Tratamento de Eventos no Javascript

Eventos são procedimentos executados em consequência a uma ação. Por exemplo, quando o usuário clica em um botão, é disparado um evento deste elemento chamado “click”.

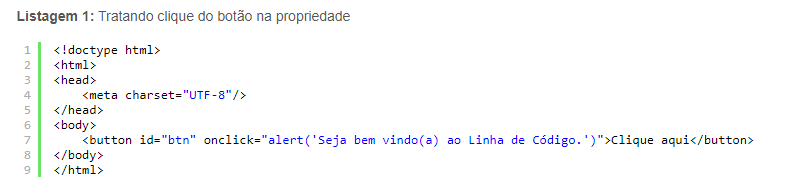
Quando se pressiona uma tecla sobre outro elemento, é disparado um evento chamado “keydown” e quando a tecla é solta, o evento “keyup” é disparado.

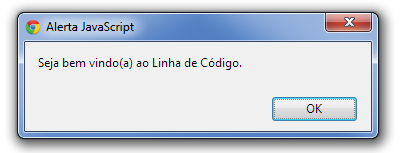
Várias ações, não só na interface gráfica, fazem com que eventos sejam disparados. Então, sabendo que algo é feito quando certas ações são executadas, pode ser interessante e necessário ter controle sobre esta situação.

Nesse ponto entra o TRATAMENTO DE EVENTOS, que consiste em definir o que será feito quando um determinado evento for disparado.

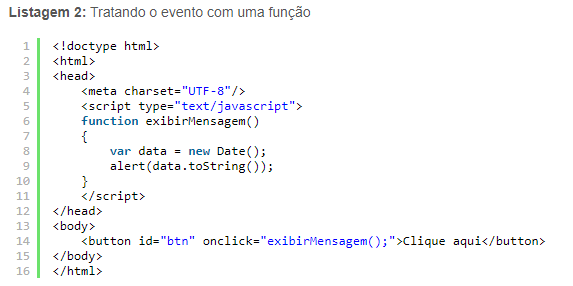
Exemplos de Funções do JAVASCRIPT

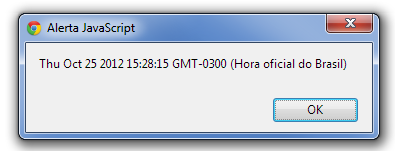
Exemplo da propriedade onclick



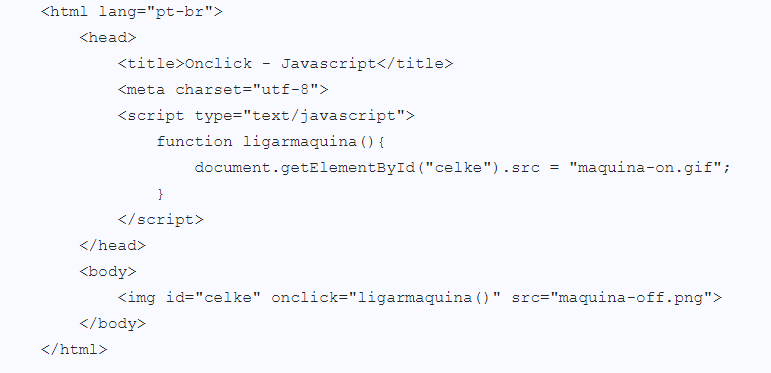


Exemplo com função (mais utilizado):





FUNÇÃO para evento ONCLICK do mouse (escolha usuário):



**FUNÇÃO PARA VALIDAR O CAMPO CPF**

function validarCPF(cpf) {

cpf = cpf.replace(/[^\d]+/g,'');

if(cpf == '') return false;

// Elimina CPFs invalidos conhecidos

if (cpf.length != 11 ||

cpf == "00000000000" ||

cpf == "11111111111" ||

cpf == "22222222222" ||

cpf == "33333333333" ||

cpf == "44444444444" ||

cpf == "55555555555" ||

cpf == "66666666666" ||

cpf == "77777777777" ||

cpf == "88888888888" ||

cpf == "99999999999")

return false;

// Valida 1o digito

add = 0;

for (i=0; i < 9; i ++)

add += parseInt(cpf.charAt(i)) \* (10 - i);

rev = 11 - (add % 11);

if (rev == 10 || rev == 11)

rev = 0;

if (rev != parseInt(cpf.charAt(9)))

return false;

// Valida 2o digito

add = 0;

for (i = 0; i < 10; i ++)

add += parseInt(cpf.charAt(i)) \* (11 - i);

rev = 11 - (add % 11);

if (rev == 10 || rev == 11)

rev = 0;

if (rev != parseInt(cpf.charAt(10)))

return false;

return true;

}

Na **linha 2**, removemos todos os caracteres não númericos do CPF passado como parâmetro, eliminando uma possível máscara.

