

O que é JavaScript?

- É uma linguagem de script, baseada em ECMAScript padronizada pela Ecma international nas especificações ECMA-262 e ISO/IEC 16262.
- É uma linguagem de programação executada do interior de programas e/ou de outras linguagens de programação, não se restringindo a esses ambientes.
- As linguagens de script servem para estender a funcionalidade de um programa e/ou controlá-lo, acessando sua API - Application Programming Interface (ou Interface de Programação de Aplicativos).
- Essa linguagem é utilizada para construção de páginas da Web utilizando recursos dinâmicos. Podemos criar efeitos especiais, controlar os dados digitados em um formulário e criar algumas animações, e deve ser desenvolvida junto com HTML utilizando editores de texto simples, como o Bloco de Notas, ou editores próprios para gerar HTML, como o Microsoft FrontPage.

O que JavaScript não é?

- É comum confundir a linguagem JavaScript com a linguagem Java, mas, atenção, JavaScript não é Java. Java (desenvolvida pela Sun Microsystems) é uma linguagem de programação orientada a objetos completa, que pode ser usada para projetar aplicações isoladas (que não exigem um browser para rodar) ou mini-aplicações (applets). Outras diferenças:
- A linguagem de programação JavaScript, desenvolvida pela Netscape, Inc., não faz parte da plataforma Java;
- O JavaScript não cria applets nem aplicações independentes;
- Em sua forma mais comum hoje, JavaScript é embutido dentro de um documento HTML e pode fornecer níveis de interatividade com páginas da Web que não podem ser obtidos com HTML simples;
- Java é uma linguagem de programação OOP, ao passo que JavaScript é uma linguagem de scripts OOP;
- Java cria aplicações executadas em uma máquina virtual ou em um browser, ao passo que o código JavaScript é executado apenas em um browser;
- JavaScript é passada ao cliente (browser) como texto e é interpretada. Java é compilada em um tipo especial de código (bytecodes), que são passados ao cliente para serem executados;
- JavaScript usa tipagem fraca - as variáveis não precisam ser declaradas, e uma variável ora pode guardar strings, ora números. Java usa tipagem forte – as variáveis precisam ser declaradas e usadas para um tipo de dados específico;

- O código Java precisa ser compilado, ao passo que os códigos JavaScript estão totalmente em texto.

Configurações

Antes de iniciar o curso, é preciso verificar se o seu navegador está configurado para executar os scripts em JavaScript.

No Internet Explorer

- Vá em “Ferramentas” no menu principal.
- Selecione “Opções da Internet”
- Navegue até a guia “Segurança”
- Clique em “Nível personalizado”
- Depois vá até a seção Script
- Em “Scripts ativos”, selecione “habilitar”
- Clique em “OK”

No Google Chrome

- Vá ao canto superior direito do navegador e clique neste ícone:
- Selecione a opção “configurações”
- Clique em “Mostrar configurações avançadas”
- No item “Privacidade” clique no botão “Configurações de Conteúdo”
- No item “JavaScript” selecione a opção “Permitir que todos os sites executem JavaScript (recomendado)”
- Clique em “OK”
- Segurança

Por ser uma linguagem que é executada no computador do cliente, o JavaScript precisa ter severas restrições para evitar que se façam códigos maliciosos que possam causar danos ao usuário.

As principais limitações do JavaScript para garantia de segurança são a proibição de:

- Abrir e ler arquivos diretamente da máquina do usuário
- Criar arquivos no computador do usuário (exceto cookies)
- Ler configurações do sistema do usuário
- Acessar o hardware do cliente
- Iniciar outros programas
- Modificar o valor de um campo de formulário do tipo `<input>`

No entanto, essas limitações interferem muito pouco no desenvolvimento de aplicações web sérias com a linguagem.

Local do JavaScript

O JavaScript pode ser colocado nas seções **<body>** ou **<head>** de uma página HTML. E o código em JavaScript deve ser inserido entre as **tags <script> e </script>**.

