Capitulo 04 – JAVASCRIPT COM CSS

Sabemos que o CSS (Cascading Style Sheets) é muito usado para auxiliar o desenvolvimento do design de uma aplicação Web. O CSS oferece ferramentas interessantes quanto à formatação de página, fontes, objetos e formulários a serem usados numa aplicação para Internet.

Primeiramente, vamos focar rapidamente na construção de folhas de estilo (CSS).

Temos, basicamente, três tipos de construção de elementos no CSS:

a) "redefinição" de tags HTML:

```
body
{
  text-align       : center;
  background : url('./imagens/fundo3.jpg') no-repeat;
}

h1
{
    font-family      : Bradley Hand ITC, Tahoma, sans-serif;
    font-size        : 30px;
    color            : #ff3300;
    background-color : #ffff00;
    font-style       : italic;
    font-weight      : bold;
}
```

b) criação de classes CSS:

```
.imagem
{
    position         : absolute;
    top              : 100px;
    left             : 680px;
    background       : url('./imagens/flor2.jpg') no-repeat;
    width            : 50%;
    height           : 500px;
}

.posicaoTexto
{
  position         : absolute;
    top              : 500px;
    left             : 650px;
}
```

c) criação de IDs CSS:

- **HTML**

```
<html><head><title></title>
</head>
<link href="menu.css" rel="stylesheet" type="text/css"/>
<body>
  <ul id="menu">
     <li><a href="">Home</a></li>
     <li class="submenu"><a href="">Empresa</a>
          <ul>
               <li><a href="">História</a></li>
               <li class="submenu"><a href="">Funcionários</a>
                    <ul>
                      <li><a href="">Internos</a></li>
                           <li><a href="">Externos</a></li>
                    </ul>
                  </li>
               <li><a href="">Escritório</a></li>
              </ul>
        </li>
     <li><a href="">Serviços</a></li>
    </ul>
</body></html>
```

- **CSS**

```
body  {
  font: 62.5% verdana;
}

ul#menu, ul#menu ul  {
    padding: 0;
    margin: 0;
    width: 150px;
    border-bottom: 1px solid #ccc;
    background: #fff;
    font-size: 100%;
}

ul#menu li  {
  position: relative;
  list-style: none;
}

ul#menu li a  {
  display: block;
    text-decoration: none;
    color: #777;
    border: 1px solid #ccc;
    border-bottom: 0;
}

ul#menu ul  {
  position: absolute;
    display: none;
    left: 149px;
    top: 0;
}

ul#menu li ul li a  {
  padding: 2px 5px;
}

ul#menu li:hover ul ul,ul#menu li:hover ul ul ul  {
  display: none;
}

ul#menu li:hover ul,ul#menu li li:hover ul,ul#menu li li li:hover ul  {
  display: block;
}

ul#menu li.submenu:hover  {
  background-color: #cc0000;
  color: #ffffff;
}
```

CSS e Javascript, combinados com HTML, proporcionam um recurso poderoso para a criação de páginas HTML dinâmicas, onde é possível alterar o design da página sem maiores complicações.

Para isso, é importante saber como combinar CSS e Javascript com o DOM (Document Object Model) para criar efeitos para todos os navegadores.

Para alterar o CSS de qualquer elemento de uma página HTML utilizando Javascript, utiliza-se o método para localizar o elemento que deseja alterar e a propriedade a ser modificada.

Veja um exemplo simples:

- **CSS**

```
#minha_div
{
    width     : 200px;
    height    : 200px;
    background: #00FF00;
}
```

- **HTML**

```
<html>
<head>
<title>DOM - Document Object Model</title>
   <script src="meu_script.js"></script>
   <link rel="stylesheet" href="meu_css.css">
</head>
<body>
  <a href="" id="clique">Clique</a>
  <div id="minha_div"></div>
</body>
</html>
```