O condicional if da **linha 5** verifica se número de dígitos da string já limpa é igual a 11 e checa por valores iguais. Esta verificação é necessária uma vez que se aplicarmos o algoritmo do CPF sobre o número "111.111.111-11" teoricamente os dígitos verificadores estão corretos, mas este NÃO é um número válido.

As **linha 17 à 25** verificam se o primeiro dígito verificador é válido de acordo com o algoritmo do CPF. Caso negativo, a validação já retorna false encerrando a função.

Já as **linha 26 à 34** verificam se o segundo dígito verificador é válido.

Caso o algoritmo alcance a **linha 35** temos um CPF válido e o boolean true é retornado.

**FUNÇÃO PARA VALIDAR CAMPO EMAIL:**

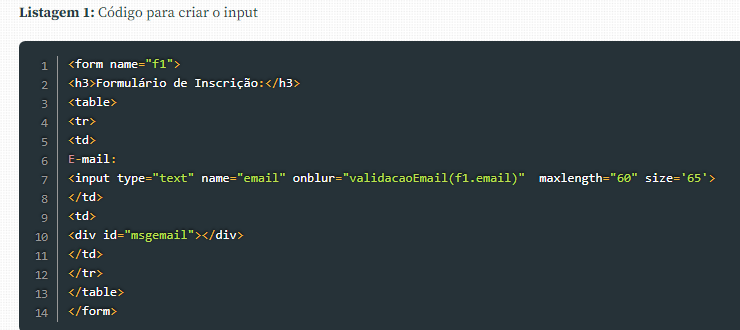
Basicamente um email deve possuir as seguintes opções em seu campo.

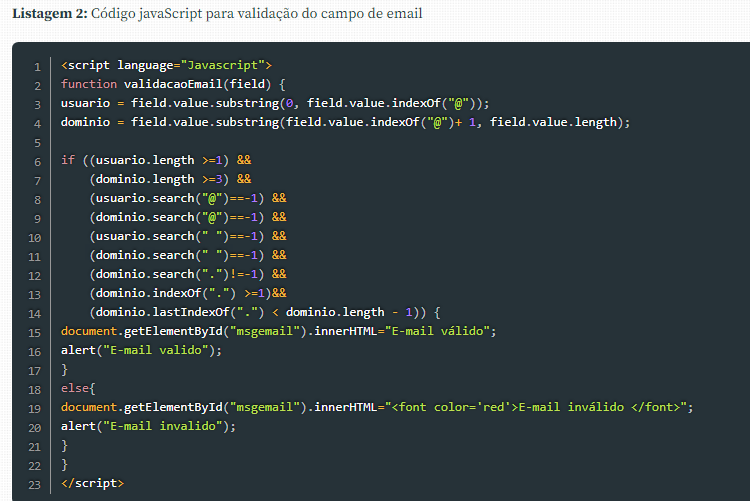
* Não possuir espaços;
* Possuir o @;
* Possuir algum caracter após o @;
* Possuir algum caracter antes do @;
* Possuir pelo menos um ponto após o caracter depois do @;
* Possuir algum caracter após o ponto.

Exemplos de email válido: andre@java.com, andre@java.com.br, andre@java.net .

Como já é de conhecimento de todos, o email possui duas partes separadas pelo @, nesse caso iremos chamar a primeira de usuário e a segunda de domínio.

Exemplo HTML com a chamada da função:





**Criando banners automáticos**

Os banners automáticos são aqueles que independem do usuário.

Nesse exemplo de código abaixo temos três imagens, que serão trocadas entre si a cada segundo, e logo ao chegar na última, retornará a primeira, fazendo, assim, um ciclo infinito.



