1.1 Cascading Style Sheets – CSS

Uma folha de estilos consiste de uma ou mais definições de estilo (tamanho de fonte, estilo da fonte, alinhamento de texto, cor de texto e do fundo, margens, altura da linha, etc) para elementos HTML que podem ser linkados ou embutidos em documentos HTML. Esta funcionalidade foi criada para propiciar aos desenvolvedores e projetistas Web a possibilidade de contar com estilos e posicionamentos consistentes no documento.

1.1.1 O Uso de Estilos

<STYLE>

Existem várias formas de se definir estilos. Eles podem ser inseridos "inline", usando o atributo `STYLE` definido na seção `<STYLE>` ou linkado, usando o comando `<LINK>`. Ainda existem várias formas de se aplicar um estilo. Podem ser aplicados globalmente no documento ou individualmente a cada elemento.

1.1.2 Definição de Estilos

O primeiro local para se definir um estilo é no comando `<STYLE>`. Este comando possui um atributo `type` que informa ao browser qual tipo de estilo está sendo definido. Para CSS, o valor desse atributo é "`text/CSS`".

O Exemplo 3.7 mostra o uso do comando `<STYLE>` para a definição de estilos. Nesse exemplo são mostrados quatro diferentes estilos aplicados ao texto no corpo do documento. A Figura 3.7 mostra o resultado do exemplo num browser.

Exemplo 3.7 – Texto do cabeçalho redefinido com class para o IE

```
<html>
<title>JavaScript Lumina</title>
<head>
  <style type="text/css">
  <!--
    h1{
      color: green;
      font-style: italic;
      font-size: 12;
    }

    h1.special{
      font-size: 24;
    }

    .newheader{
      color: yellow;
    }

    #caps{
      font-variant: small-caps;
```

```
    }
  -->

  </style>
</head>
<body>
  <h1>Green, italic, and a point size of 12</h1>
  <h1 class="newheader">Yellow, italic, and a point size of 12</h1>
  <h1 class="special">Point size of 24</h1>
  <p>
    Here is some <span id="caps">capitalized</span> text.
  </p>

</body>
</html>
```

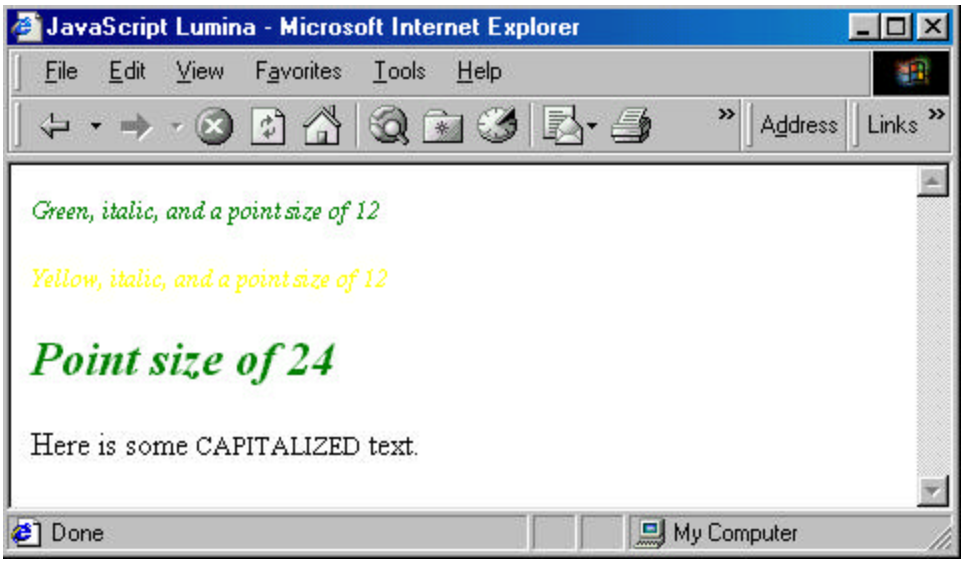

Figura 3.7 - Texto do cabeçalho redefinido com class para o IE

<LINK>

Utiliza-se se for preferível armazenar todos ou um conjunto de estilos em um único local para que as mudanças sejam globalmente aplicadas. Este elemento é usado para ligar estilos externos ao documento. Os atributos do elemento `<LINK>`, `type` e `ref` são usados para definir o tipo de link e a URL que contém o estilo desejado.

CSS externos são construídos de forma similar ao CSS definido dentro do documento. A folha de estilos é armazenada numa URL que contém as definições de estilos, mas os elementos HTML `<STYLE>` e `</STYLE>` não são necessários. Este elemento deve estar contido na seção `<HEAD>` do documento em questão. Para se incluir um arquivo de estilos chamado "`chapters.css`" deve-se usar:

```
<HEAD>
   <LINK rel = "stylesheet" type = "text/css" href =
   "http://www.mcp.com/stylesheets/chapters.css">
```

```
<HEAD>
```

1.1.3 Aplicação de Estilos

Após definir os estilos, pode-se aplicá-los usando os atributos `id` ou `class` ou ainda aplicá-los de forma global. Para aplicar um estilo globalmente para cada instância do mesmo elemento, deve-se especificar o elemento e as mudanças de estilo correspondentes Por exemplo, para mudar todas as instâncias do elemento `<P>` para serem exibidas em tamanho 16 e itálico, deve-se inserir o seguinte código na seção `<HEAD>`:

```
<HEAD>
   <STYLE type = ”text/css” >
   <!--
    P{
       font-style: italic;
       font-size: 16pt;
    }
   -->
   </STYLE>
<HEAD>
```

1.1.3.1 O Atributo id

Outra forma de se aplicar estilos é através do atributo `id`. Assim, define-se estilos na seçao `<HEAD>` do documento. O Exemplo 3.8 mostra a criação de um estilo “heading” usado posteriormente no elemento `<P>`. A Figura 3.8 ilustra a página resultante em um browser.

