6

O browser

COM O GRANDE NÚMERO DE VERSÕES, PLATAFORMAS E FABRICANTES de browsers, cada um suportando extensões proprietárias e introduzindo recursos incompatíveis, é útil poder indentificar o browser que está carregando uma determinada página. Com esta informação, pode-se tomar a decisão utilizar instruções que só existem naquela versão de browser, ou de redirecionar a janela para outra página. Informações sobre o cliente que acessa uma página são mantidas pela propriedade global `navigator`.

Objeto Navigator

O objeto *Navigator*[1] representa as propriedades do browser. Usando suas propriedades e métodos booleanos (que retornam `true` ou `false`) é possível identificar as possibilidades de um cliente e desenvolver páginas personalizadas com conteúdo específico para aproveitar ao máximo os recursos existentes.

Navigator define as caracteríticas de um único objeto, representado pela propriedade global[2] `navigator`. Todas as suas propriedades são somente-leitura. Todas as cópias de `navigator` em uma mesma aplicação são idênticas e possuem as mesmas propriedades.

As informações que se pode obter através da propriedade navigator são:

- Marca, nome, plataforma e versão do browser do cliente
- Plug-ins instalados no cliente (em browsers Netscape).
- Tipos de dados MIME suportados pelo browser e pela plataforma do cliente, através de plug-ins e programas externos ao browser habilitados a funcionarem como aplicações auxiliares para tipos desconhecidos do browser (Netscape).

---

[1] Assim como *Window*, não existe na documentação do JavaScript 1.1 um objeto ou construtor chamado 'Navigator'. Usamos este nome apenas para referirmos ao tipo que define os métodos e propriedades do objeto `navigator` e manter a consistência com os outros objetos.

[2] No Internet Explorer, navigator não é global, mas é propriedade de window.

Quatro propriedades contém informações do fabricante do browser, e outras duas, são vetores com objetos do tipo *PlugIn* e *MimeType*, usados para identificar plug-ins e tipos suportados. Não estão disponíveis em todos os browsers. As propriedades de *Navigator* estão listadas na tabela abaixo. Para utilizá-las, é preciso usar o objeto `navigator` da forma:

```
navigator.propriedade
```

Propriedade	**Descrição**

`userAgent`	Contém *String.* Informação contida no cabeçalho HTTP `User-Agent`. Esta propriedade é a combinação das propriedades `appCodeName` e `appVersion`.
	Exemplos:
	`Mozilla/4.0 (compatible; MSIE 4.0; Windows 95)`
	`Mozilla/4.5 [en] (Win95; I)`

`appCodeName`	Contém *String.* Contém o nome interno do browser. *Exemplo:* `Mozilla`

`appVersion`	Contém *String.* Contém informações sobre a versão.
	Exemplos:
	`4.0 (compatible; MSIE 4.0; Windows 95)`
	`4.5 [en] (Win95; I)`

`appName`	Contém *String.* Contém o nome oficial do browser.
	Exemplos:
	`Microsoft Internet Explorer`
	`Netscape`

`mimeTypes`	Contém *Array.* Um vetor de tipos MIME que descrevem os tipos MIME reconhecidos e suportados pelo browser, internamente, via plug-ins ou através de aplicações auxiliares (do sistema operacional).
`plugins`	Contém *Array.* Um vetor com todos os plug-ins instalados no cliente.

Com as propriedades `userAgent`, e `appName`, obtemos diversas informações sobre o browser. Para utilizá-las é preciso isolar o sub-string com a informação correspondente. Infelizmente os formatos diferem entre os principais fabricantes, mas é possível identificar as três informações mais importantes, onde ocorrem as maiores diferenças: nome e fabricante, versão e plataforma.

Identificação do nome do fabricante

O nome e fabricante do produto é fácil de obter. Usa-se a propriedade `appName`. Sabendo-se de antemão o nome utilizado pelo fabricante em cada tipo de browser, é possível usá-lo para comparar com o string contido em `appName`, e verificar se o browser atual é um deles:

```
if (navigator.appName == "Microsoft Internet Explorer") {
   // código que só será executado em browsers Internet Explorer
}
```

Identificação da versão

Obter a versão do browser é uma tarefa mais complicada. Ela está disponível tanto na propriedade `userAgent`, como na propriedade `appVersion`. A versão aqui refere-se ao *browser de referência* "Mozilla", tanto para Netscape como para o Internet Explorer. O Internet Explorer 3, por exemplo, é compatível com o Netscape Navigator 2.0, portanto a versão que aparece para o Internet Explorer 3 é 2.0 e não 3.0[3].