Funcionamento:

Na body, você deverá inserir uma imagem (img) e determinar um id para representá-la. Será nessa imagem que o evento irá ocorrer.

Dentro da tag head, já na parte superior do código, é que vamos criar esse evento. Eu criei três funções, cada uma irá determinar a imagem e seu tempo de duração.

A primeira função do código é “slide1()”, ela irá determinar que a imagem será 1.bmp, e através do evento setTimeout, ela irá executar o “slide2()” um segundo após ser executada. Esse último, por sua vez, irá trocar a imagem por 2.bmp e depois de um segundo, executará o “slide3()”. Esse fará a mesma troca de imagem, sendo 3.bmp, mas retornará a função “slide1()”, fazendo com que o ciclo continue.

### Criando banners automáticos com links

Esse é parecido com o anterior, mas com a diferença de que cada banner, ao ser clicado, levará a um link diferente.



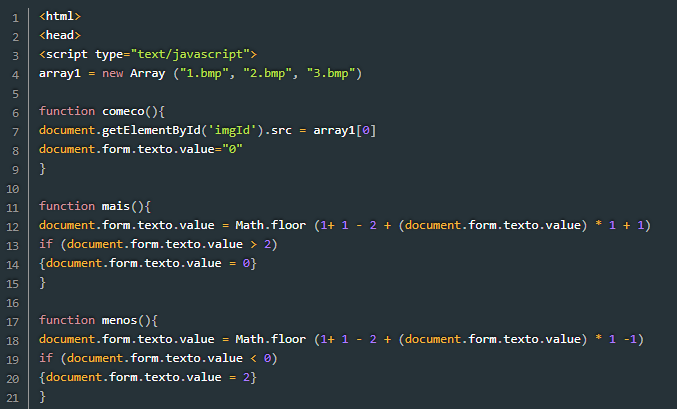
Funcionamento:

Na body, vamos criar uma imagem, assim como no exemplo anterior, mas com a diferença de que essa imagem estará dentro de uma tag “a”, que fará com que ela se torne um link.

Especifique o id dessa nova tag, para que possa ser citada no [**código JavaScript**](http://www.devmedia.com.br/curso/curso-de-javascript-completo/388), em head. Dentro dela, crie a função “slide1()”, que irá determinar a imagem do slide e o tempo de duração, assim como no exemplo do banner sem link, mas com a diferença da parte do código que especifica um “href” (link) para a tag “a”, que a imagem do banner está inserida. Assim também se verifica no “slide2()” e no “slide3()”, onde retorna ao início do banner.

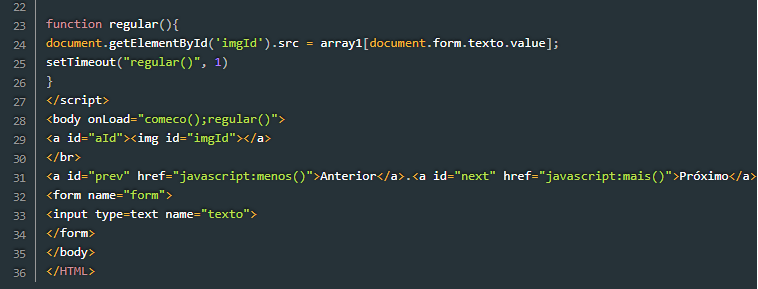
### Criando um slide

Neste exemplo, ele terá três opções de imagens, e botões de navegação para retroceder ou avançar.



No body, vamos criar uma imagem, assim como no exemplo anterior, mas com a diferença de que essa imagem estará dentro de uma tag “a”, que fará com que ela se torne um link.

Especifique o id dessa nova tag, para que possa ser citada no [**código JavaScript**](http://www.devmedia.com.br/curso/curso-de-javascript-completo/388), em head.



Dentro dela, criei a função “slide1()”, que irá determinar a imagem do slide e o tempo de duração, assim como no exemplo do banner sem link, mas com a diferença da parte do código que especifica um “href” (link) para a tag “a”, que a imagem do banner está inserida.

Assim também se verifica no “slide2()” e no “slide3()”, onde retorna ao início do banner.