Exemplo 3.8 – Especificando estilos com o atributo id

```
<html>
<head>
  <title>JavaScript Lumina</title>
  <style type="text/css">
  <!--
    #heading{
      font-style: italic;
      font-size: 24;
    }
  -->
  </style>
</head>
<body>
  <p id="heading">
    Welcome to my page! Style Italic and Font 24
  </p>
  <p>
    Here is some text. Normal
  </p>
</body>
</html>
```

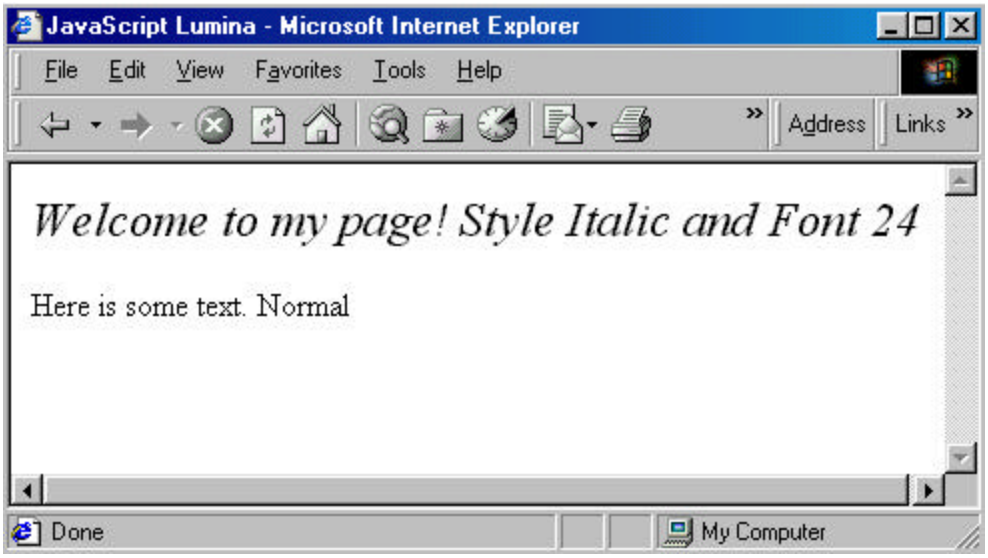

Figura 3.8 - Especificando estilos com o atributo id

1.1.3.2 O Atributo class

Estilos definidos por `class` são explicitamente chamados com os elementos HTML que os utilizam. Classes de estilo, como mostra o exemplo X.3 são uma forma de agrupar propriedades de estilos que serão utilizadas em várias partes do documento.

O exemplo 3.9 define duas classes de estilo e os aplica a diferentes instâncias do elemento <P>. A Figura 3.9 ilustra a página resultante no browser.

Exemplo 3.9 – Usando o atributo class para selecionar um estilo

```
<html>
<head>
  <title>JavaScript Lumina</title>
  <style type="text/css">
  <!--
    .heading{
      font-style: italic;
      font-size: 24;
    }

    .content{
      font-size: 12;
    }
  -->
  </style>
</head>
<body>
  <p class="heading">
    Welcome to my page! Style Font 24
  </p>
  <p class="content">
    Here is some text. Style Font 12
  </p>
```

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

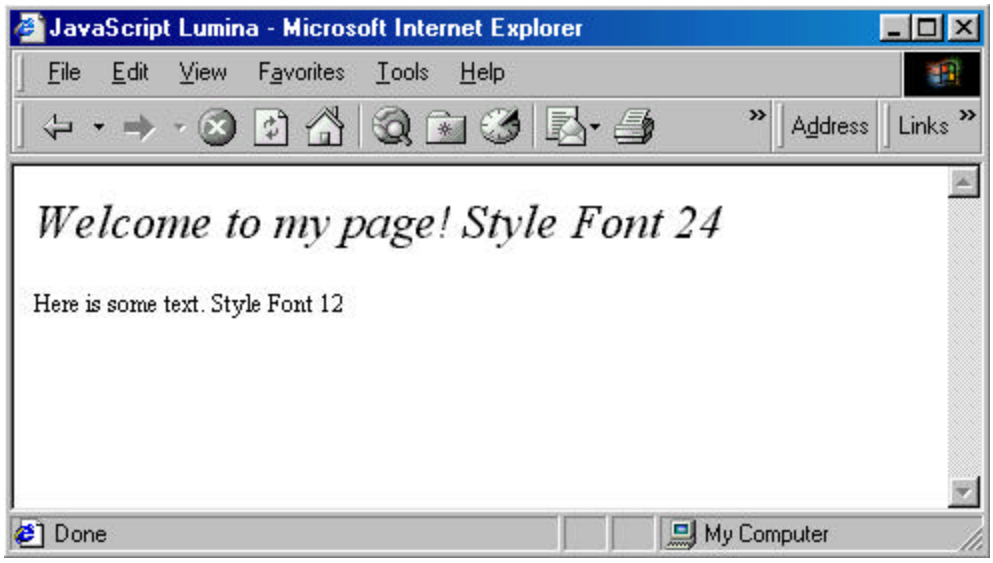

Figura 3.9 - Usando o atributo class para selecionar um estilo

*1.1.3.3 O elemento *

O último método de se aplicar um estilo é através dos elementos <SPAN> e </SPAN>. Estes elementos são usados para marcar uma parte específica do texto na qual o estilo deve ser aplicado. Este elemento é usado quando um trecho do texto dentro de um elemento precisa ter um estilo aplicado.

O exemplo 3.10 mostra como aplicar um estilo a uma parte do texto que já possui um estilo pré-definido. A Figura 3.10 ilustra a página resultante no browser.

Exemplo 3.10 – Usando o tag

```
<html>
<head>
  <title>JavaScript Lumina</title>
  <style type="text/css">
  <!--
    p{
      font-style: italic;
      font-size: 24;
    }

    #smaller{
      font-size: 12;
    }
  -->
  </style>
</head>
<body>
```

```
  <p>
    Here is some text that has global styles applied, but
    <span id="smaller">here is some smaller text (font 12)</span> in
    the middle of this paragraph (font 24).
  </p>
</body>
</html>
```

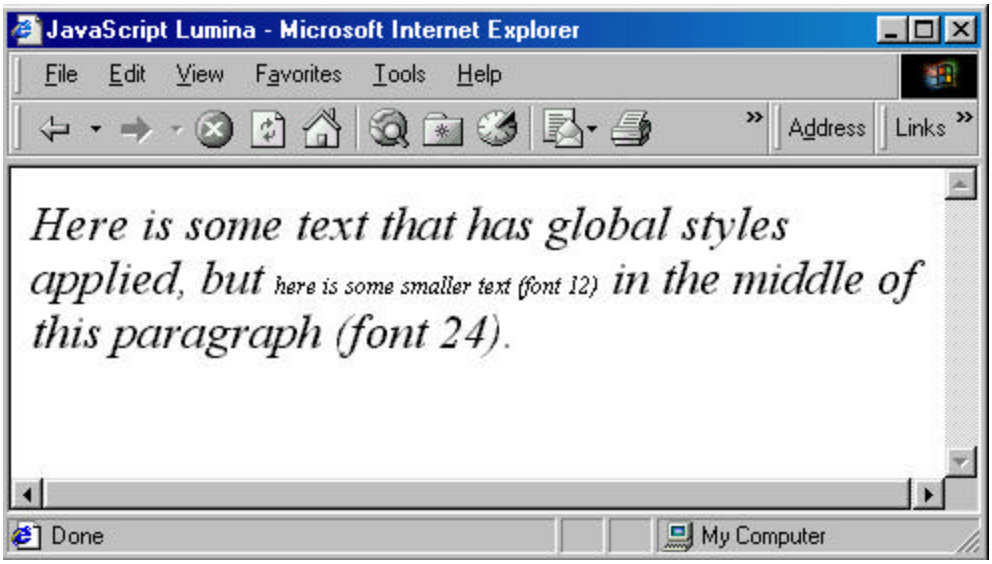

Figura 3.10 - Usando o tag

1.1.3.4 Seletores Mistos

A implementação de múltiplas folhas de estilo traz a possibilidade de definição de estilo múltiplo para elementos HTML. Qualquer definição de estilo numa folha de estilo sempre terá precedência sobre qualquer estilo definido anteriormente no documento.

