Desenvolvimento de software

Aula 05: Interfaces Visuais para Web

Apresentação

A Internet nem sempre foi tão rápida e com visual tão avançado, mas as necessidades comerciais trazidas pelo usuário e comprador geraram uma oportunidade para a franca evolução das interfaces, incorporando conceitos como usabilidade, acessibilidade e até mesmo responsividade, devido ao advento das plataformas móveis.

Ao criarmos nossas interfaces Web deveremos estar atentos aos padrões da W3C, de forma a utilizar tecnologias como HTML, JavaScript e CSS da melhor forma, sempre buscando criar interfaces que sejam ao mesmo tempo robustas e intuitivas, buscando satisfazer os mais diferentes perfis de usuários.

Objetivos

- Identificar os componentes de uma interface Web;
- Explicar o uso de tecnologias cliente (HTML/CSS/JavaScript);
- Empregar as tecnologias cliente para a criação de interfaces Web.

Linguagem HTML 5

1991

A linguagem HTML, do inglês Hypertext Markup Language, foi formalizada em 1991, baseada nas propostas de Tim Berners-Lee, físico britânico que buscava um meio de divulgação de suas pesquisas entre seus colegas.

1999

Inicialmente controlada pela IETF (Internet Engineering Task Force), passou a ter suas especificações mantidas pela W3C (World Wide Web Consortium) a partir de 1996, sendo a recomendação HTML 4.01 publicada em 1999.

2000

A partir de 2000, a W3C passa a recomendar a sua nova especificação XHTML, a qual traz uma sintaxe mais rígida para o HTML, baseando-se nas regras do XML (Extended Markup Language).

2008

Finalmente, a partir da primeira publicação, em 2008, pela W3C, surge a especificação do HTML 5, oficializada em 2014, cuja característica principal é um novo mecanismo de extensibilidade que permite combinar diferentes sintaxes baseadas em XML dentro de uma mesma página.

Como nas versões iniciais, o HTML 5 é uma linguagem de marcação baseada em etiquetas (tags) que devem controlar diversos aspectos do texto. Nas versões iniciais do HTML ocorria o controle tipográfico e estrutural, mas no HTML 5 a preocupação é apenas estrutural, sendo delegado ao CSS (Cascade Style Sheet) o controle de características tipográficas.

Sintaxe do HTML 5

Como o HTML na versão 5 segue o padrão de escrita do XML, a primeira linha deve definir o tipo de documento.

Em seguida, definimos o documento HTML, dividindo-o nas seções de **cabeçalho e corpo**.

No cabeçalho ficam informações de configuração e pré-carga, enquanto no corpo será colocada a parte visual da página.

Sempre devemos nos preocupar com a documentação correta de nosso código, e o uso de comentários ajuda muito na compreensão do que foi criado.

Comentário

Em termos de HTML, os comentários são iniciados com um símbolo de exclamação seguido de dois traços, sendo terminados com dois traços, conforme podemos observar no fragmento de código anterior.

Incialmente, vamos nos preocupar com o tipo de informação que pode ser inserida na seção de cabeçalho, como o título da página e a definição dos caracteres e acentuações adotados.

Nesse fragmento de código, podemos observar a inserção do título de uma página com o uso de **<title>** e a escolha da acentuação utilizada através de **<meta/>**.

Diversas tags aceitam a definição de atributos específicos, como **<meta charset="UTF-8"/>**, e tais atributos devem ter seus valores inseridos entre aspas ou apóstrofes. Essa é apenas uma das diversas características controladas com o uso de **<meta/>**.

As tags mais comuns na seção de cabeçalho são resumidas a seguir.

Tag	Utilização
<title>	Define o título da página, normalmente apresentado no topo da janela.
<meta>	Permite a inclusão de informações e controle de características do documento, como o uso de UTF-8 para acentuação da língua portuguesa.
<link>	Efetua a ligação da página com um recurso externo, como folha de estilo CSS ou biblioteca JavaScript.
<style>	Define uma folha de estilo interna para a página.

Na parte do corpo ficam os elementos estruturais constituintes da parte visual. É onde será definido o conteúdo, que pode ser estruturado em termos de títulos, parágrafos, listas, tabelas e diversos outros elementos que promoverão a organização visual do texto da página.