- **JAVASCRIPT**

```
window.onload=function()
{
    // localizando o elemento div da página HTML
    var div = document.getElementById('minha_div');

    // localizando o elemento link da página HTML
    var clique = document.getElementById('clique');

    // programando o evento CLICK do elemento link
    clique.onclick=function(){
        // Verifica se getComputedStyle é suportado
        if( 'getComputedStyle' in window ) {
            var largura = window.getComputedStyle(div).width;
        } else {
            // Obtém a largura para navegadores antigos
            var largura = div.currentStyle.width;
```

```
        }

        // configurando largura máxima
        var largura_maxima = 1000;

        // função que aumentará a largura do objeto
        function aumenta_largura() {
                // Aumento em 100 px a largura
                largura = parseInt( largura ) + 100;

                // configurando a nova largura do div
                div.style.width = largura + 'px'

                // Verifica se a largura atingiu a largura máxima
                if ( largura <= largura_maxima )  {
                        // Continua chamando a função até atingir
                        setTimeout(aumenta_largura, 50);
                }
        };

        // chamada da função
        aumenta_largura();

        // retorna false para impedir atualização da página
        return false;
    }
}
```

Capitulo 05 – JAVASCRIPT COM DOM-HTML

Comenta-se tanto sobre DOM... mas o que é DOM?

O DOM (Document Object Model) é uma interface que representa como os documentos HTML são lidos pelo seu browser. Após o browser ler seu documento HTML, ele cria um objeto que faz uma representação estruturada do seu documento e define meios de como essa estrutura pode ser acessada.

DOM auxilia na padronização dos objetos e a forma como esses objetos relacionam-se entre sí dentro de uma aplicação. O DOM é uma API (conjunto de funções, rotinas, métodos, classes e procedimentos padronizados que oferecem recursos que facilitam o processo de desenvolvimento de uma aplicação) que padroniza a estrutura de documentos HTML.

Qual a vantagem de se usar DOM?

O DOM permite criar aplicações que atualizam os dados da página sem que seja necessário uma atualização. Pode-se criar aplicações customizáveis como, mudar o design da página, arrastar, mover, excluir elementos da página.

Temos três divisões dentro do DOM:

a) DOM Core (funções gerais de uma interface – é uma Application Programming Interface(API) que define um conjunto de objetos e interfaces para manipular objetos de uma página)
b) DOM HTML (representação da estrutura de um documento HTML)
c) DOM XML (representação da estrutura de um arquivo XML)

Esse modelo foi criado visando simplificar a programação Javascript dentro do HTML.

O DOM auxilia o programador a acessar estruturas, criar, modificar, adicionar, retirar e manipular elementos e conteúdos de documentos HTML de forma mais simples e eficiente.

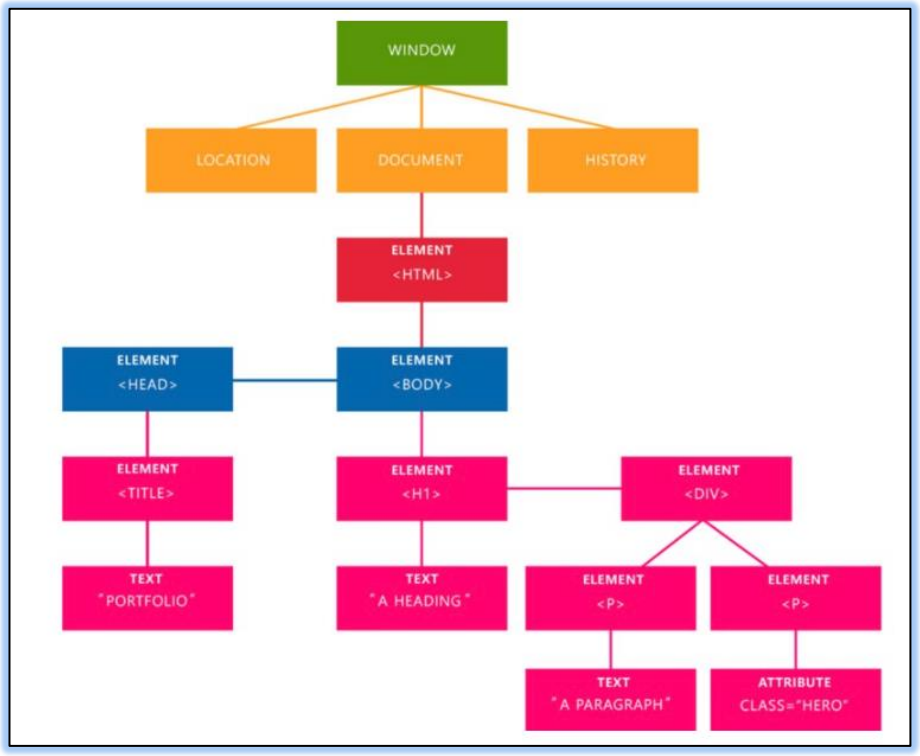
O DOM Core foi desenvolvido para ser usado em qualquer linguagem de programação.