O Exemplo 3.11 mistura seletores de diferentes tipos para atingir o efeito mostrado na Figura 3.11.

Exemplo 3.11 – Combinando estilos em uma única definição de estilos

```
<html>
<head>
  <title>JavaScript Lumina</title>
  <style type="text/css">
  <!--
    h1{
      color: green;
    }

    #ital{
      font-style: italic;
    }

    .newheader{
      color: yellow;
```

```
    }

    #myid{
      font-size: 12;
    }
  -->
  </style>
</head>
<body>
  <h1>Green</h1>
  <h1 class="newheader" id="myid">Yellow and point size 12</h1>
  <h1 class="newheader">Yellow and <span id="ital">italic</span></h1>
  <h2>This will be default text<h2>
</body>
</html>
```

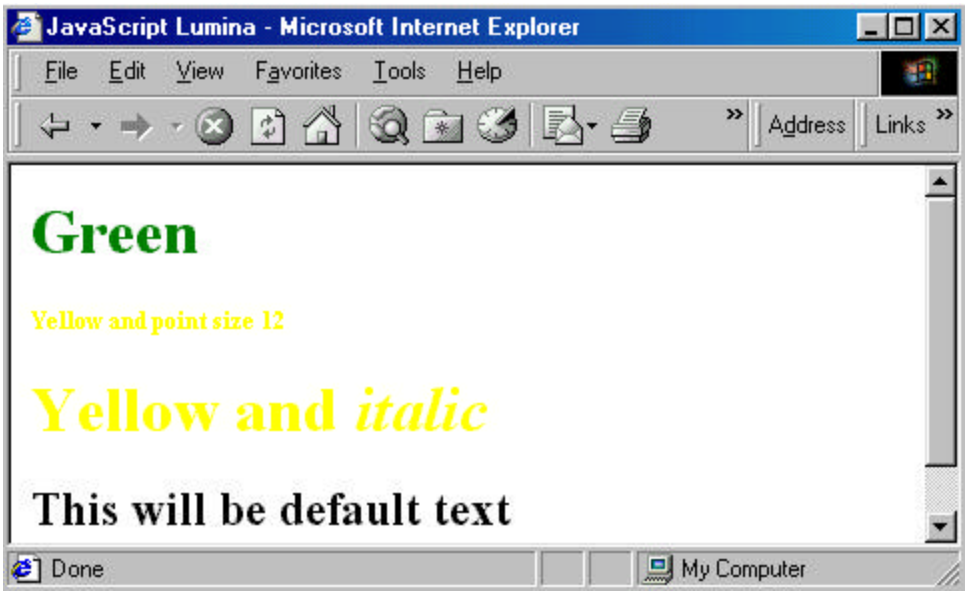

Figura 3.11 - Combinando estilos em uma única definição de estilos

1.2 Layers

Algumas das limitações das páginas Web têm sido a dificuldade de se posicionar texto ou imagem precisamente em uma página, e de sobrepor elementos HTML. Um dos aspectos mais inovadores dos browsers atuais é o suporte para layers (camadas). Layers permitem a definição de conteúdo sólido ou transparente num documento HTML que pode ser posicionado de forma precisa no local desejado. Os layers podem ser implementados utilizando várias ténicas diferentes.

Com exceção do Navigator 4, um layer pode ser criado usndo um comando HTML e aplicando propriedades de estilo (CSS) para posicionar, mostrar e controlar o que o comando vai executar. Simplesmente pela natureza do que o comando `<DIV>` representa (uma divisão de dados), foi a

escolha mais lógica para a criação de layers. O Navigator 4 inclui dois novos comandos: <LAYER> e <ILAYER> (o i vem de inflow). Diferentemente do comando <DIV>, eles possuem atributos que essencialmente permitem sua estilização. Para criar um layer com posicionamento absoluto, usa-se o comando <LAYER>, e para um layer com posicionamento relativo, usa-se <ILAYER>.

1.2.1 Uso de Folhas de Estilo para Layers

A maneira própria e padronizada de se implementar layers é controlá-los através do uso de folhas de estilo, que incluem posicionamento, estilo e visibilidade. As propriedades CSS que serão utilizadas são:

- Position
- Left, right
- Height , width
- Z-index
- Visibility

1.2.1.1 Position

A propriedade position é usada para informar ao browser como fixar um determinado elemento numa posição do documento. Existem quatro valores para esta propriedade:

Valores	**Descrição**
Absolute	As coordenadas absolutas do elemento serão especificadas. Se não estiverem especificadas o browser assume que x e y são zero.
Fixed	O elemento não deve mudar quando a janela é rolada.
Relative	Permite posicionar o elemento com base no elemento anterior.
Static	Valor default

O Exemplo 3.12 mostra a implementação do posicionamento absoluto de elementos HTML, e a Figura 3.12 ilustra a página resultante.

Exemplo 3.12 – Especificando o posicionamento absoluto de elementos com propriedades CSS

```
<html>
<head>
  <title>JavaScript Lumina</title>
  <style type="text/css">
  <!--
  #first{
```

```
    background-color: green;
    left: 0;
    position: absolute;
    top: 0;
  }

  #second{
    background-color: red;
    left: 30;
    position: absolute;
    top: 30;
  }

  #third{
    background-color: blue;
    left: 60;
    position: absolute;
    top: 60;
  }
  -->
  </style>
</head>
<body>
  <p name="layer1" id="first">
    Layer 1
  </p>
  <p name="layer2" id="second">
    Layer 2
  </p>
  <p name="layer3" id="third">
    Layer 3
  </p>
</body>
</html>
```