Nesse fragmento de código, referente ao corpo da página, podemos observar a tag **<p>** para a definição de um parágrafo, e a construção de uma lista com **<ul>** e **<li>**.

A lista é iniciada e finalizada com **<ul>** e cada elemento é inserido em um conjunto **<li>**.

Essas tags criarão um efeito visual como o que pode ser observado a seguir:

A sintaxe HTML traz diversas tags estruturantes para o conteúdo do **body**, até mesmo por ser uma linguagem de hipermídia, onde é comum a criação de listas, camadas e até tabelas (estas últimas apenas para dados tabulares, segundo recomendações atuais).

Podemos observar algumas das tags estruturais a seguir:

Tag	Utilização
<h1> <h2> <h3> <h4> <h5> <h6> <h7>	Definem elementos de texto utilizados como títulos em diferentes tamanhos de fonte, sendo **<h1>** para o maior e **<h7>** para o menor.
**<b> <i> <u> **	Negrito, itálico e outros efeitos de texto. São consideradas obsoletas, sendo recomendado o uso de folhas de estilo CSS.
**<p> **	Definem trechos de texto, sendo que **<p>** ressalta o parágrafo e ** ** é muito utilizado para a criação de camadas.
** **	Iniciam a criação de listas de elementos, sendo numeradas com ** **.
** **	Define um item de lista.
** **	Quebra de linha.
<table> <tr> <td>	Utilizadas para a criação de tabelas. Hoje em dia seu uso é recomendado apenas para dados tabulares.
****	Define um trecho avulso, ao qual podem ser aplicadas formatações.

Uma funcionalidade essencial das páginas HTML é a possibilidade de **navegar** entre páginas de um site ou entre sites, permitindo acessar os mais diversos recursos ao longo da Web, identificados de forma única através de sua **URL** (Universal Resource Locator).

Criaremos hiperlinks para efetuar essa navegação, sendo utilizada a tag **<a>**, ou âncora (do inglês **anchor**), sempre apontando para alguma **URL** e apresentando um texto que representa o recurso. Quando o usuário clica sobre o texto é redirecionado para o recurso.

Com o uso conjunto de todos esses elementos, seremos capazes de criar nossas páginas, podendo tratar de simples páginas, ainda sem elementos dinâmicos e formatações adequadas, mas com toda a parte estrutural definida.

Podemos observar uma página completa a seguir:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Página de Exemplo </title>
      <meta charset="UTF-8"/>
   </head>
   <body>
      <h1>Alguns links úteis <h1>
      <ul>
         <l1> <a href="//www.mpf.mp.br/">Ministério Público Federal</a> </l1>
         <l1> <a href="//www.pf.gov.br/">Polícia Federal</a> </l1>
         <l1> <a href="//portal.stf.jus.br">Superior Tribunal Federal</a> </l1>
      <ul>
   </body>
</html>
<div>
```

Temos uma lista onde cada item corresponde a determinado hiperlink para algum site, além de trazer um título e uma definição para acentuação.

Como citado anteriormente, precisamos utilizar as tags <ul> e <l1> em conjunto, sendo que a primeira determina o início e final da lista, enquanto a segunda determina os itens que farão parte dessa lista.

CSS

A metodologia de criação de sites atualmente aceita, prioriza a divisão da página em termos de estrutura e formatação, ficando a estrutura a cargo do HTML e a formatação efetuada via folhas de estilo **CSS** (Cascade Style Sheets).

Com essa regra definida, nosso próximo passo é configurar os elementos tipográficos e o aspecto visual da página com o uso do CSS.

Vamos modificar o exemplo anterior, aplicando uma formatação através de folhas de estilo:

Exemplo 3