Usa-se o DOM principalmente para atualizar uma página Web (DOM é bastante utilizado com Ajax) ou quando se quer construir uma interface de usuário avançada. Com o DOM pode-se mover itens dentro de uma página ou criar efeitos CSS bastante interessantes sem precisar nem mesmo recarregar a página.

Podemos dizer que as páginas Web são estruturadas similares a uma árvore. Isso se deve ao fato de podermos ver uma página Web como uma árvore, com uma raiz como o elemento HTML e os seus filhos como o HEAD e o BODY que por sua também possuem elementos filhos e assim sucessivamente. Os elementos que não possuem filhos são chamados de nós folhas. Por exemplo, os elementos TITLE, STYLE, SCRIPT, LI, H1, P, TD.

Essa estrutura de árvore é a forma que o navegador organiza as marcações do HTML, é dessa forma que o navegador Web enxerga um documento HTML. A leitura da árvore se dá sempre da esquerda para a direita, assim teremos a página Web original.

Abaixo, temos uma representação do DOM:



Observe que temos quatro pontos importantes:

1) **Document**: **que como o nome diz, trata-se do documento HTML;
2) **Elements**: **são todas as tags que estão no arquivo HTML – são elementos da árvore DOM.
3) **Texts**: É o conteúdo das tags.
4) **Attributes**: É a junção de todos atributos para um nó específico. No caso, o attribute class=”hero” está apontando para o elemento <p>.

Toda vez que se abre uma página no browser cria-se um objeto document que permite o acesso a todos os elementos HTML através do Javascript. As principais propriedades e métodos do objeto document são:

PROPRIEDADES e MÉTODOS	EXPLICAÇÃO
init()	Usado para carregar mais de um função javascript quando o documento (a página) estiver carregada no browser.
método getElementById(“id”)	Acesso o elemento DOM do atributo “id” retornando a referencia ao elemento indicado pelo parâmetro “id”.
propriedade style	Essa propriedade permite manipular CSS dentro do Javascript. Sintaxe: elemento.style.propriedade = “valor”. Exemplos de propriedades CSS que podem ser usadas: backgroundColor zindex textindent

	borderTopWidth paddingLeft listStyleImage
método getElementsByName(“nome”)	Acessa todos os elementos do atributo denominado pelo atributo “name” retornando um array com os demais atributos a ele associados.
método getElementsByClassName(“nome da classe”)	Acessa todos os elementos do atributo identificado pela classe atribuída ao elemento.
propriedade innerHTML	Essa propriedade permite a definição de um conteúdo HTML para um elemento do DOM.

Vamos ver os exemplos para melhor compreensão:
Exemplo com init():

```
<html><head><title>Untitled</title>
<script>
window.onload = init;
function init() {
  function Manha() {
    alert("Bom dia!"); }

  function Tarde() {
    alert("Boa Tarde!"); }

  function Noite() {
    alert("Boa Noite... bom descanso!"); }
}
  Manha();
  Tarde();
  Noite();
</script></head><body></body></html>
```

Outra construção para o mesmo exemplo:

```
<html><head><title>Untitled</title>
<script>
  function Manha() {
  alert("Bom dia!"); }

  function Tarde() {
  alert("Boa Tarde!"); }

  function Noite() {
  alert("Boa Noite... bom descanso!"); }

  function init() {
    Manha();
    Tarde();
    Noite();
  }
```

```
  window.onload = init;
</script></head><body></body></html>
```

Ainda, outra construção para o mesmo exemplo:

```
<html><head><title>Untitled</title>
<script>
  window.onload = function() {
    function Manha() {
     alert("Bom dia!"); }

    function Tarde() {
     alert("Boa Tarde!"); }

    function Noite() {
     alert("Boa Noite... bom descanso!"); }
  }
</script></head><body></body></html>
```

Exemplo com método getElementById(“id”) e propriedade style:

```
<html><head><title>Untitled</title>
<script>
  function formatacaoTitulos() {
    var titulo1 = document.getElementById("titulo1");
    titulo1.style.backgroundColor = "#ff00B0";
    var titulo2 = document.getElementById("titulo2");
    titulo2.style.border = "4px solid #bb00ff";
    var titulo3 = document.getElementById("titulo3");
    titulo3.style.fontSize = "30px";
  }
  function init() {
    formatacaoTitulos();
  }
  window.onload = init;
</script>
</head>
<body>
  <h1 id="titulo1">BEM VINDO AO CURSO DE JAVASCRIPT</h1>
  <h1 id="titulo2">Capitulo 1: Sintaxe Básica JavaScript</h1>
  <h1 id="titulo3">Capitulo 2: JavaScript e CSS</h1>
</body></html>
```

