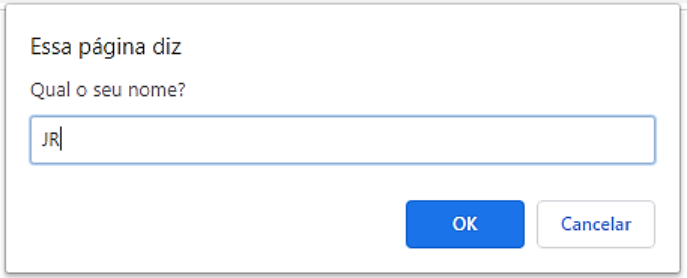
Outra forma de interação com o usuário é por meio das caixas de diálogo do tipo prompt. Esta caixa exibe uma mensagem para o usuário, assim como as demais, porém disponibiliza um campo de texto para que o usuário preencha alguma informação. Esse valor, preenchido pelo usuário, é retornado pela função `prompt` para o seu código JavaScript fazer uso dele. É possível ainda fazer com que o campo texto que será apresentado para o usuário já apareça com algum texto preenchido. Para fazer isso, o segundo parâmetro da função prompt, que é opcional, deve ser informado. A sintaxe de execução deste comando é a seguinte:

```
prompt(conteúdo [, texto_sugerido]);
```

Linguagem JavaScript (Eventos)

prompt.html

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5          <title>Caixa de Diálogo em JavaScript!</title>
6      </head>
7      <body>
8          <script>
9              resposta = prompt('Qual o seu nome?', 'Digite aqui o seu nome');
10             alert('Bem vindo(a) ao site ' + resposta);
11         </script>
12     </body>
13 </html>
```



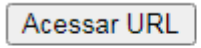
Abrindo uma URL

Entre os diversos objetos que o JavaScript disponibiliza para você utilizar, está o que representa a barra de endereços do navegador: o objeto `location`. Por meio deste objeto é possível acessar a URL da página aberta no navegador e também definir um novo endereço, redirecionando o usuário para outra página dinamicamente via JavaScript. Para compreender o uso do objeto `location` e de sua propriedade `href`, responsável pela informação de endereço de URL do navegador, considere o código-fonte apresentado a seguir:

Linguagem JavaScript (Eventos)

abrindoURL.html

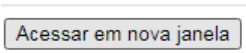
```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5          <title>Abrindo uma URL em JavaScript!</title>
6      </head>
7      <body>
8          <input type=BUTTON onClick="acessarUrl();" value="Acessar URL">
9          <script>
10             function acessarUrl()
11             {
12             location.href = "http://www.google.com.br";
13             }
14         </script>
15     </body>
16 </html>
```



A instrução **`open`** recebe como parâmetro o endereço da URL que deve ser aberta em uma nova guia do navegador. Observe a seguir uma adaptação do código-fonte anteriormente apresentado neste tópico, desta vez fazendo o uso do comando **`open`** do JavaScript:

open.html

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5          <title>abrindo em uma nova guia em JavaScript!</title>
6      </head>
7      <body>
8          <input type=BUTTON onClick="acessarUrlNovaJanela();" value="Acessar em nova janela">
9          <script>
10             function acessarUrlNovaJanela(){
11             open("https://youtu.be/qBf8AyxJTXY");
12              }
13         </script>
14     </body>
15 </html>
16
```



Agendando execução de códigos

Outra funcionalidade muito interessante do JavaScript é o agendamento de execução de códigos. Suponha que após alguns segundos de visita de um visitante na página, você deseje apresentar uma mensagem a ele. Essa e outras necessidades podem ser atendidas por meio do uso do comando `setTimeout.`

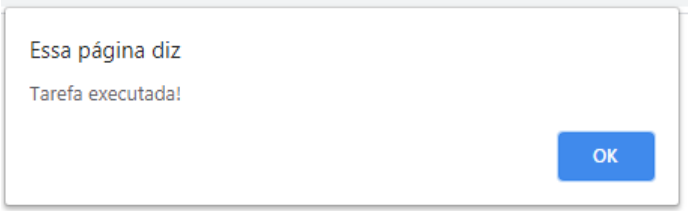
O comando `setTimeout` basicamente opera com dois parâmetros: o primeiro deles é o nome da função JavaScript que você pretende invocar. Por este motivo, é recomendado que o código em questão seja criado dentro de uma função customizada. O segundo parâmetro dessa instrução é o tempo, em milissegundos, que deve haver de intervalo entre a execução do comando `setTimeout` e a execução do código desejado.

Basicamente, este comando informa via JavaScript a função que deve ser executada e daqui há quanto tempo ela deve ser executada. Um exemplo de utilização da função `setTimeout` pode ser observado no código-fonte apresentado a seguir:

Linguagem JavaScript (Eventos)

agendando.html

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5          <title>Agendando execução de códigos em JavaScript!</title>
6      </head>
7      <body>
8          <script>
9              function minhaTarefa(){
10                 alert("Tarefa executada!");
11                 }
12             setTimeout(minhaTarefa, 1000);
13         </script>
14     </body>
15 </html>
```



Alterando uma imagem via JavaScript

Com o JavaScript, é possível acessar boa parte dos recursos das páginas web, se devidamente programados. Este acesso é realizado capturando uma referência para o elemento em questão por meio da instrução `getElementById,` disponibilizada por meio do objeto `document` do JavaScript. Mas para que isso funcione, o elemento deve apresentar o atributo `id` em sua respectiva marcação HTML, atrelando o elemento em questão a um nome único, o que permite o seu acesso por meio deste nome, via JavaScript. Para ilustrar um exemplo de manipulação de elemento de página via JavaScript, considere o código-fonte apresentado a seguir, que construirá um botão HTML responsável por trocar uma imagem da página:

Linguagem JavaScript (Eventos)

alterandoimagem.html

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5          <title>Alterando uma imagem via JavaScript!</title>
6      </head>
7      <body>
8          <img src="img/apagada.png" id="corImagem">
9          <input type=BUTTON onClick="trocarImagem();" value="Trocar imagem">
10         <script>
11             function trocarImagem(){
12             var imagem = document.getElementById("corImagem");
13             imagem.src = "img/acessa.png";
14             }
15         </script>
16     </body>
17 </html>
18
```

Trocar imagem

Trocar imagem