```
<!DOCTYPE html>
<html>
   <head>
      <title>Página de Exemplo </title>
      <meta charset="UTF-8"/>
      <style>
         h1 {color: orange}
         a {color: yellow}
         a:hover {color: red}
         body {background-color: #000032; color:yellow}
      </style>
   </head>
   <body>
      <h1>Alguns links úteis<h1>
      <ul>
         <l1> <a href="//www.mpf.mp.br/">Ministério Público Federal</a> </l1>
         <l1> <a href="//www.pf.gov.br/">Polícia Federal</a> </l1>
         <l1> <a href="//portal.stf.jus.br">Superior Tribunal Federal</a> </l1>
      <ul>
   </body>
</html>
```

Observe como o acréscimo da tag **<style>** com as formatações adequadas trouxe enorme diferença para o aspecto visual da página.

A sintaxe geral do CSS pode ser definida como:

Iniciamos com o seletor e, entre as chaves, definimos os vários elementos de formatação como fontes e cores, alinhamentos, figuras de fundo, entre vários outros, sempre seguindo a sintaxe **propriedade: valor.**

No exemplo anterior, a tag **<h1>** foi formatada com a fonte de cor laranja, enquanto **<body>** recebe fundo de tonalidade azul (RGB=#000032) e fonte amarela.

O conjunto de seletores **a** e **a:hover** traz um comportamento bem interessante. O primeiro define a formatação do link no modo normal e o segundo define a formatação quando o mouse passa sobre o mesmo.

Atenção

Além de formatar as tags, é possível utilizar classes e identificadores como seletores do CSS, bastando colocar o sinal de hash na frente dos identificadores e ponto na frente das classes.

A questão maior é quando devemos utilizar identificadores ou classes. Os identificadores são utilizados em elementos que não se repetem, como camadas, e as classes são utilizadas para utilizar a mesma formatação em diversos pontos da página.

Podemos observar esses seletores mais adiante:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Exemplo de Camadas</title>
      <meta charset="UTF-8"/>
      <style>
         #C1 {position:absolute; top:10px; left:10px; width:300px; height:200px; background-color:lightblue}
         #C2 {position:absolute; top:50px; left:50px; width:220px; height:120px; background-color:blue; color:yellow}
         .destaque {background-color:black; color:orange}
      </style>
   </head>
   <body>
      <div id="C1">Camada 1 com algum <span class="destaque">texto ressaltado</span> </div>
      <div id="C2">Camada 2, e agora o<span class="destaque">texto ressaltado</span> está aqui </div>
   </body>
</html>
```

Aqui nós utilizamos o CSS para definir o posicionamento, o tamanho e as cores de duas camadas, identificadas como C1 e C2 a partir do atributo **id** da ** **, e com o uso de **** e o atributo **class** aplicamos a formatação específica para alguns fragmentos de texto.

JavaScript

A linguagem JavaScript nem sempre teve esse nome. Desenvolvida originalmente pela Netscape, ela se chamava Mocha, e teve o nome modificado posteriormente para LiveScript, quando ocorreu seu lançamento junto ao navegador Netscape 2.0 versão beta, em setembro de 1995.

Em dezembro de 1995, em um anúncio conjunto com a Sun Microsystems, é lançado o Netscape 2.0B3, com suporte à tecnologia de Applets. O nome da linguagem foi modificado para JavaScript, o que causa muita confusão até hoje com relação à sua relação com a linguagem Java.

É importante que tenhamos em mente que JavaScript não é Java, e a similaridade entre as duas linguagens ocorre apenas pelo fato de ambas utilizarem sintaxes baseadas na linguagem C.

Utilizamos o JavaScript para controle de elementos da página HTML e viabilização da interatividade, caracterizando-se originalmente como uma tecnologia cliente. Pode ser embebida na própria página, ou organizada no formato de biblioteca, como arquivo externo.

Atenção

Recentemente, o JavaScript passou a ser utilizado também do lado servidor, através de tecnologias como o node.js.

Outro ambiente que recebeu a possibilidade de desenvolvimento com JavaScript foi o dos dispositivos móveis, com uso de ferramentas como Ionic.

Sintaxe do JavaScript

Inicialmente, devemos compreender a declaração de variáveis e operações aritméticas da linguagem.

Uma variável pode ser declarada com o uso de var, ou simplesmente com sua inicialização. Como o JavaScript não é fortemente tipado, a variável assume o tipo do valor associado a ela.

Os tipos com os quais a linguagem trabalha são: inteiro, ponto flutuante, booleano, texto, objeto e vetor (como objeto da classe Array).