Código no <body>	Código no <head>
<pre><html> <body> <script type="text/javascript"> alert("Código no body"); </script> </body> </html></pre>	<pre><html> <head> <script type="text/javascript"> alert("Código no head"); </script> </head> <body> </body> </html></pre>

JavaScript Externo

Os scripts também pode ser colocado em arquivos externos, que são úteis quando queremos reutilizar o código em muitas páginas web. Esses arquivos tem a extensão .js. Para utilizar um script externo, coloque o nome do arquivo de script no atributo src na tag <script>.

Vantagens:

- Separa HTML e código;
- Facilita a manutenção;
- Acelera o carregamento da página.

Arquivo: exemplo.html	Arquivo: exemplo.js
<pre><html> <head> <script type="text/javascript" src="exemplo.js"> </script> </head> <body> </body> </html></pre>	<pre>alert("Código ja externo");</pre>

Camadas de desenvolvimento

Com a chegada dos Padrões Web, o conceito de desenvolvimento em camadas tornou-se um importante ponto a ser considerado na construção de aplicações Web.

Tal conceito preconiza a separação dos códigos de desenvolvimento em três camadas separadas:

- Camada de estruturação de conteúdos constituída pela marcação HTML
- Camada de apresentação constituída pelas folhas de estilos

- Camada de comportamento constituída pelos scripts que determinam comportamentos como scripts desenvolvidos com JavaScript.

As principais vantagens de adotar a prática de separação das camadas são:

- Elimina a necessidade de repetição de códigos em diferentes páginas;
- Facilita o reaproveitamento de trechos de códigos em outros projetos;
- Facilita a busca e correção de eventuais bugs nos códigos;
- Facilita a manutenção e o entendimento dos códigos.

O Que eu posso fazer com o JavaScript? Exemplos

Validar um valor de entrada

```
<html>
<body>
  <p>Entre com um número maior que 10</p>
  <input id="numero" type="number">
  <button
    type="button"
    onclick="minhaFuncao();">
    Validar
  </button>
  <p id="exemplo"> </p>
  <script type="text/javascript">
    function minhaFuncao() {
      var x, text;
      x = document.getElementById("numero").value;
      if (isNaN(x) || x < 10) {
        text = "Valor inválido";
      } else {
        text = "Valor válido";
      }
      document.getElementById("exemplo").innerHTML = text;
    }
  </script>
</body>
</html>
```

Validar um valor de entrada

Entre com um número maior que 10

Valor inválido

Mudar o estilo de um elemento HTML

```
<html>
<body>
  <p id="exemplo">Mudar o estilo de um elemento HTML</p>
  <script type="text/javascript">
    function minhaFuncao() {
      var x = document.getElementById("exemplo");
      x.style.fontSize = "25px";
      x.style.color = "blue";
    }
  </script>
  <button
    type="button"
    onclick="minhaFuncao();">
    Mude o estilo
  </button>
</body>
</html>
```

Mudar o estilo de um elemento HTML

Mude o estilo

Mudar o estilo de um elemento HTML

Mude o estilo

Mudar a imagem

```
<html>
<body>
  
  <p>Mudar a imagem</p>
  <script type="text/javascript">
    function minhaFuncao() {
      var image = document.getElementById('exemplo');
      if (image.src.match("feliz")) {
        image.src = "sheldon_triste.jpg";
      } else {
        image.src = "sheldon_feliz.jpg";
      }
    }
  </script>
</body>
</html>
```



Mudar a imagem



Mudar a imagem

JavaScript é case sensitive

A linguagem JavaScript é sensível ao tamanho de caixa (case sensitive). Isso significa que nomes de variáveis, funções e demais identificadores são diferenciados quando escritos com letras maiúsculas ou minúsculas. Por exemplo: as variáveis `total`, `Total`, `toTal` e `TOTAL` são diferentes. O método `write()` deve ser escrito em minúscula, pois escrever `Write()` ou `WRITE()` causará um erro no script.