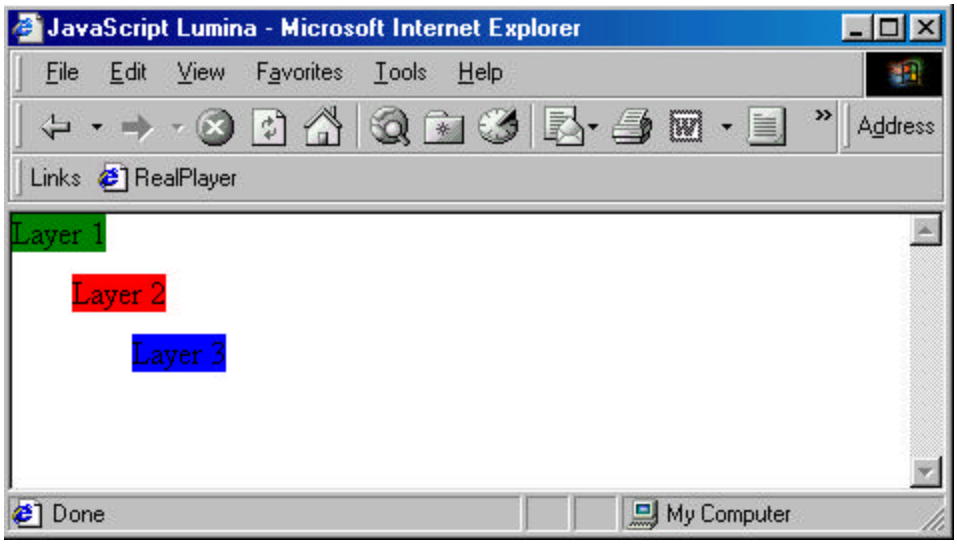

Figura 3.12 - Especificando o posicionamento absoluto de elementos com propriedades CSS

1.2.1.2 Left, right, top e bottom

São utilizadas para informar o browser o local exato em que o elemento deve ser exibido. Estas propriedades não são usadas todas juntas, por exemplo, `left` e `right` não devem estar ambas definidas, o mesmo vale para `top` e `bottom`.

O Exemplo 3.13 mostra a implementação do posicionamento de quatro elementos HTML do tipo layer, usando as propriedades `left`, `right`, `top` e `bottom`. A Figura 3.13 ilustra a página resultante, com os objetos dispostos precisamente nos locais definidos.

Exemplo 3.13 – Posicionando objetos usando as propriedades left, right, top e bottom

```
<html>
<head>
  <title>JavaScript Lumina</title>
  <style type="text/css">
  <!--
  #first{
  background-color: green;
  left: 0;
  position: absolute;
  top: 0;
  }

  #second{
  background-color: red;
  position: absolute;
  right: 0;
  top: 0;
  }

  #third{
  background-color: blue;
  bottom: 0;
  position: absolute;
  right: 0;
  }

  #fourth{
  background-color: yellow;
  bottom: 0;
  left: 0;
  position: absolute;
  }

  -->
  </style>
</head>
<body>
  <p name="layer1" id="first">
    Layer 1
  </p>
  <p name="layer2" id="second">
    Layer 2
  </p>
  <p name="layer3" id="third">
    Layer 3
```

```
  </p>
  <p name="layer4" id="fourth">
    Layer 4
  </p>
</body>
</html>
```

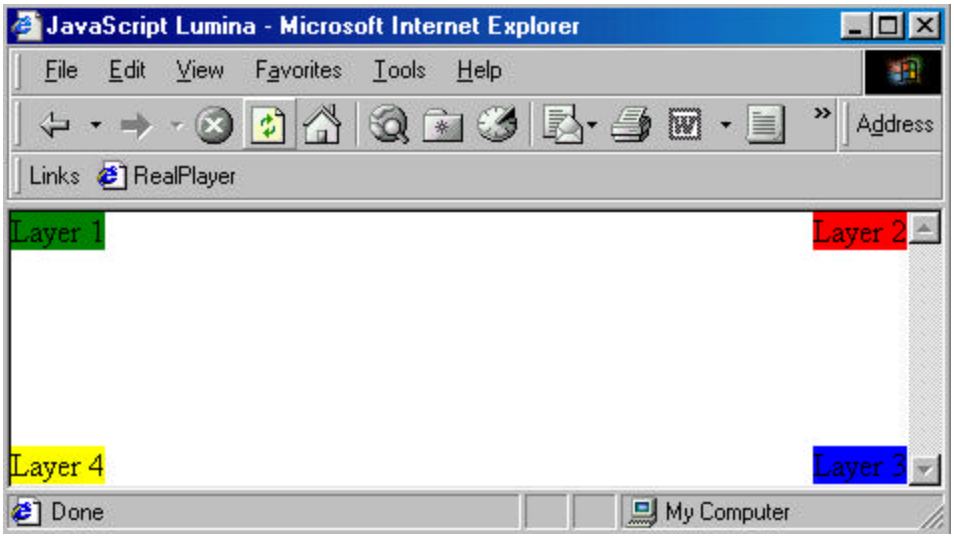

Figura 3.13 – Usando as propriedades right, left, top e bottom

1.2.1.3 Height e width

Estas duas propriedades permitem definir a altura e a largura do layer, e devem receber valores numéricos.

O Exemplo 3.14, continuação do exemplo anterior, altera o tamanho e a largura dos objetos exibidos, através das propriedades `height` e `width`. A Figura 3.14 mostra os objetos com os tamanhos alterados.

Exemplo 3.14 – Usando height e width para especificar o tamanho do layer

```
<html>
<head>
  <title>JavaScript Lumina</title>
  <style type="text/css">
  <!--
  #first{
  background-color: green;
  height:20;
  left: 0;
  position: absolute;
  top: 0;
  width: 40;
  }

  #second{
  background-color: red;
  height:10;
```

```
  position: absolute;
  right: 0;
  top: 0;
  width: 30;
  }

  #third{
  background-color: blue;
  bottom: 0;
  height:100;
  position: absolute;
  right: 0;
  width: 50;
  }

  #fourth{
  background-color: yellow;
  bottom: 0;
  height:100;
  left: 0;
  position: absolute;
  width: 200;
  }

  -->
  </style>
</head>
<body>
  <p name="layer1" id="first">
    Layer 1
  </p>
  <p name="layer2" id="second">
    Layer 2
  </p>
  <p name="layer3" id="third">
    Layer 3
  </p>
  <p name="layer4" id="fourth">
    Layer 4
  </p>
</body>
</html>
```