Os operadores matemáticos são os mesmos utilizados pelo Java; afinal, ambas as linguagens derivaram do C, mas com pequenas diferenças: enquanto no Java a divisão entre inteiros resulta em um número inteiro, no JavaScript não temos como diferenciar um inteiro de um real, e por isso a divisão poderá retornar um número real.

Exemplo 5

```
<!DOCTYPE html>
<html>
<script>
   var a = 5, b = 32, c = 7;
   document.writeln("<br/>A:"+a+" B:"+b+" C:"+c);
   b = b - (c * 2);
   b /= a;
   document.writeln("<br/>A:"+a+" B:"+b+" C:"+c);
   b++;
   document.writeln("<br/>A:"+a+" B:"+b+" C:"+c);
<script>
</body>
</html>
```

Aqui são declaradas três variáveis com o uso de var, sendo efetuadas algumas operações aritméticas sobre as mesmas.

A cada passo é impresso na página o valor das variáveis através de **document.writeln**. É interessante observar que esse comando escreve sobre a página HTML, permitindo a inclusão de tags, como o uso de **
** para quebra de linha.

Se estivéssemos efetuando essas mesmas operações em Java, para variáveis inteiras, o resultado de B na segunda linha seria 3, e não 3.6, como ocorre no JavaScript.

Assim como os operadores aritméticos do JavaScript são similares aos do Java, os relacionais e lógicos também são, e podemos utilizá-los da mesma forma nas duas linguagens, apenas observando o fato de que o JavaScript não é fortemente tipado.

Uma observação a ser feita é que a comparação entre Strings é muito diferente entre as duas linguagens. Enquanto Java necessita do uso de equals, o JavaScript utiliza o operador para comparação de igualdade.

Java	JavaScript

String a;
boolean b;
 	
// Operações sobre a
 	
b = (a.equals("XPTO"));
	var a,b;
	
	// Operações sobre a
	
	b = (a == "XPTO");

Estruturas de decisão e de repetição

A similaridade entre Java e JavaScript não fica apenas na definição dos operadores, mas também na utilização da mesma sintaxe para as estruturas de decisão e de repetição.

É claro que os métodos de entrada e saída de dados são completamente diferentes, até mesmo pela finalidade de cada uma das linguagens. No caso do JavaScript, toda entrada e saída deve ocorrer a partir do navegador.

Vamos observar um exemplo com uso de if:

Exemplo 6

```
<!DOCTYPE html>
<html>
<body>
   <script>
      var x = eval(prompt("Entre com o valor:",""));

      if(x > 5)
         document.writeln("<h1>Valor maior que 5</h1>");
      else
         document.writeln("<h1>Valor menor ou igual a 5</h1>"

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

A função **prompt** serve para solicitar ao usuário a digitação de um valor a partir de um diálogo de entrada, enquanto a função **eval** efetua a conversão de String para número.

Em nosso exemplo, a variável **x** recebe o valor digitado pelo usuário com o uso da função prompt, mas com o valor, que era originalmente texto, convertido para número com o uso de eval.

De acordo com o valor digitado a página irá apresentar uma das alternativas de frase, ambas formatadas pela tag **<h1>**, sendo a escolha da frase a ser apresentada controlada pela estrutura **if..else**, segundo a condição **x < 5**.

Agora temos um exemplo com uso de **for**, que imprime os números primos de 1 a 50:

Exemplo 7

```
<!DOCTYPE html>
<html>
   <body>
      <ul>
      <script>

      for(i=1 ; i<=50 ; i++){
         primo = true;
         for(j=2 ; j<i ; j++)
            if( i%j==0 ){
               primo = false;
               break;
            }
         if(primo
            document.writeln("<li>"+i+"</li>");
      }

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

Observe que cada frase é iniciada com **
** e terminada com **
, o que faz com que sejam transformadas em uma lista iniciada imediatamente antes do script com a tag **<ul>.

Nesse código, utilizamos um **for** executando de 1 a 50, e dentro utilizamos outro **for** para dizer se é primo, segundo a regra de que se i for divisível por algum número entre 2 e i-1 ele não é primo.

Tendo verificado que o número não é primo, não é necessário testar outras divisões, o que justifica o uso de **break** para sair do loop interno.