Comentários

Comentários são pequenos textos que o desenvolvedor insere ao longo do script com a finalidade de facilitar o entendimento e a manutenção do código. A linguagem JavaScript admite três tipos de marcadores para comentários:

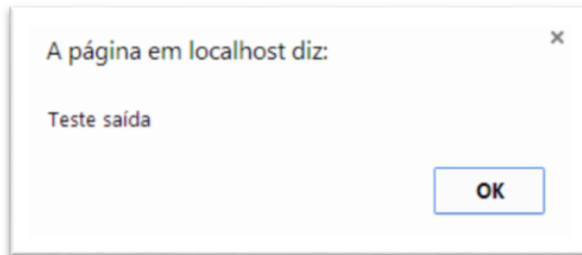
- **Comentário em linha única (variante 1):**
`//comentários`
- **Comentário em linha única (variante 2):**
`<!--comentários`
- **Comentário em múltiplas linhas:**
`/*
Comentários
/*`

Saídas do JavaScript

O JavaScript tem várias possibilidades de exibição de dados, por exemplo:

- Usando uma caixa de alerta: função `window.alert ()` ou apenas `alert ()`.
Exemplo:

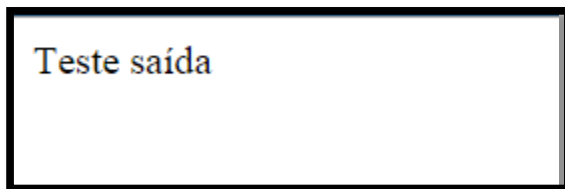
```
<html>
<body>
  <p id="exemplo"></p>
  <script type="text/javascript">
    alert("Teste saída");
  </script>
</body>
</html>
```



- Usando uma saída HTML: função document.write ()

Exemplo:

```
<html>
<body>
  <p id="exemplo"></p>
  <script type="text/javascript">
    document.write("Teste saída");
  </script>
</body>
</html>
```



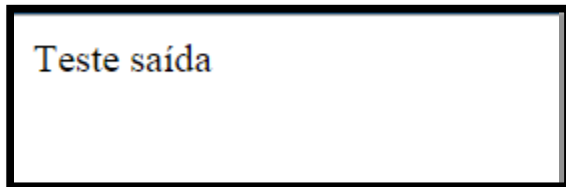
- Usando innerHTML

Para acessar um elemento HTML você pode utilizar o método `document.getElementById(id)`, `id` é o identificador do elemento. Esse método pode retornar o objeto de qualquer elemento na página que tenha um `id` único, e também funciona na maioria dos navegadores.

Já o `innerHTML` serve para alterarmos ou inserirmos conteúdo ou uma marcação HTML em um objeto. Uma das grandes vantagens do `innerHTML` é você manipular dados da página sem que esta precise ser recarregada novamente.

Exemplo:

```
<html>
<body>
  <p id="exemplo"></p>
  <script type="text/javascript">
    document.getElementById("exemplo").innerHTML = "Teste saída";
  </script>
</body>
</html>
```



Entrada de dados usando window.prompt()

Esse é o comando para entrada de dados, usando uma variável que deve ser previamente declarada. Pode ser escrito omitindo-se a palavra window.

Sintaxe:

variável=window.prompt("mensagens para entrada de dados", "título");

Exemplo:

```
<html>
<body>
  <p id="exemplo"></p>
  <script type="text/javascript">
    var x = window.prompt("Entre com seu nome");
    document.getElementById("exemplo").innerHTML = x;
  </script>
</body>
</html>
```

Variáveis

As variáveis permitem guardar dados na memória da máquina durante a execução de programas, páginas ou scripts. Sempre que forem usadas elas estarão prontas para entrarem em ação e receber ou enviar dados conforme a solicitação do usuário.

Regras para criar uma variável

1. Um nome de variável deve sempre começar por uma letra;
2. Não utilize símbolos especiais tais como: (! @ # \$ % ^ & * : ; " ' , . / ? + = ~ `);
3. Procure escrever os nomes das variáveis sempre em minúsculo uma vez que JavaScript é sensível a letras maiúsculas e minúsculas;
4. Procure dar nomes as variáveis que lembrem o seu conteúdo.