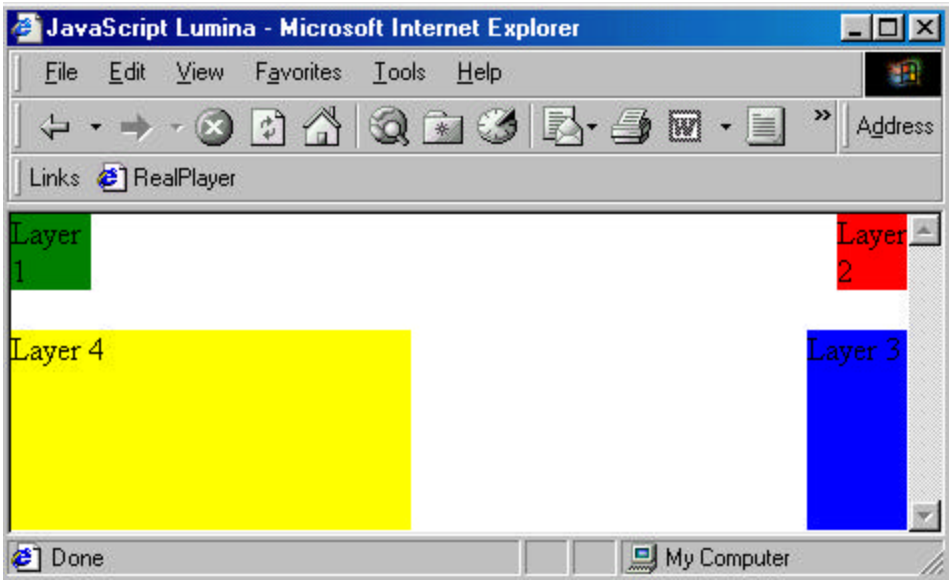

Figura 3.14 – Usando height e width para especificar o tamanho do layer

O Exemplo 3.15, mostra a sobreposição dos elementos HTML implementados em layers, usando apenas as propriedades definidas anteriormente. A Figura 3.15 exibe a sobreposição dos quadrados coloridos, sendo que a ordem da sobreposição é a ordem em que os elementos aparecem no código.

Exemplo 3.15 – Sobreposição de layers

```
<html>
<head>
 <title>JavaScript Lumina</title>
 <style type="text /css">
 <!--
 #first{
 background-color: green;
 height: 100;
 left: 0;
 position: absolute;
 top: 0;
 width: 100;
 }

 #second{
 background-color: red;
 height: 100;
 left: 20;
 position: absolute;
 top: 20;
 width: 100;
 }

 #third{
 background-color: blue;
 height: 100;
```

```
 left: 40;
 position: absolute;
 top: 40;
 width: 100;
 }

 #fourth{
 background-color: yellow;
 height: 100;
 left: 60;
 position: absolute;
 top: 60;
 width: 100;
 }

 -->
 </style>
</head>
<body>
 <p name="layer1" id="first">
  Layer 1
 </p>
 <p name="layer2" id="second">
  Layer 2
 </p>
 <p name="layer3" id="third">
  Layer 3
 </p>
 <p name="layer4" id="fourth">
  Layer 4
 </p>
</body>
</html>
```

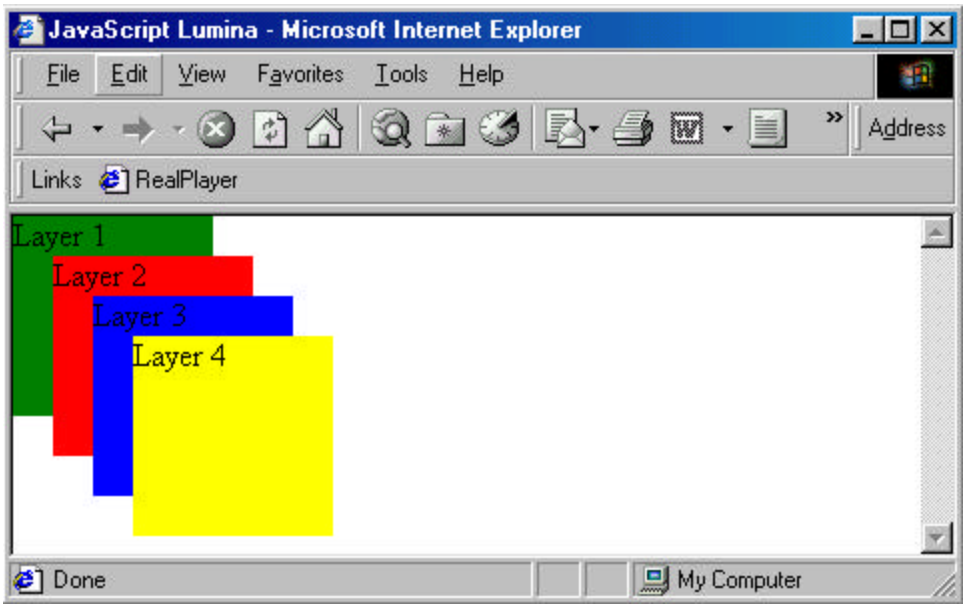

Figura 3.15 – Sobreposição de layers

1.2.1.4 z-index

A propriedade `z-index` possibilita a sobreposição dos layers através do uso de uma pilha. O atributo `z-index` deve receber um inteiro positivo que especifica em qual posição da pilha o layer deve ser colocado. Quanto maior for o `z-index`, mais acima fica o layer na sobreposição. Se o atributo `z-index` não estiver definido, admite-se que seu valor é zero.

O Exemplo 3.16 mostra uma sobreposição de objetos utilizando o `z-index` para gerar a ordem em que os objetos devem ser exibidos. A Figura 3.16 mostra os objetos e a sobreposição dos mesmos.

Exemplo 3.16 – Trocando o z-index dos layers (Ordem dos layers)

```
<html>
<head>
  <title>JavaScript Lumina</title>
  <style type="text/css">
  <!--
  #first{
  background-color: green;
  height: 100;
  left: 0;
  position: absolute;
  top: 0;
  width: 100;
  z-index: 2;
  }

  #second{
  background-color: red;
  height: 100;
  left: 20;
  position: absolute;
  top: 20;
  width: 100;
  }

  #third{
  background-color: blue;
  height: 100;
  left: 40;
  position: absolute;
  top: 40;
  width: 100;
  z-index: 1;
  }

  #fourth{
  background-color: yellow;
  height: 100;
  left: 60;
  position: absolute;
  top: 60;
  width: 100;
  }

  -->
  </style>
```

```
</head>
<body>
  <p name="layer1" id="first">
    Layer 1
  </p>
  <p name="layer2" id="second">
    Layer 2
  </p>
  <p name="layer3" id="third">
    Layer 3
  </p>
  <p name="layer4" id="fourth">
    Layer 4
  </p>
</body>
</html>
```