Finalmente, um exemplo com utilização da estrutura **while**:

Exemplo 8

```
<!DOCTYPE html>
<html>
   <body>
      <table border="1" width="100%">
         <tr>
            <td>
               A
            </td>
            <td>
               Fatorial(A)
            </td>
         </tr>
      <script>

         var a = 1;
         while(a<10){
            document.writeln("<tr><td>"+a+"</td>");
            fat = b = a;
            while(--b>=1){
               fat *= b;
            document.writeln("<td>"+a+"</td></tr>");
            a++;
          }
      </script>
      </table>
   </body>
</html>
```

Aqui nós imprimimos o fatorial de cada número entre 1 e 9. Para tal, utilizamos um controle **while** externo, aumentando o valor de a enquanto não atingir o valor 10, e internamente efetuamos o cálculo do fatorial com outro bloco **while**.

Nas linhas acima, podemos observar que os valores de

fat

e
b

são ambos inicializados com o valor atual de
a

. Em seguida ocorre um loop onde
b

é decrementado e já testado se o valor é maior que
1

; se for maior, o valor de
fat

é multiplicado por
b

naquela rodada.

Atenção

Esse processo apenas segue a forma natural de cálculo de fatorial. Se **a** vale 4, então 4! = 4 * 3 * 2 * 1. Se observarmos a expressão, podemos dizer que começa com o valor de a (4) e multiplica por todos os menores que 4 até chegar a 1.

Funções e eventos

As funções, assim como os métodos em ambientes orientados a objetos, correspondem a ações ou processos, que são nomeados, e que podem ou não receber parâmetros e retornar valores.

Nós criamos funções no JavaScript com o uso de **function**, os parâmetros não são tipados, e elas podem retornar um resultado para o chamador com uso de **return**.

Quanto ao conceito de eventos, é o mesmo utilizado nos ambientes desktop, ou seja, um conjunto de ações predeterminadas para as quais podemos dar uma resposta programada ao chamador.

Nesse caso, o modelo de resposta parte dos atributos de evento existentes nas tags HTML sendo relacionados com funções em JavaScript. Esse modelo é bem simples, e permite grande flexibilidade.

Observe este exemplo de tratamento de eventos:

Exemplo 9

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8"/>
   </head>
   <body>
      <button onclick="mostraMensagem()>CLique Aqui <button>
      <script>
         function mostraMensagem(){
            alert("OLA MUNDO");
         }
      </script>
   </body>
</html>
```

A página contém apenas um botão. Ao clicar sobre ele, a mensagem “OLA MUNDO” aparecerá em um **alert**.

Podemos observar a associação do evento de clique com a função de tratamento na seguinte linha:

Formulários

Quando efetuamos o cadastro em algum site de compras pela Web, o que estamos efetuando é o preenchimento de um formulário HTML. Essas informações são enviadas para o servidor de duas formas, denominadas “métodos”:

GET	Os dados são enviados através da própria URL.
POST	Os dados são enviados em background.

Atenção

Todo formulário é iniciado com a tag <form>, que terá como principais atributos o método de envio (**method**) e o destino para a informação (**action**).

Dentro do formulário, podemos incluir os componentes de entrada de dados, como caixas de texto, botões de rádio, caixas de marcação, entre outros. A grande maioria é definida através da tag <input>, cujos atributos principais são type, referente ao tipo de componente, name, utilizado para identificação do dado na chegada ao servidor, e value, para identificar o valor preenchido.

Os tipos que a tag <input> pode utilizar são explicados na tabela seguinte.

Tipo	Utilização
text	Cria uma caixa de texto, podendo ter um valor inicial definido através de **value**.
hidden	Quando queremos guardar um valor para envio, mas sem ficar visível.
radio	Opções mutuamente exclusivas. Quando marcamos um, todos com o mesmo atributo **name** serão desmarcados. O que ficar marcado, ao final, enviará seu atributo **value** como dado para o servidor.
checkbox	Quando queremos a possibilidade de marcar várias opções. Todos que forem marcados enviarão seus atributos **value** como dados para o servidor.
submit	Cria um botão de envio para o formulário.
reset	Cria um botão que reinicia (limpa) o formulário.

Podemos observar, a seguir, um formulário simples, com o uso das tags **<form>** e **<input>**.

Exemplo 10