Para declarar uma variável em Javascript, você poderá utilizar ou não a denominação **var** (sempre em letras minúsculas), mas não é preciso especificar o tipo de variável. Sempre é recomendado usar a palavra-chave **var**, isto faz com que a variável seja considerada local e não global, evitando conflitos e vulnerabilidades.

Se criar uma variável e atribuir um conteúdo texto e na linha seguinte atribuir um conteúdo numérico, o próprio navegador interpreta a instrução e converte normalmente sem nenhum problema.

Exemplo:

```
<html>
<head>
  <title>exemplo</title>
</head>
<body>
  <script type="text/javascript">
    var nome="Clayton Martins";
    alert("Nome = "+nome);
  </script>
</body>
</html>
```

Essa página diz

Nome = Clayton Martins

OK

Literais

Na terminologia JavaScript, a palavra literal designa qualquer dado – valor fixo (não variável) – que se insere no script. Nos exemplos a seguir, os valores 45 e Alô Mundo! são literais:

```
a = 45;
```

```
mensagem = "Alô Mundo!";
```

Existem seis tipos de dados literais conforme descritos a seguir:

- Inteiros;
- Decimais;
- Booleanos;
- Strings;
- Arrays;
- Objetos.

Inteiros

Os literais inteiros, na sintaxe JavaScript, são os números inteiros escritos em base decimal (base 10), hexadecimal (base 16) ou octal (base 8). Na sintaxe CSS, números hexadecimais deverão ser precedidos do sinal #. Na sintaxe JavaScript, números hexadecimais deverão ser precedidos de 0x.

Exemplo de declarações usando o literal inteiro:

```
c = 32; // base 10
d = -119; // base 10
e = 0x110B6; // base hexadecimal
f = 0xf56a2; // base hexadecimal
```

Decimais

Os literais decimais, na sintaxe JavaScript, são os números constituídos por um número inteiro e um número fracionário. As casas decimais são separadas por um ponto (.), como mostrado nos exemplos a seguir:

```
a = 3.1416;
b = -76.89;
c = .33333;
```

Observação: a sintaxe JavaScript admite a notação científica (ou notação exponencial) para escrever tanto literais inteiros como literais fracionários.

Booleanos

Os literais booleanos, na sintaxe JavaScript, são as palavras-chave `true` e `false` (minúsculas) destinadas a definir as condições de verdadeiro e falso, respectivamente, para uma variável ou equivalente, como mostrado a seguir:

```
a = true;
b = false;
```

Booleanos são amplamente usados em estruturas de controle com a finalidade de testar a veracidade (ou validade) de uma determinada condição, permitindo ao script tomar uma decisão baseado na condição de falso ou verdadeiro da condição.


Strings

Um string é uma sequência de caracteres como "Minha Faculdade", que em javascript deve ser escrito entre aspas. Você pode usar aspas simples ou duplas.

Se você precisar usar aspas simples dentro de uma string em javascript, você pode declarar esta string com aspas duplas. Assim, o interpretador javascript vai interpretar as aspas simples como sendo parte da strings.

Exemplo:

```
<html>
<body>
  <script type="text/javascript">
    var x = "Clayton";
    alert(x);
  </script>
</body>
</html>
```



Essa página diz
Clayton


OK

Arrays

Matrizes de JavaScript são escritas com colchetes, cujos itens são separados por vírgulas.

Exemplo:

```
<html>
<body>
  <p id="exemplo"></p>
  <script type="text/javascript">
    var x = ["Lagoinha", "Jericoacoara", "Pipa"];
    document.getElementById("exemplo").innerHTML = x;
  </script>
</body>
</html>
```




Lagoinha,Jericoacoara,Pipa

Objetos

Objetos JavaScript são escritos entre chaves, cujas propriedades do objeto são escritos como pares de valores, separados por vírgulas.