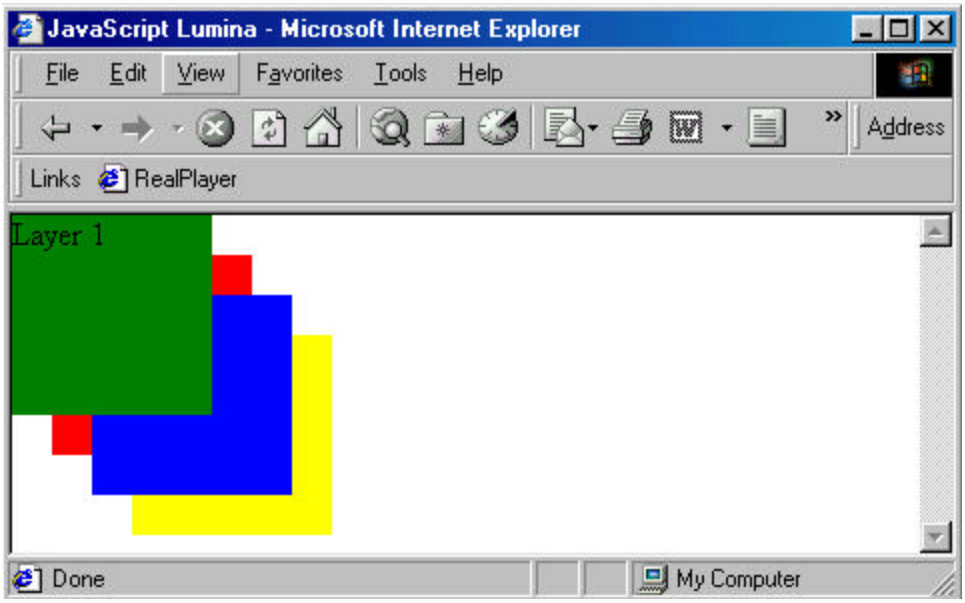

Figura 3.16 – Trocando o z-index dos layers (Ordem dos layers)

1.2.1.5 Visibility

A propriedade `visibility` é responsável por ocultar ou deixar visível um determinado layer. Possui quatro valores possíveis:

Valores	**Descrição**
Collapse	Possui o mesmo significado de `hidden`, exceto quando utilizado em tabelas.
Hidden	Oculta o elemento.
Inherit	Valor defaul do browser, recebe o valor do elemento anterior.
Visible	Torna o elemento visível.

O Exemplo 3.17, como os exemplos anteriores, mostra a exibição de layers sobrepostos, porém agora com a propriedade `visibility` aplicada de forma a ocultar dois dos objetos que poderiam ser exibidos. A Figura 3.17 mostra os layers em que os atributos `visibility` não receberam o valor `hidden`.

Exemplo 3.17 Ocultando layers

```
<html>
<head>
  <title>JavaScript Lumina</title>
  <style type="text/css">
  <!--
  #first{
  background-color: green;
  height: 100;
  left: 0;
  position: absolute;
  top: 0;
  width: 100;
  z-index: 2;
  visibility: hidden;
  }

  #second{
  background-color: red;
  height: 100;
  left: 20;
  position: absolute;
  top: 20;
  width: 100;
  }

  #third{
  background-color: blue;
  height: 100;
  left: 40;
  position: absolute;
  top: 40;
  width: 100;
  z-index: 1;
  visibility: hidden;
  }

  #fourth{
  background-color: yellow;
  height: 100;
  left: 60;
  position: absolute;
  top: 60;
  width: 100;
  }

  -->
  </style>
</head>
<body>
  <p name="layer1" id="first">
    Layer 1
  </p>
```

```
  <p name="layer2" id="second">
    Layer 2
  </p>
  <p name="layer3" id="third">
    Layer 3
  </p>
  <p name="layer4" id="fourth">
    Layer 4
  </p>
</body>
</html>
```

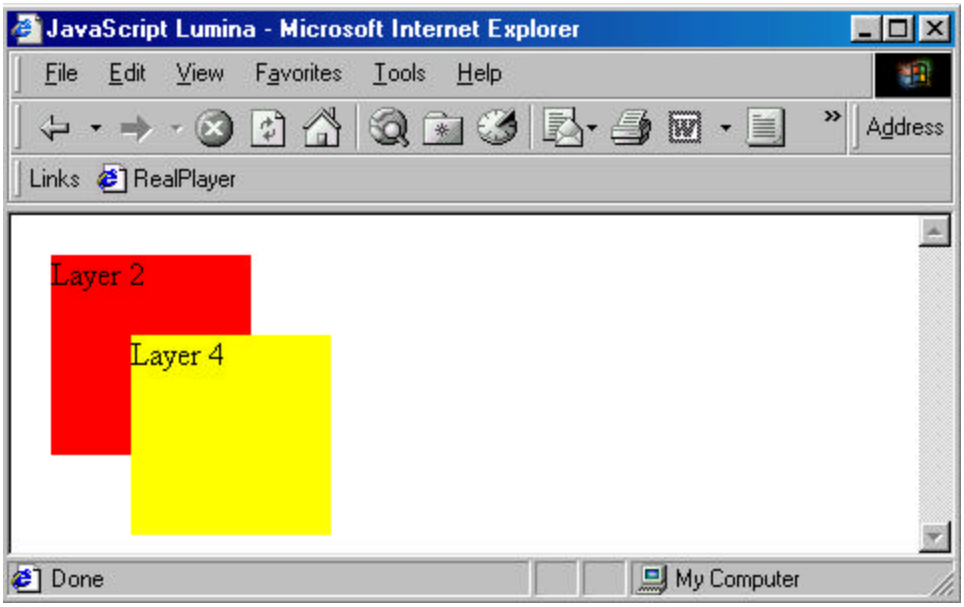

Figura 3.17 - Escondendo layers

1.2.2 Gerenciamento de Layers Sobrepostos

As propriedades CSS permitem a especificação de efeitos visuais em documentos HTML. Por exemplo, suponha a exibição de um relatório de dados numa página Web. O objetivo é exibir uma tela por vez e permitir ao usuário ver uma outra tela. Isso pode ser feito através de layers, criando o mesmo efeito de janelas sobrepostas do ambiente windows.

1.2.2.1 Criação de Efeitos de Animação

Um aspecto intrigante dos layers é a forma como podem ser usados para criar animações. O JavaScript permite o posicionamento dinâmico dos layers e desta forma podem-se criar funções recursivas nas quais os layers se movimentem no documento. Esta é a combinação HTML e JavaScript para a criação de HTML dinâmico, ou DHTML.

Com o uso de layers, JavaScript e folhas de estilo pode-se criar (com HTML), projetar (com CSS) e manipular (com JavaScript) elementos numa página. Com os avanços tais como melhor suporte do modelo de objetos do documento (DOM) e XML (eXtensive Markup Language) nos browsers, as possibilidades são sem fim.

** e <iframe>**

O melhor método para a implementação dos layers é através dos elementos `<DIV>` e `<IFRAME>`. Os elementos `<LAYER>` e `<ILAYER>` da Netscape estão desaprovados.

1.2.3 Definição de Blocos de Dados

O primeiro passo na criação de layers é definir o bloco de dados que será considerado um layer. Para uma aplicação comum, a primeira janela do browser deve ser considerada o primeiro layer. Um botão no primeiro layer pode disparar um menu, considerado então no segundo layer, no topo do documento. Se um clique no menu dispara um submenu em cascata então este é um novo layer (podendo estar ou não no mesmo nível do anterior).