```
<!DOCTYPE html>
<html>
   <head>
      <title>Exemplo de Formulário</title>
      <meta charset="UTF-8"/>
      <style>
         body {background-color:blue; color:white; font-style: bold}
      </style>
   </head>
   <body>
      <form action="//servidor/app.asmx" method="get">
         Nome:<input name="N1" type="text"/><br/><br/>
         Opções de Envio: <br/>
         <input type="checkbox" name="OPT" value="Revista"/>
         Revista Mensal
         <input type="checkbox" name="OPT" value="Digital"/>
         Versão Digital
         <input type="checkbox" name="OPT" value="SMS"/>
         Mensagens SMS<br/><br/>
         <input type="submit" value="Cadastrar"/>
      </form>
   </body>
</html>
```

Nesse exemplo, se escrevermos o nome XPTO e selecionarmos as opções “Versão Digital” e “Mensagem SMS”, ao clicarmos em Cadastrar verificaremos a seguinte URL na barra de navegação:

//servidor/app.asmx?N1=XPTO&OPT=Digital&OPT=SMS

O trecho ressaltado após a **interrogação** corresponde ao que chamamos de **Query String**, onde estarão os dados, no formato **nome=valor**, separados pelo sinal **&**.

Caso modificássemos o método para **“post”**, a informação enviada não ficaria visível para o usuário, mas vale ressaltar que estaria transitando em um pacote sem criptografia na rede para o caso do **HTTP** comum.

Validação de formulários

Após entender como criar os formulários, devemos aprender a efetuar a validação dos dados preenchidos.

Essa validação será feita através de **eventos** e métodos de acesso aos elementos HTML do JavaScript, e a validação dependerá bastante dos tipos de componente e informação considerados.

A forma mais aceita para o tratamento de elementos do HTML a partir do JavaScript é através do uso de identificadores nas tags HTML, com instruções getElementById, as quais seguem o formato abaixo.

Quanto aos eventos, podemos observar alguns, no quadro seguinte, com suas respectivas aplicações no processo de validação.

Evento	Aplicação
onsubmit	Efetua a validação do formulário imediatamente antes do envio para o servidor. Necessita o retorno booleano, indicando se os valores podem ser enviados ou não.
onclick	Normalmente uma chamada explícita de validação. Bastante utilizado em botões de rádio e caixas de marcação.
onchange	Ocorre quando o valor (**value**) sofre uma alteração.
onfocus	Ocorre quando o componente ganha o foco. Pode ser utilizado, por exemplo, para apagar o valor do campo.
onblur	Ocorre na perda do foco pelo componente. É comum a aplicação de máscaras em valores numéricos como CEP e CPF.
onsearch	Este evento é iniciado quando um usuário digita algo em um campo de pesquisa **(type="search")**.
onselect	Utilizado quando algum texto é selecionado no campo.

Exemplo 11