Exemplo:

```
<html>
<body>
  <script type="text/javascript">
    var usuario = {nome:"Clayton", sobrenome:"Martins",
altura:1.90};
    document.write(usuario.nome);
  </script>
</body>
</html>
```



Clayton

Operadores aritméticos em javascript

São basicamente os mesmos vistos em outros cursos de programação, a saber:

Operador	Símbolo
soma	+
subtração	-
multiplicação	*
divisão	/
resto	%
incremento de 1	++
decremento de 1	--

E os outros já conhecidos operadores de atribuição:

Operador	Exemplo na Forma Normal	Exemplo na Forma Reduzida
+=	x = x + y	x += y
-=	x = x - y	x -= y
*=	x = x * y	x *= y
/=	x = x / y	x /= y

Caso especial do string

O operador + também pode ser utilizado para concatenar strings.

Exemplo:

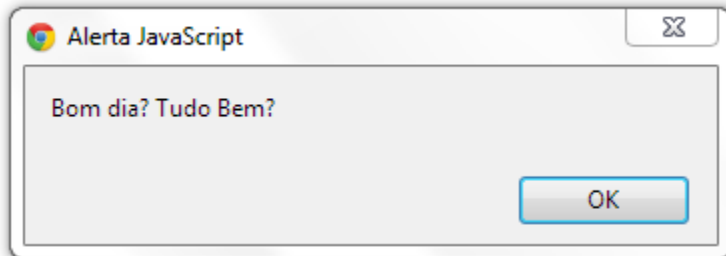
```
<html>
<body>
  <p id="exemplo"></p>
  <script type="text/javascript">
    var nome = "Clayton";
    var sobrenome = " Martins";
    document.getElementById("exemplo").innerHTML =
nome+sobrenome;
  </script>
</body>
</html>
```



Exercícios

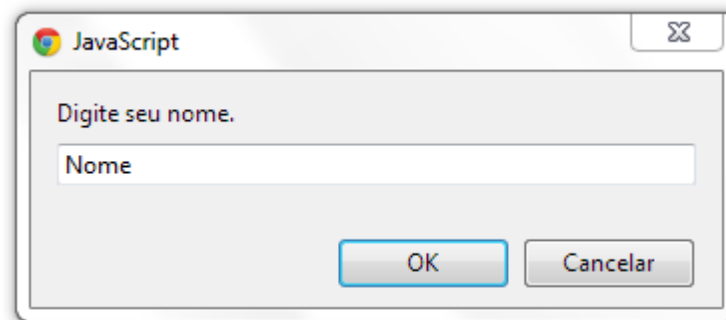
Exercício 1

Criar uma página usando o comando alert para desejar um Bom Dia.
Na tela:



Exercício 2

Escreva um código em Javascript para permitir a entrada de um nome em uma variável e depois exibi-lo em uma caixa de diálogo.



Exercício 3

Crie um código que imprima duas variáveis: nome e sobrenome.

Exercício 4

Transporte o código do item anterior para um arquivo externo e faça sua página chamar este arquivo.

Funções em JavaScript

Uma função é um bloco de código desenvolvido para executar uma tarefa específica, se for chamada. Você pode reutilizar o código: definir o código uma única vez, e usá-lo muitas vezes.

Você pode usar o mesmo código muitas vezes com argumentos diferentes, para produzir resultados diferentes.

Sintaxe básica

```
function nomeFunção(parâmetros) {  
    instruções;  
    return dado;  
}
```

Observações:

- Nomes de funções podem conter letras, números, sublinhados e cifrões (as mesmas regras das variáveis);
- Os parênteses podem incluir nomes de parâmetros separados por vírgulas: (parameter1, parameter2, ...);
- O código a ser executado, pela função, é colocado dentro das chaves;
- Dentro da função, os parâmetros são usados como variáveis locais.

Chamada da função

Para que uma função seja executada, é preciso que a mesma seja chamada. Para isto, basta colocar nome da função.

Exemplo:

```
<html>  
<body>  
    <script type="text/javascript">  
        function primeiraFuncao(){  
            document.write("Minha primeira função");  
        }  
        primeiraFuncao();  
    </script>  
</body>  
</html>
```

Retorno da função

Quando o JavaScript atinge uma instrução de retorno, a função é encerrada e retorna um determinado dado para quem chama.