A definição dos blocos de dados é feita através do uso do elemento `<DIV>`. Os atributos do elemento são:

Atributos	**Descrição**
Align	Usado para alinhar os dados. Recebe os valores `left`, `right`, `center` e `justify`. O uso de CSS é mais aconselhável para o alinhamento.
Class	Lista de classes de estilo, separadas por vírgula, que faz do elemento uma instância das classes.
Dir	Especifica a direção do texto: ltr (left to right) e rtl (right to left).
Id	Usado para definir o tipo do estilo que deve ser aplicado aos dados.
Lang	Identifica o código da linguagem humana em que os dados estão.
Name	Usado para dar um nome ao bloco. Pode ser usado pelo JavaScript para manipular os layers.
Style	Permite especificar uma definição de estilo ao invés de uma folha de estilos.
Title	Provê um título para a divisão.

1.2.3.1 Definição de um bloco de dados

Deve-se colocar o bloco de dados entre os elementos `<div>` e `</div>`. O Exemplo 3.18 mostra a criação de duas divisões

chamadas “layer1” e “layer2” no corpo do documento HTML, ilustradas na Figura 3.18.

Exemplo 3.18 – Usando o elemento `<div>`

```
<html>
<head>
  <title>JavaScript Lumina</title>
</head>
<body>
  Before the first block.
  <div name="layer1">
    <hr>
    DIV 1
    <hr>
  </div>
  After the first block.
  <br>
  Before the second block.
  <div name="layer2">
    <h3>
      DIV 2
    </h3>
    <p>
      I am inside the second DIV block.
    </p>
  </div>
  After the second block.
</body>
</html>
```

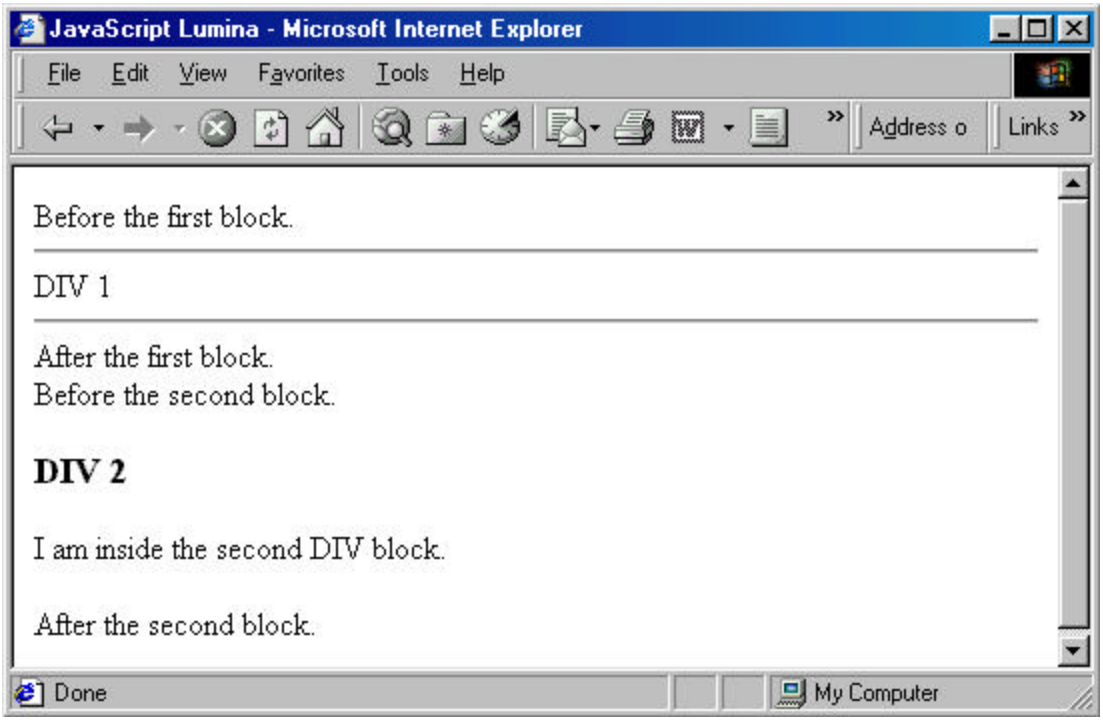

Figura 3.18 – Definindo blocos de dados no documento

1.2.3.2 Posicionamento de um bloco de dados

Para fazer o posicionamento de um bloco de dados deve-se atribuir ao atributo `id` as propriedades de estilo, que devem conter a posição correta em que o bloco deve ser exibido.

O Exemplo 3.19 mostra a aplicação de folhas de estilo para posicionar um bloco (divisão), contendo os atributos que definem altura, largura, coordenadas `x` e `y` (left e top), cor e forma de posicionamento. A Figura 3.19 apresenta o resultado em um browser.

Exemplo 3.19 – Aplicando style sheets e posições no bloco `<div>`

```
<html>
<head>
  <title>JavaScript Lumina</title>
  <style type="text/css">
  <!--
    #first{
    background-color: green;
    height: 100;
    left: 275;
    position: absolute;
    top: 40;
    width: 100;
    }

    #second{
    background-color: red;
    height: 100;
    left: 100;
    position: absolute;
    top: 55;
    width: 100;
    }
  -->
  </style>
</head>
<body>
  Before the first block.
  <div name="layer1" id="first">
    <hr>
    DIV 1
    <hr>
  </div>
  After the first block.
  <br>
  Before the second block.
  <div name="layer2" id="second">
    <h3>
      DIV 2
    </h3>
    <p>
      I am inside the second DIV block.
    </p>
```

```
  </div>
  After the second block.
</body>
</html>
```