Usar `appVersion` é mais fácil pois a versão está logo no início do string. Tanto nos browsers da Microsoft quando nos browsers da Netscape a primeira coisa após a versão é um espaço. Portanto, basta identificar o espaço, e recuperar o substring que está antes dele. Eis duas formas de fazer isto:

```
espaco = navigator.appVersion.indexOf(" ");
versao = parseFloat(navigator.appVersion.substring(0, espaco));

versao = parseInt( navigator.appVersion.split(" ")[0] );
```

Se apenas interessa o valor maior da versão, pode-se truncar o que está depois do ponto usando simplesmente:

```
versao = parseInt(navigator.appVersion);
```

Depois de extraída a versão, ela pode ser usada para executar trechos de código dependentes de browser:

```
if (versao < 3.0 && navigator.appName == "Netscape") {
   // código para browsers Netscape de versões inferiores a 3
}
```

Identificação da plataforma

A posição e o tamanho da informação sobre a plataforma diferem nos browsers Internet Explorer e Netscape Navigator. O ideal, portanto, é identificar primeiro o browser, para depois identificar a plataforma. Strings como "Win", "Mac", "95" e "NT" estão presentes em ambos os browsers. Pode-se então localizar esses *strings* em `appVersion`:

```
if (navigator.appVersion.lastIndexOf('Win') != -1) {
   // é Windows... que tipo?
if (navigator.appVersion.lastIndexOf('NT') != -1)
   // é Windows NT
else if (navigator.appVersion.lastIndexOf('Mac') != -1) {
   // é Macintosh
} else {
   // é outra plataforma... Unix talvez
}
```

---

[3] Para obter a versão verdadeira do Internet Explorer, é preciso extrair a informação localizada entre parênteses no meio do string `appVersion` do Internet Explorer `(compatible; MSIE 4.0; Windows 95)`

Exercício Resolvido

Escreva um programa JavaScript que redirecione a janela do browser de acordo com o tipo de browser que a carregar. Para redirecionar, use a instrução:

```
location.href = "url destino";
```

As páginas destino estão localizadas no diretório `cap6/`. Os arquivos destino são:

- a página `msie.html`, se o browser for Microsoft Internet Explorer 4, ou superior
- a página `netscape.html`, se o browser for Netscape 3 ou superior.
- a página `outro.html` se o browser não estiver entre os tipos acima mas suportar JavaScript 1.1.

Se o browser não suportar JavaScript 1.1, deve permanecer na página.

Solução

A solução está mostrada no código a seguir (arquivo `redir.html`) e deve estar em um bloco `<SCRIPT>` no início da página.

```
<html> <head>
  <script language="JavaScript1.1">
  <!--
     browser = navigator.appName;
     versao = parseInt(navigator.appVersion);
     netscape = "Netscape";
     explorer = "Microsoft Internet Explorer";

     if (browser == netscape && versao >= 3) {
        location.href = "netscape.html";
     } else if (browser == explorer && versao >= 4) {
        location.href = "msie.html";
     } else {
        location.href = "outro.html";
     }
  // -->
  </script>
</head>
<!-- Somente browsers que não suportam JavaScript 1.1 continuarão -->
(...)
```

Browsers que são exceções (não foram previstos pelo código) sempre devem permanecer na página ou ficar com parâmetros *default* (que não dependam do código) pois sempre existe a possibilidade do browser não entender JavaScript de forma alguma. O bloco `<SCRIPT>` com o atributo `LANGUAGE=JavaScript1.1` garante que browsers que *suportam* versões de JavaScript inferiores a 1.1 não irão tentar interpretar o código.

Métodos

Os métodos de navigator são apenas dois:

Método	**Ação**
`javaEnabled()`	Retorna `true` se o suporte a Java está habilitado.
`taintEnabled()`	Retorna `true` se o modelo de segurança data-tainting está habilitado.

Se o usuário não suporta data-tainting (o que é comum), certas operações como a chamada de métodos em outras janelas poderá não funcionar, se contiverem arquivos de servidores diferentes. Se o suporte a Java não estiver ativado, o browser não será capaz de executar applets. Sabendo disso, o programador poderá oferecer uma alternativa.