Exemplo:

```
<html>  
<body>  
    <p id="pagina"></p>  
    <script type="text/javascript">  
        function outraFuncao(){  
            return "Meu primeiro retorno";  
        }  
        document.getElementById("pagina").innerHTML = outraFuncao();  
    </script>  
</body>  
</html>
```

```
    </script>
</body>
</html>
```

Eventos

JavaScript é uma linguagem dirigida por eventos. Eventos (tais como, clicar no mouse, ou pressionar um botão) são utilizados para controlar a interação do usuário com o aplicativo. Programas convencionais funcionam de maneira diferente. Um programa convencional executa seu código sequencialmente.

```
function exibe() {
    if ( !(confirm("Quer encerrar?")) ) {
        documento.write ("Oi, pessoal da Internet!");
    }
}
```

Um programa que queira que o usuário confirme a exibição de uma frase poderia usar a função acima para obter a entrada do usuário. Entretanto, este programa ficaria preso na função `exibe()`, esperando por uma resposta. Enquanto isso, não é possível ter outra operação sendo executada. Quaisquer outras entradas e operações são suspensas até que o usuário responda à pergunta. Uma abordagem melhor seria usar um dos manipuladores de eventos de JavaScript para ativar a função abaixo.

```
function exibe( ) {
    documento.write ("Oi, pessoal da Internet!");
}
```

Manipuladores JavaScript são representados como atributos especiais que modificam o comportamento de uma tag HTML à qual são anexados.

Atributos de manipulação de eventos começam todos com "On" e identificam os diversos eventos que podem ocorrer. O valor associado ao manipulador pode ser uma sequência de declarações JavaScript, ou uma chamada de função JavaScript.

```
<html>
<body>
<script>
    function mostra(){
        if (confirm("Tem certeza de que você é um programador?")){
            alert('OK! Então vamos programar!');
        }else{
            alert('Você precisa estudar mais!');
        }
    }
}
```

```
</script>
<button onClick = "mostra()">CLIQUE AQUI</button>
</body>
</html>
```

Eventos de Sistemas

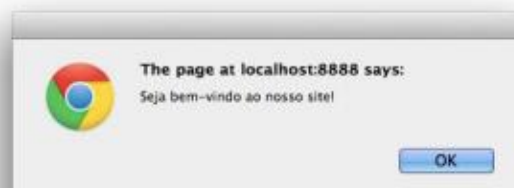
São os que entram em ação automaticamente sem a intervenção do internauta. Este evento é ativado após a página HTML ser completamente carregada. Ele pode ser associado às tags <BODY> ou <FRAMESET>.

OnLoad

Este evento é ativado após a página HTML ser completamente carregada.

Exemplo:

```
<html>
<head>
    <script type="text/javascript">
        function chegada() {
            window.alert("Seja bem-vindo ao nosso site!");
        }
    </script>
</head>
<body OnLoad="chegada()">
    Veja que interessante utilização do evento <|>OnLoad</|>.
</body>
</html>
```



OnUnload

Este evento é ativado após a página HTML ser abandonada (seja através do clique sobre algum link, ou sobre os botões de avanço/retrocesso do browser).

Exemplo:

```
<html>
<head>
```



```
<title>exemplo</title>
</head>
<body OnUnload="fecha()" >
  <script type="text/javascript">
    function fecha(){

window.open("http://www.google.com.br","pesquisa","");
    }
  </script>
  Quando fecha esta janela, abre a janela do Google.
</body>
</html>
```

Eventos de Mouse

São aqueles que acionam ações mediante o uso do mouse, abaixo uma relação dos eventos de mouse:

onClick: O evento mais básico de mouse é tratado pelo manipulador **OnClick**. Este evento é ativado sempre que se dá um clique sobre um objeto que aceita evento de clique de mouse. Objetos que aceitam um evento **OnClick** são links, caixas de verificação e botões.