Exemplo com método getElementsByName(“nome”):

```
<html><head><title>Untitled</title>
<script>
  function formatacaoTitulos() {
    var elementos = document.getElementsByName("titulo1");
    elementos[0].style.backgroundColor = "#ff00B0";
    elementos[1].style.border = "4px solid #bb00ff";
    elementos[2].style.fontSize = "30px";
  }

  function init() {
    formatacaoTitulos();
  }

  window.onload = init;
</script></head><body>
  Nome do Curso:<input type="text" name="titulo1"><br>
  Data Inicio do Curso:<input type="text" name="titulo1"><br>
  Duração do Curso:<input type="text" name="titulo1"><br>
</body></html>
```

Exemplo com método innerHTML:

```
<html><head><title>Untitled</title>
<script>
  function formatacaoTitulos() {
    var titulo  = document.getElementById("titulo1");
    var texto   = document.getElementById("texto");
    var botao   = document.getElementById("botao");
    var elementos = titulo.innerHTML;
    botao.onclick = function()  {
      titulo.innerHTML = "Curso de JavaScript";
      titulo.style.background = "#ff00B0";
      texto.innerHTML = "<h2>Sejam Bem Vindos ao Curso!</h2>";
      texto.style.color = "#bb00ff";
      texto.style.fontSize = "30px";
      document.frmExemplo.duracao.value = "30 horas";
    }
  }
  function init() {
    formatacaoTitulos();
  }
  window.onload = init;
</script></head>
<body>
 <form name="frmExemplo" action="" method="">
  <h1 id="titulo1"></h1><br>
  Duração do Curso:<input type="text" name="duracao"><br>
  <div id="texto"></div><br>
  <button type="button" id="botao">Testando o INNERHTML</button>
 </form> </body></html>
```

Dentro do DOM, temos as coleções HTML. Vejamos algumas no quadro a seguir:

ELEMENTO	OBJETO	PROPRIEDADES e MÉTODOS	EXPLICAÇÃO
InterfaceHTMLDocument	**document**		
		propriedade title	Retorna a string contida na tag <title> da página HTML.
		propriedade URL	Retorna a URL da página HTML.
		propriedade cookie	Define ou retorna o nome e o valor de um cookie de uma página aberta no browser.
		propriedade domain	Retorna o domínio onde a página aberta no browser está hospedada.
		propriedade forms	Retorna a quantidade de formulários existentes na página.
		propriedade images	Retorna a a quantidade de imagens existentes na página aberta.
InterfaceHTMLLinkDocument	**link**		
		propriedade disabled	Aplica-se a links que apontam para CSS e podem ser habilitados ou desabilitados.
		propriedade href	Recupera a URL vinculada ao link existente na página.
		propriedade media	Recupera o caminho e nome do arquivo vinculado ao link da página.
		propriedade rel	Recupera o tipo do arquivo vinculado ao link da página.
		propriedade target	Mostra o alvo do link ao ser clicado.
InterfaceHTMLStyleDocument	**style**		
		propriedade disabled	Permite habilitar ou desabilitar o CSS associado ao objeto.
		propriedade media	Recupera o valor do atributo media da tag link.
		propriedade type	Recupera o valor do atributo type da tag link.
InterfaceHTMLFormDocument	**form**		
		propriedade action	Recupera o valor do atributo action da tag form.
		propriedade method	Recupera o valor do atributo method da tag form onde nesse caso pode ser get ou post.
		propriedade name	Recupera o valor do atributo name da tag form.
		propriedade target	Recupera o valor do atributo target da tag form.
InterfaceHTMLSelectDocument	**select**		
		propriedade length	Retorna a quantidade de elementos existentes na lista.
		propriedade type	Retorna o tipo de dados da lista.