```
<!DOCTYPE html>
<html>
   <head>
      <title>Exemplo de Formulário</title>
      <meta charset="UTF-8"/>
      <style>
         body {background-color:blue; color:white; font-style: bold}
      </style>
   </head>
   <body>
      <form action="//servidor/app.asmx" method="get">
         Nome:<input name="N1" type="text"/><br/><br/>
         Opções de Envio: <br/>
         <input type="checkbox" name="OPT" value="Revista"/>
         Revista Mensal
         <input type="checkbox" name="OPT" value="Digital"/>
         Versão Digital
         <input type="checkbox" name="OPT" value="SMS"/>
         Mensagens SMS<br/><br/>
         <input type="submit" value="Cadastrar"/>
      </form>
      <script>
var objNome = document.getElementById("N1");
var objOpt1 = document.getElementById("OPT1");
var objOpt2 = document.getElementById("OPT2");
var objOpt3 = document.getElementById("OPT3");
function validar(){
   if(objNome.value==""){
      alert("Nome deve ser preenchido");
      objNome.focus();
      return false;
   }
   if(!objOpt1.checked && !objOpt2.checked &&
      !objOpt3.checked){
      alert("Selecione ao menos uma opção de envio");
      return false;
   }
return false;
      }
      </script>
   </body>
</html>
```

Devemos observar o formato da função de validação, a qual deverá retornar **true** ou **false** para o evento **onsubmit**, de forma a permitir ou não o envio da informação para o servidor. Por essa razão, a chamada desse evento é um pouco diferente dos outros.

No código JavaScript devemos estar atentos à forma como se relaciona com o HTML, capturando os componentes a ser validados nas variáveis objNome, objOpt1, objOpt2 e objOpt3, com o uso de **getElementById**, o qual recebe como parâmetro o **id** equivalente na tag da página.

Capturados nessas variáveis, podemos manusear as propriedades de cada componente em meio à função **validar**, relacionada com a submissão do formulário através do evento **onsubmit**.

Caso o nome não seja preenchido, a mensagem “Nome é obrigatório” é apresentada e o foco é direcionado para a caixa de texto referente a esse dado, através do método **focus()**.

Para o teste dos componentes do tipo checkbox, devemos verificar se nenhum deles foi marcado, e para isso utilizamos suas propriedades checked. Lembrando que, pelo fato de a propriedade ser booleana, a negação será equivalente à comparação com false.

Ao final de cada comando **if** ocorre uma instrução **return false**, a qual sai imediatamente da função, retornando o valor false para **onsubmit**, e impedindo o envio. Apenas se ambos os testes forem falsos irá ocorrer **return true**.

O evento onsubmit é destinado a uma validação global antes do envio, mas não é a única opção do modelo de validação. Outras validações e formatações podem ser efetuadas no momento da perda do foco pela caixa de texto, ou quando selecionamos o elemento de uma lista de valores, entre diversas outras opções.

Atividade

1. Quando construímos uma interface para Web devemos utilizar diversas tecnologias cliente, voltadas para a estruturação, formatação e dinamização do conteúdo da página. Qual seria a tecnologia voltada para a estruturação da página?

a) HTML
b) CSS
c) JavaScript
d) XML
e) Java

2. O uso de CSS é baseado em elementos denominados seletores. Considerando o fragmento de código abaixo, quais seriam os seletores corretos, respectivamente, para formatar o cabeçalho com fonte azul e apenas a primeira div com fundo amarelo?

<h1> Cabeçalho </h1>
Div1 </form>
Div2 </form>

a) h1, div
b) .Cabeçalho, .Div1
c) .h1, .primeira
d) h1, #Div1
e) h1, #primeira

---

3. Quando criamos um sistema de validação para nossos formulários, além de fornecer uma interface mais robusta, diminuímos o número de chamadas inválidas ao servidor e, consequentemente, o fluxo de rede.

Os formulários HTML são validados pelo JavaScript a partir de eventos disponíveis para as tags e, em uma situação em que desejemos validar o preenchimento do CPF na saída de um campo texto, devemos utilizar:

a) onsubmit
b) onclick
c) onchange
d) onblur
e) Onfocus

Referências

CASSATI, J.

Notas

Programação cliente em sistemas web

. Rio de Janeiro: Estácio, 2016.

DEITEL, P; DEITEL, H.

Ajax, rich internet applications e desenvolvimento web para programadores.

São Paulo: Pearson Education, 2009.

MILLETO, E; BERTAGNOLLI, S.

Desenvolvimento de software II:

introdução ao desenvolvimento web com HTML, CSS, JavaScript e PHP. Porto Alegre: Bookman, 2014.

PLOTZE, R.

Tecnologias web.

Rio de Janeiro: Estácio, 2016.

Próxima aula

- As características do Java para Web.
- O uso de tecnologias Servlet e JSP.
- O framework JSF.

Explore mais

Leia os textos:

- Introdução ao HTML 5; <https://www.w3schools.com/html/html5_intro.asp >
- Tutorial de CSS; < https://www.w3schools.com/css/ >
- Básico de JavaScript; < https://developer.mozilla.org/pt-BR/docs/Aprender/JavaScript >
- Tutorial de JavaScript; <https://www.w3schools.com/js/ >
- Eventos. <https://www.w3schools.com/tags/ref_eventattributes.asp >