onDblclick: Ocorre quando o botão do mouse sofrer um clique duplo.

onmousemove: Ocorre quando o ponteiro do mouse passar sobre o objeto.

onmouseover: Ocorre quando o ponteiro do mouse ficar acima do objeto.

onmouseout: Ocorre quando você retirar o ponteiro do mouse do objeto.

onSubmit: Ocorre quando o internauta clica no botão enviar com objetos de formulários. Ele é associado ao objeto form (definido pela tag **<FORM>**).

```
<html>
<head>
  <meta charset="utf-8">
  <script type="text/javascript">
    function VerificaSubmete () {
      var nomeUsuario = document.getElementById ("nomeUsuario");
      if (nomeUsuario.value.length < 3) {
        alert ("O nome do usuário deve ter pelo menos 3 caracteres.");
```

```
        return false;
    }
    var senha = document.getElementById ("senha");
    var resenha = document.getElementById ("resenha");
    if (senha.value.length < 6) {
        alert ("A senha deve ter pelo menos 6 caracteres!");
        return false;
    }
    if (resenha.value != senha.value) {
        alert ("As duas senhas são diferentes!");
        return false;
    }
    var aceitaTermos = document.getElementById ("aceitaTermos");
    if (!aceitaTermos.checked) {
        alert ("Você deve aceitar o Acordo do Usuário para se registrar!");
        return false;
    }
    return true;
}
</script>
</head>
<body>
    <form id="formulario" method="post" action="#URL de destino após
validar corretamente#" onsubmit="return VerificaeSubmete ()">
        Nome de Usuário: <input type="text" name="nomeUsuario"
id="nomeUsuario" />
        <br />
        Senha: <input type="password" name="senha" id="senha" />
        <br />
        Repita a Senha: <input type="password" name="resenha"
id="resenha" />
```

```
<br /><br />

<input type="checkbox" name="aceitaTermos" id="aceitaTermos" />

<label for="aceitaTermos"> Eu aceito o acordo de usuário e a
Política de Privacidade </label>

<br /><br />

<input type="submit" value="Cadastrar" />

</form>

</body>

</html>
```

onFocus: Ocorre quando um objeto ganhar o foco com o click do mouse ou o uso da tecla TAB dentro de um campo do formulário. Isto acontece quando o usuário clica em um objeto específico. Este evento pode ser associado aos objetos text, password, textarea e select (definidos pelas tags <INPUT>, <TEXTAREA> e <SELECT>).

```
<html>

<head>

  <meta charset="utf-8">

</head>

<body>

  <p>Quando você insere o campo de entrada, é acionada uma
  função que define a cor de fundo como amarela.</p>

  <p>Quando você sai do campo de entrada, é acionada uma
  função que define a cor do plano de fundo para vermelho.</p>
```

Digite seu nome: <input type="text" id="entrada" onfocus="funcaoFocus()" onblur="funcaoBlur()">

```
<script>

  function funcaoFocus() {

    // Muda a cor do fundo da entrada de texto para amarelo.

    document.getElementById("entrada").style.background =
"yellow";

  }

  function funcaoBlur() {
```

// Muda para vermelho o fundo da entrada de texto quando clica fora da caixa de entrada, onde estava o onfocus

```
        document.getElementById("entrada").style.background = "red";
    }
</script>
</body>
</html>
```

onChange: Este evento é ativado sempre que um objeto perde o foco e o seu valor é alterado. Ele é associado aos objetos text, password, textarea e select (definidos pelas tags <INPUT>, <TEXTAREA> e <SELECT>).

```
<html>
<head>
    <meta charset="utf-8">
</head>

<body>
```

<p>Modifique o texto no campo de entrada e clique fora do campo para disparar o evento onchange.</p>

Digite algum texto: <input type="text" name="txt" value="Olá" onchange="minhaFuncao(this.value)">

```
    <script>
        function minhaFuncao(val) {
            alert("O valor foi modificado. O novo valor é: " + val);
        }
    </script>

</body>
</html>
```