		propriedade selectedIndex	Define o índice da opção selecionada pelo usuário, sempre começando a contagem com zero.
		propriedade disabled	Trabalha com true e false e é usado para habilitar ou desabilitar uma opção da lista.
		propriedade size	Define a quantidade de itens visíveis da lista.
InterfaceHTMLInputDocument	**input**		
		propriedade name	Retorna ou define o conteúdo do atributo name do objeto input.
		propriedade type	Retorna ou define o tipo do atributo type do objeto input.
		propriedade value	Retorna ou define o conteúdo do atributo value do objeto input.
		propriedade alt	Retorna ou define o conteúdo do atributo alt do objeto input.
		propriedade size	Retorna ou define o conteúdo do atributo size do objeto input.
		propriedade defaultChecked	Retorna ou define o conteúdo do atributo defaultChecked do objeto input.
		propriedade checked	Retorna ou define o conteúdo do atributo checked do objeto input.
		propriedade disabled	Retorna ou define o conteúdo do atributo disabled do objeto input.
		propriedade maxLength	Retorna ou define o conteúdo do atributo maxLength do objeto input.
		propriedade acessKey	Retorna ou define o conteúdo do atributo acessKey do objeto input.
		propriedade defaultValue	Retorna ou define o conteúdo do atributo defaultValue do objeto input.
		propriedade accept	Retorna ou define o conteúdo do atributo accept do objeto input.
		propriedade readOnly	Retorna ou define o conteúdo do atributo readOnly do objeto input.
		método click()	Simula o evento clique de um botão.
		método select()	Seleciona o conteúdo do objeto input (text, file e password).
		método focus()	Dá foco ao objeto input.
		método blur()	Retira o foco do objeto input.
InterfaceHTMLTextArea-Document	**textarea**		
		propriedade cols	Define a quantidade de colunas do textarea.
		propriedade defaultValue	Retorna o conteúdo padrão estipulado na criação do objeto textarea.
		propriedade disabled	Habilita ou desabilita o textarea.
		propriedade rows	Define a quantidade de linhas do textarea.
		propriedade name	Define ou restaura o nome do objeto textarea.
		propriedade value	Retorna o conteúdo existente para o objeto textarea.
		método select()	Seleciona o conteúdo do objeto

			textarea.
		método focus()	Dá foco ao objeto textarea.
		método blur()	Retira o foco do objeto textarea.
InterfaceHTMLButtonDocument	**button**		
		propriedade disabled	Habilita ou desabilita o botão.
		propriedade name	Retorna o nome dado na criação do botão
		propriedade value	Retorna o conteúdo do atributo value do botão
InterfaceHTMLFieldsetDocument	**fieldset**		
		propriedade fieldset	Retorna a referencia ao objeto form que contem o objeto fieldset.
InterfaceHTMLLabelDocument	**label**		
		propriedade accesskey	Retorna ou define o atributo accesskey do objeto.
		propriedade form	Retorna a referência ao objeto form onde o objeto label foi declarado.
		propriedade htmlFor	Retorna uma string associando o rótulo a um controle com o atriuto id igual à string.
InterfaceHTMLImageDocument	**img**		
		propriedade height	Define a altura da imagem a ser exibida na página.
		propriedade isMap	Define um possível mapeamento feito na imagem.
		propriedade name	Retorna ou define o nome do objeto imagem.
		propriedade src	Associa o arquivo a imagem.
		propriedade width	Define a largura da imagem a ser exibida na página.
InterfaceHTMLObjectDocument	**object**		
		propriedade data	Define ou retorna o conteúdo do atributo data do objeto.
		propriedade declare	Define ou retorna o conteúdo do atributo declare do objeto.
		propriedade form	Define ou retorna o conteúdo do atributo form do objeto.
		propriedade name	Define ou retorna o conteúdo do atributo name do objeto.
		propriedade tabIndex	Define ou retorna o conteúdo do atributo tabIndex do objeto.
		propriedade width	Define ou retorna o conteúdo do atributo width do objeto.
		propriedade type	Define ou retorna o conteúdo do atributo type do objeto.
InterfaceHTMParamDocument	**param**		
		propriedade name	Define ou retorna o parâmetro name para um objeto genérico.
		propriedade type	Define ou retorna o parâmetro type para um objeto genérico.
		propriedade value	Define ou retorna o parâmetro value para um objeto genérico.