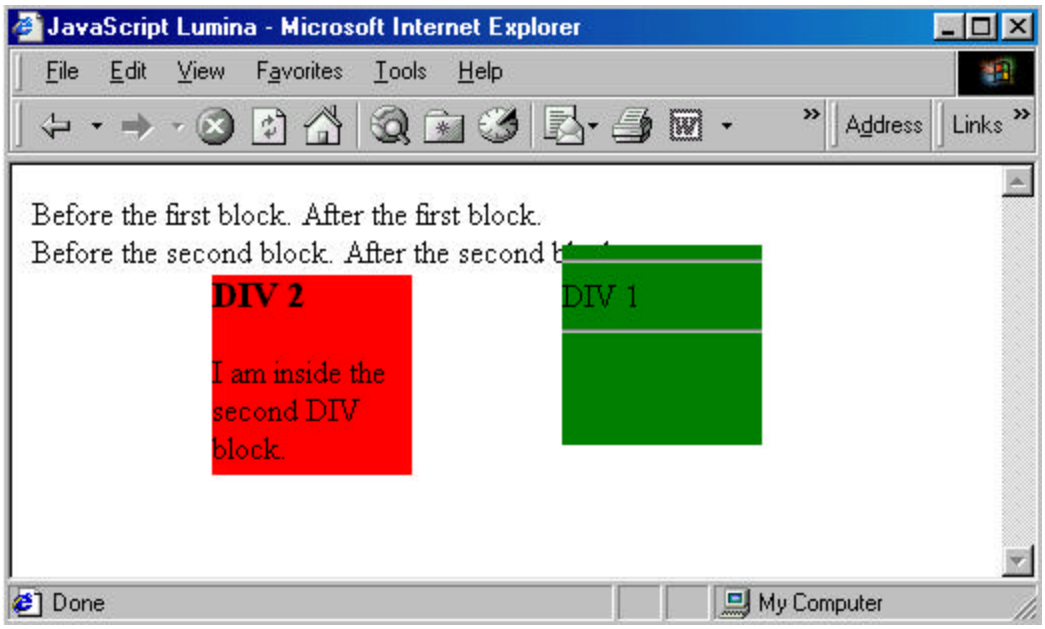

Figura 3.19 – Posicionando blocos de dados

1.2.3.3 Manipulação com JavaScript

A sintaxe a ser utilizada em JavaScript para manipular layers é diferente para os browsers Navigator e Internet Explorer. Nos browsers Navigator, o acesso é feito através do array `layers`. No caso do exemplo X.8 apresentado anteriormente, o atributo `name` do elemento `<div>` seria usado para especificar o nome do layer:

```
document.layers['layer1']
```

Nos browsers Internet Explorer o acesso é feito através do array `all`. O atributo `id` do elemento `<div>` especificaria o layer:

```
Document.all['first ']
```

Quando se criam scripts para rodar em ambos os browsers, o mesmo valor deve ser atribuído aos atributos id e name do elemento . O próximo passo é armazenar a referência ao array, para que se possa obter o layer através de uma variável. O código a seguir mostra estas questões:

```
    var layer = new String();
    var style = new String();

    if(isIE){
        layer = ".all";
        style = ".style";
    }else if(isNav){
        layer = ".layers";
        style = "";
    }
```

O método `eval` é também bastante utilizado, recebe um string e o avalia como uma expressão JavaScript. Pode-se construir uma chamada para verificar e alterar uma propriedade de um layer, tal como a visibilidade.

O Exemplo 3.20 mostra definições de layers, uma função para checar o browser que está sendo utilizado e atualizar as variáveis necessárias, e a utilização do método `eval`, que recebe um string com o código para alterar a propriedade de visibilidade de um bloco. A Figura 3.20 mostra a página resultante num browser. O botão Hide ao ser pressionado muda a propriedade de visibilidade tornando o bloco invisível, e o botão Show altera a propriedade deixando o bloco visível.

Exemplo 3.20 – Acessando layers com JavaScript

```
<html>
<head>
  <title>JavaScript Lumina</title>
  <style type="text/css">
  <!--
    #layer1{
    background-color: green;
    height: 100;
    left: 10;
    position: absolute;
    top: 50;
    width: 100;
    }
  -->
  </style>
  <script type="text/javascript" language="JavaScript1.2">
  <!--
    // Create global variables for browser type
    var isIE = new Boolean(false);
    var isNav = new Boolean(false);
    var unSupported = new Boolean(false);
    var layer = new String();
    var style = new String();

    // Determine if the browser is Internet Explorer, Navigator,
    // or other. Also, set the layer variable depending on the
    // type of access it needs.
    function checkBrowser(){
      if(navigator.userAgent.indexOf("MSIE") != -1){
        isIE = true;
        layer = ".all";
        style = ".style";
      }else if(navigator.userAgent.indexOf("Nav") != -1){
        isNav = true;
        layer = ".layers";
```

```
        style = "";
      }else{
        unSupported = true;
      }
    }

    // Take the state passed in, and change it.
    function changeState(layerRef, state){
      eval("document" + layer + "['" + layerRef + "']" + style +
".visibility = '" + state + "'");
    }
  //-->
  </script>
</head>
<body onload="checkBrowser()">
  <div name="layer1" id="layer1">
    DIV 1
  </div>
  <form name="form1">
    <input type="button" value="Hide"
onclick="changeState('layer1','hidden')">
    <input type="button" value="Show"
onclick="changeState('layer1','visible')">
  <form>
</body>
</html>
```

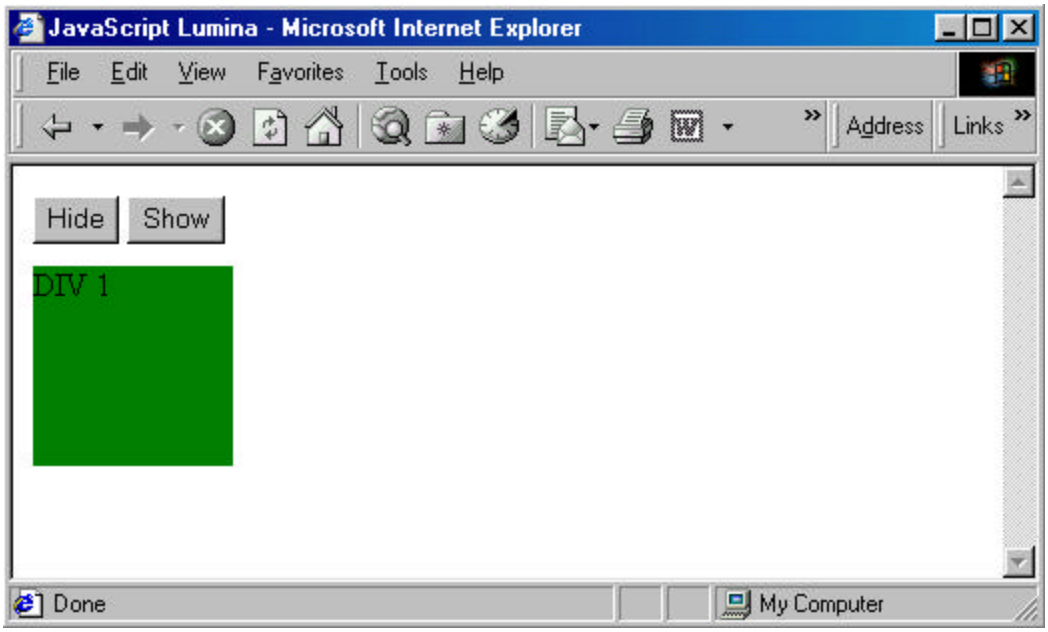

Figura 3.20 - O conteúdo da página antes de pressionar o botão