Suponha, por exemplo, que uma página use formulários HTML para oferecer uma interface de conexão a um serviço. O formulário é simples, consiste de duas caixas de texto (para login e senha) e um botão "Conectar":

```
<form action="auth.exe?verhtml" method=POST>
 <p>User ID <input type=text name=usuario size=14>
    Senha   <input type=text name=senha size=14 maxlength=10>
            <input type=submit value="Conectar">
</form>
```

Suponha agora que o autor do site decida substituí-la por um applet, oferecendo a mesma interface, mas aproveitando os recursos de segurança da linguagem Java. No lugar do formulário, a página teria um descritor HTML do tipo:

```
<applet code=PainelCon.class height=80 width=450></applet>
```

Se o usuário não suporta Java, não poderá executar o painel, e o site perderá vários de seus clientes. Se ele voltar a usar somente HTML, deixará de aproveitar os recursos mais modernos presentes nos browsers de um número crescente de clientes, que também usam serviços de seus concorrentes. Uma solução é verificar se o browser do cliente suporta Java e, caso positivo, carregar o applet; caso contrário, carregar o formulário antigo:

```
<script language=JavaScript1.1>     <!--
if (navigator.javaEnabled()) {
   document.write("<applet code=PainelCon.class height=80 width=450>");
   document.write("</applet>");
} else {
   document.write("<form action=\"auth.exe?verhtml\" method=POST>");
   document.write("<p>User ID <input type=text name=usuario size=14>");
   document.write(" Senha <input type=text name=senha size=14>");
   document.write(" <input type=submit value=\"Conectar\">");
   document.write("</form>");
}
//-->
</script>
```

```
<noscript>  <!-- browsers que tem JavaScript ativado, ignoram -->
<form action="auth.exe?verhtml" method=POST>
<p>User ID <input type=text name=usuario size=14>
 Senha <input type=text name=senha size=14 maxlength=10>
 <input type=submit value="Conectar">
</form>
</noscript>
</BODY>
```

O código acima só instala o applet se o browser suportar JavaScript e Java e tiver os dois habilitados. Browsers que suportam JavaScript e não suportam Java ou não habilitam Java no browser, interpretarão o bloco `else {...}`, e receberão um formulário HTML. Browsers que suportam JavaScript, mas estão com o mesmo desabilitado, interpretarão o código em `<noscript>` que também constrói o mesmo formulário HTML. Browsers que não suportam JavaScript, e que ignoram o significado de `<script>` e `<noscript>`, não imprimirão na tela o código em `<script>`, por estar entre comentários HTML (`<--` e `-->`) mas *interpretarão* o HTML dentro de `<noscript>` e `</noscript>` por *não* estar comentado.

Plug-ins e tipos MIME

Assim como fizemos com os applets Java, podemos usar JavaScript para verificar se o browser do usuário possui um determinado plug-in instalado. Também é possível verificar se a plataforma do cliente é capaz de executar ou interpretar algum tipo de conteúdo como formatos de imagens, vídeos, textos em Word, arquivos executáveis, etc. identificados através de tipos MIME[4]. A maior parte desses recursos está disponível apenas para browsers Netscape, portanto eles não serão explorados aqui em detalhe.

Dois tipos de objetos são acessíveis através de propriedades do objeto *Navigator* que fornecem essas informações: *PlugIn* e *MimeTypes*. Como pode haver vários plug-ins instalados e geralmente um cliente suporta vários tipos MIME, esses objetos são obtidos através de vetores.

MimeType

Cada objeto do vetor `navigator.mimetypes` é um objeto *MimeType*, que possui propriedades de tipo, descrição, extensões de arquivo e plug-ins habilitados para suportar o tipo. Para obter uma lista de todos os tipos MIME suportados em um browser Netscape Navigator 3.0 ou superior, pode-se fazer:

---

[4] Multipart Internet Mail Extension. Padrão Internet para identificação de conteúdo baseado em um par de identificadores que representam um tipo genérico (imagem, texto, aplicação, etc.) e um tipo específico (JPEG, GIF, EXE, HTML, TEX, etc.). O formato é tipo_genérico/tipo_específico. Exemplos: image/jpeg, image/gif, text/html, text/plain, application/msword, application/exe.

```
for (i=0; i < navigator.mimeTypes.length; i++) {
   document.write("<br>" + navigator.mimeTypes[i].type)
}
```

O número de tipos suportados geralmente é extenso, pois não se limita ao browser. Todas as aplicações que registram um tipo de dados no sistema operacional estão disponíveis como aplicações auxiliares e seus tipos MIME são levados em consideração. A melhor forma de ter acesso aos objetos é utilizando o nome do tipo entre colchetes (vetor associativo), em vez do índice:

```
tipo = navigator.mimetypes["tipo/subtipo"];
```

A tabela abaixo lista as propriedades do tipo *MimeType* e a informação que contém. Todas são *read-only*. O exemplo apresentado em cada mostra em negrito os valores de cada propriedade para o tipo `text/html`:

Propriedade	**Descrição** *(e tipo de dados)*
`name`	*String*. Tipo MIME no formato tipo/subtipo. *Exemplo*: `navigator.mimetypes["text/html"].name` contém "**text/html**"
`description`	*String*. Descrição em inglês do tipo de conteúdo representado pelo tipo MIME. Exemplo: `navigator.mimetypes["text/html"].description` contém "**Hypertext Markup Language**"
`suffixes`	*String*. Lista de extensões comuns de arquivos associados com este tipo MIME. Exemplo: `navigator.mimetypes["text/html"].suffixes` contém "**html, htm, shtml**"
`enabledPlugin`	*PlugIn*. Referência ao objeto *PlugIn* que suporta este tipo, ou `null` se não é suportado por um plug-in (é nativo ou suportado por aplicação externa). Se um tipo é suportado por um plug-in, um arquivo contendo este tipo de dados poderá ser incluído na página através de um descritor `<EMBED>`. *Exemplo*: `navigator.mimetypes["text/html"].enabledPlugin` contém **null**

PlugIn

Cada plug-in instalado no browser (Netscape) do cliente é um objeto *PlugIn* que possui propriedades contendo o seu nome, nome de arquivo, descrição e vetor de tipos MIME suportados pelo plug-in. Um vetor de todos os plug-ins é obtido através da propriedade `navigator.plugins`. Se um plug-in existe, ele pode ser incluído na página usando um descritor `<EMBED>`.

As propriedades do tipo *PlugIn* estão relacionadas na tabela abaixo:

Propriedade	**Descrição** *(e tipo de dados)*
`name`	*String.* Nome em inglês descrevendo o plug-in.
`description`	*String.* Descrição detalhada do plug-in.
`filename`	*String.* Arquivo do sistema que suporta o plug-in.
`mimetypes`	*Array* de *MimeType*. Vetor com os tipos MIME suportados pelo plug-in.
`length`	*Number.* Quantidade de tipos MIME suportados. Mesmo que `mimetypes.length`

O trecho de código abaixo (arquivo `plugins.html`) pode ser usado para imprimir uma lista com todos os plug-ins instalados em um browser (somente Netscape):

```
<script>
numPlugins = navigator.plugins.length;
document.write("<p>Plug-ins instalados: " + numPlugins);

if (numPlugins > 0) {
   for (i = 0; i < numPlugins; i++)  {
      document.write("<p><b>Nome: </b>" + navigator.plugins[i].name);

      document.writeln("<br><b>Arquivo: </b>");
      document.write(navigator.plugins[i].filename);
      document.write("<br><b>Descrição: </b>");
      document.write(navigator.plugins[i].description);
      document.write("<br><b>Qte. de tipos MIME suportados: </b>");
      document.write(navigator.plugins[i].length);
   }
}
</script>
```

Data-tainting

O modelo de segurança do JavaScript impede que programas enviados por um servidor tenham acesso a propriedades de documentos enviados por outro servidor e assim possam utilizar informação privativa. Com esse modelo, é impossível que uma janela leia, por exemplo, o histórico (objeto `window.history`) de outra janela, caso o documento tenha originado de um servidor diferente.

O browsers Netscape 3.0 em diante, suportam um modelo segurança conhecido como *data-tainting*[5] (marcação de dados). Com ele é possível flexibilizar estas restrições de forma segura, mas isto depende do cliente, que deve ativar o recurso no seu browser, já que ele não e ativado por default. Com o *data-tainting* ativado, uma janela poderá ter acesso a propriedades de outra janela não importando de onde veio o documento, mas o autor da

[5] taint é mancha, nódoa ou marca.

outra janela pode "manchar" (*taint*) certos valores de propriedades que devem ser mantidas privativas e impedir que essas informações sejam passadas sem a permissão do usuário.

Na prática, data-tainting é pouco útil pois os browsers Netscape não são instalados com o recurso habilitado. Para instalá-lo é preciso que o cliente tome a iniciativa de definir uma variável de ambiente no seu sistema. Os browsers Internet Explorer também não suportam o recurso, embora sejam mais flexíveis em relação à segurança. Um programador pode verificar se data-tainting está habilitado, usando o método `taintEnabled()` de `navigator`:

```
if (navigator.taintEnabled()) {
   // instruções se taint está habilitado
}
```

Se tiver a sorte de encontrar um browser com o recurso ativado, poderá passar propriedades entre janelas, tirando as "manchas" de segurança com `untaint(`*`objeto`*`)` ou evitar que outras propriedades sejam lidas sem autorização com `taint(`*`objeto`*`)`. O objeto passado como argumento não é alterado, mas a função retorna um objeto alterado.

Exercício

6.1 Escreva um programa de diagnóstico que imprima na tela um relatório completo sobre o browser do cliente. Primeiro, o programa deverá identificar o fabricante e versão do browser. Se for Netscape 3.0 em diante, deve imprimir uma lista de plug-ins instalados e todos os tipos MIME suportados. Se for outro browser (Netscape 2.0, Internet Explorer ou outro), deve imprimir apenas as informações de fabicante, versão e plataforma.