		propriedade valueType	Define ou retorna o parâmetro valueType (tipo do dado) para um objeto genérico.
InterfaceHTMLTableDocument	**table**		
		propriedade border	Retorna ou define o conteúdo para o atributo border da tabela.
		propriedade cellPadding	Retorna ou define o conteúdo para o atributo cellPadding da tabela.
		Propriedade cellSpacing	Retorna ou define o conteúdo para o atributo cellSpacing da tabela.
		propriedade rows	Retorna o conteúdo das linhas da tabela.
		propriedade tBodies	Retorna o conteúdo dos elementos tbody da tabela.
		propriedade tFoot	Retorna o conteúdo dos elementos tfoot da tabela.
		propriedade tHead	Retorna o conteúdo dos elementos thead da tabela.
		propriedade width	Define a largura da tabela na página.
		método createCaption()	Cria a seção de título de uma tabela.
		método createTfoot()	Cria a seção de rodapé de uma tabela.
		método createThead()	Cria a seção de cabeçalho de uma tabela.
		método deleteCaption()	Exclui o cabeçalho de uma tabela.
		método deleteThead()	Cria a seção de título de uma tabela.
		método deleteRow()	Exclui a seção de cabeçalho de uma tabela.
		método insertRow()	Cria uma linha na tabela (tr).

Exemplo com o objeto document:

```
<html><head><title>Untitled</title>
<script>
  function dados() {
    alert("Qtde de formulários: " + document.forms.length);
  }

  function init() {
    dados();
  }
  window.onload = init;
</script></head>
<body>
 <form name="frmExemplo1" action="" method="post">
  Nome Completo: <input type="text" name="txt_nome"><br>
 </form>
 <form name="frmExemplo2" action="" method="get">
  Endereço Completo: <input type="text" name="txt_endereco"><br>
 </form>
</body></html>
```

Exemplo com o objeto link:

```
<html><head><title>Untitled</title>
<link href="www.google.com.br" rel="qq coisa" type="text/link" media="todas"></link>
<script>
  function dados() {
    var elemento;
    var mensagem;
    var botao;
    elemento = document.getElementsByTagName("link");
    botao = document.getElementById("botao");
    mensagem = "Exibindo as propriedades do objeto LINK: ";
    mensagem += "href=" + elemento[0].href;
    mensagem += "rel=" + elemento[0].rel;
    mensagem += "type=" + elemento[0].type;
    mensagem += "media=" + elemento[0].media;
    botao.onclick = function() {
      elemento[0].disabled = true;
      alert(mensagem);
    }
  }
  function init() {
    dados();
  }
    window.onload = init;
</script>
</head>
<body>
 <form name="frmExemplo1" action="" method="">
 <button type="button" id="botao">Testando o objeto LINK</button>
 </form></body></html>
```

Exemplo com o objeto style:

```
<html><head><title>Untitled</title>
<style type="text/css" media="teste">
body  {
  font  : 20px Tahoma;
  color : "#ff00B0";
}

h1  {
  font  : 30px Arial;
  color : "#bb00ff";
  style : bold;
}
</style>

<script>
  function dados() {
    var frase = document.getElementById("frase");
    frase.style.backgroundColor = "#ff00B0";
    var elemento = document.getElementsByTagName("style");
    alert("Type=" + elemento[0].type);
    alert("Media=" + elemento[0].media);
    elemento[0].disabled = true;
  }

  function init() {
    dados();
  }

  window.onload = init;
</script></head>
<body>
 <form name="frmExemplo1" action="" method="">
  <div id="frase">Testando o objeto STYLE</div><br>
 </form> </body></html>
```

Exemplo com o objeto form:

```
<html><head><title>Untitled</title>
<link href="www.google.com.br" rel="qq coisa" type="text/link" media="todas"></link>
<script>
  function dados() {
    var elemento;
    var botaoReset;
    var botaoSubmit;
    elemento    = document.getElementsByTagName("form").formExemplo;
    botaoReset  = document.getElementById("btnReset");
    botaoSubmit = document.getElementById("btnSubmit");
    botaoReset.onclick = function() {
      alert("clicou no RESET");  }

    botaoSubmit.onclick = function() {
      alert("clicou no SUBMIT"); }
  }

  function init() {
    dados();
  }
  window.onload = init;
</script></head>
<body>
 <form name="frmExemplo" action="" method="">
 <label for="nome">Login</label>
 <input type="text" name="txtLogin" id="txtLogin"><br>
 <label for="senha">Senha</label>
 <input type="password" name="txtSenha" id="txtSenha"><br>
 <br><input type="reset" id="btnReset" value="Limpar Campos"><br>
 <br><input type="submit" id="btnSubmit" value="Enviar Dados"><br>
 </form></body></html>
```

Exemplo com o objeto input:

```
<html><head><title>Untitled</title>
<link href="www.google.com.br" rel="qq coisa" type="text/link" media="todas"></link>
<script>
  function dados() {
    var elemento = document.getElementsByTagName("input");
    var botao    = document.getElementById("btn");
    var msg      = document.getElementById("msg");

    var mensagem = "<br>Exibindo as propriedades do objeto INPUT: <br><br>";
    mensagem += "Qtde=" + elemento.length + "<br>";
    mensagem += " Type=" + elemento[0].type + "<br>";
    mensagem += " Tabindex da opção VOLEIBOL =" + elemento["ckVol"].tabIndex;
    botao.onclick = function() {
      msg.innerHTML = mensagem;
    }
  }

  function init() {
    dados();
  }

  window.onload = init;
</script>
</head>
<body>
 <form name="frmExemplo" id="frmExemplo" action="" method="">
   <legend><b>Cadastro de Clientes</b></legend><br>
   <label for="CPF">CPF</label>
   <input type="text" name="txtCPF" id="txtCPF"><br>
   <label for="nome">Nome</label>
   <input type="text" name="txtNome" id="txtNome"><br>
   <label for="senha">Senha</label>
   <input type="password" name="txtSenha" id="txtSenha"><br>
   Esporte Preferido:<br>
   <input class="input" type="checkbox" name="ckFut" id="ckFut" value="ckFut"
      checked="checked">
   <label for="ckFut">Futebol</label>
   <input class="input" type="checkbox" name="ckVol" id="ckVol" value="ckVol">
   <label for="ckVol">Voleibol</label>
   <input class="input" type="checkbox" name="ckBas" id="ckBas" value="ckBas">
   <label for="ckBas">Basquete</label>
   <br><button type="button" id="btn">Testando...</button>
 </fieldset>
 <br><br>
 <div id="msg"></div>
 </form></body></html>
```

Exemplo com o objeto table:

```
<html><head><title>Untitled</title>
<script>
 function dados() {
   var elemento = document.getElementById("tabela");
   var botaoRC  = document.getElementById("btn1");
   var botaoIC  = document.getElementById("btn2");
   var botaoRL  = document.getElementById("btn3");
   var botaoIL  = document.getElementById("btn4");

   botaoRC.onclick = function() {
     elemento.deleteCaption();
   }

   botaoIC.onclick = function() {
     elemento.createCaption().innerHTML = "******* COLOCANDO OUTRO
      CABEÇALHO *******";
   }

   botaoRL.onclick = function() {
     elemento.deleteRow(-1);
   }

   botaoIL.onclick = function() {
     var linha = elemento.insertRow(4);
     for (var i=0;i<3;i++)  {
       var celula = linha.insertCell(i);
       celula.innerHTML = "inserida-" + i;
     }
   }
 }

 function init() {
   dados();
 }

 window.onload = init;
</script>
</head>
<body>
 <form name="frmExemplo" id="frmExemplo" action="" method="">
   <table width="400" border="1" cellspacing="10" cellppading="5" id="tabela">
   <caption align="top">Exemplo de TABELA</caption><br><br>
   <thead>
     <tr>
       <th scope="col">MATRICULA</th>
       <th scope="col">NOME</th>
       <th scope="col">EMAIL</th>
     </tr>
```

```
  </thead>
  <tfoot>
    <tr>
      <td colspan="3">Dados do sistema XPTO</td>
    </tr>
  </tfoot>

  <tbody>
    <tr>
      <td>234998</td>
      <td>Anderson Silva</td>
      <td>andsilva@unicamp.br</td>
    </tr>
    <tr>
      <td>235667</td>
      <td>Bruna Garcia</td>
      <td>brungardiac@unicamp.br</td>
    </tr>
    <tr>
      <td>237883</td>
      <td>Juca de Oliveira</td>
      <td>juoliveira@unicamp.br</td>
    </tr>
  </tbody>
  </table>
  <br>
  <br><button type="button" id="btn1">Remover Cabeçalho</button>
  <button type="button" id="btn2">Inserir Cabeçalho</button>
  <button type="button" id="btn3">Remover Linha</button>
  <button type="button" id="btn4">Inserir Linha</button>
 <br><br>
 </form>
</body></html>
```