JavaScript floating objects

Presented by developerWorks, your source for great tutorials

ibm.com/developerWorks

Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

Section 1. Introduction

Should I take this tutorial?

This tutorial is targeted at Web developers who want to create objects, such as images and menus, that hold their place (or float) in the browser window even when the user scrolls or resizes the window.

This tutorial assumes that you are already familiar with HTML in particular and JavaScript in general. An understanding of both dynamic positioning and objects is helpful, but not required; the basics are covered as part of this tutorial. (References to further information on these subjects are also included in the Resources on page 35section at the end of this tutorial.)

What is this tutorial about?

The first floating objects seen on the Web were likely watermarks placed in a top or bottom corner of the browser to identify a Web community such as Geocities. These were semi-transparent and meant to be unobtrusive, but were nevertheless fascinating to users who had never seen an object that didn't move when they scrolled the page.

Today, you can achieve the same effect using JavaScript in the browser -- creating content, placing it, and controlling its location as the user scrolls or resizes the page. This tutorial discusses the process necessary for creating these floaters in such a way that they are available to users of both Netscape (4.x and 6.x) and Microsoft Internet Explorer (5.x and 6.x) browsers.

In addition, this tutorial demonstrates how to allow Internet Explorer 5.5 users to reposition the floater in such a way that it holds its new position even when the page is scrolled or resized.

Tools

Example code snippets are provided throughout the tutorial to provide foundation to the concepts discussed. If you plan to physically work through the examples as you progress, ensure you have the following tools installed and working correctly:

* A text editor: HTML pages and the JavaScript sections within them are simply text. To create and read them, a text editor is all you need.
* Any browser capable of exploiting dynamic positioning using JavaScript: This includes Netscape Navigator 4.7x and 6.1 (available at *http://browsers.netscape.com/browsers/main.tmpl*), and Microsoft Internet Explorer 5.5 (available from *http://www.microsoft.com/windows/ie/downloads/archive/default.asp*). Note that while this tutorial discusses creating floaters that work in all three of the above browsers, the floater can only be repositioned in IE 5.5.

About the author

Nicholas Chase has been involved in Web site development for companies such as Lucent Technologies, Sun Microsystems, Oracle, and the Tampa Bay Buccaneers. Nick has been a high school physics teacher, a low-level radioactive waste facility manager, an online science fiction magazine editor, a multimedia engineer, and an Oracle instructor. More recently, he was the Chief Technology Officer of Site Dynamics Interactive Communications in Clearwater, Fla. He is the author of three books on Web development, including *Java and XML From Scratch* (Que). He loves to hear from readers and can be reached at *nicholas@nicholaschase.com* .

Section 2. The floater

Types of floaters

Floaters are sections of content that remain in a fixed location with respect to a browser page, and have evolved out of several differing user needs.

The first floaters were simply watermarks, partially transparent logos giving the user a visual reference to their virtual "location" in an increasingly confusing World Wide Web. They also gave their creators (such as Geocities) the ability to keep their identity in front of users, fostering more of a sense of overall community than a simple collection of disparate sites.

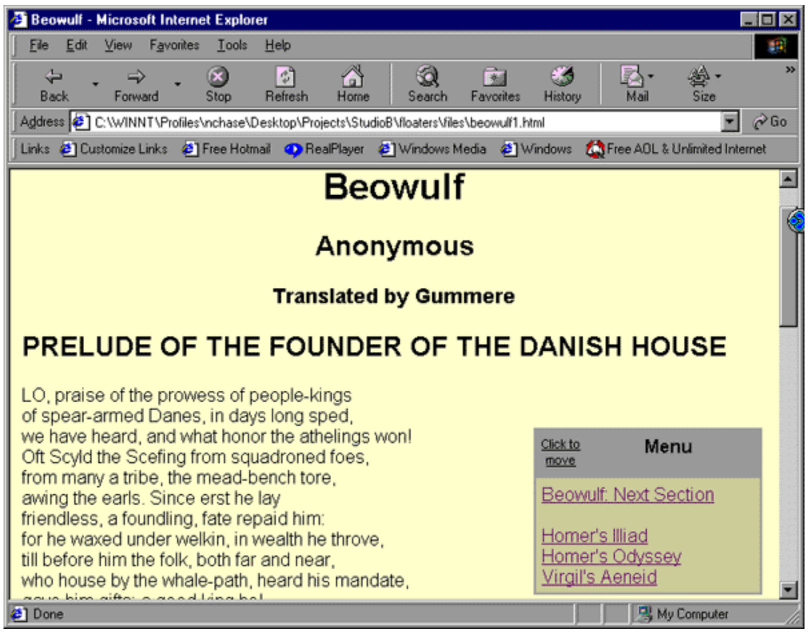
The first watermarks were implemented using ActiveX objects and other complex programming techniques. Now client-side JavaScript has progressed to the point where floaters can be used not just for watermarks, but also for menus and other content that adds value for the user. (For example, some sites host services that allow users to rate a page.)

A floater can be made up of any HTML, including images and links.

The project

This tutorial describes the process of adding a floater to a page that can be viewed in both Netscape and Microsoft browsers.

The floater consists of a menu that allows the user to access the next section of the current content or to request different content. It floats in the lower right-hand corner of the page, as shown below, and adjusts its position if the user scrolls or resizes the window.


Beowulf - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size
Address C:\WINNT\Profiles\nchase\Desktop\Projects\StudioB\floaters\files\beowulf1.html
Go
Links Customize Links Free Hotmail RealPlayer Windows Media Windows Free AOL & Unlimited Internet
Click to move
Menu
Beowulf: Next Section
Homer's Illiad
Homer's Odyssey
Virgil's Aeneid
Done
My Computer

Beowulf

Anonymous

Translated by Gummere

PRELUDE OF THE FOUNDER OF THE DANISH HOUSE

LO, praise of the prowess of people-kings
of spear-armed Danes, in days long sped,
we have heard, and what honor the athelings won!
Oft Scyld the Scefing from squadroned foes,
from many a tribe, the mead-bench tore,
awing the earls. Since erst he lay
friendless, a foundling, fate repaid him:
for he waxed under welkin, in wealth he throve,
till before him the folk, both far and near,
who house by the whale-path, heard his mandate,

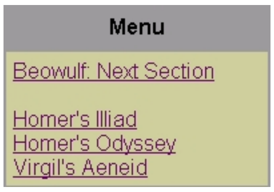
The steps ahead

Making the floater work involves the following steps:

* **Create the floater.** Typically, it's best to add the content to a `div` tag so that it can be referenced by the script without involving the actual content, in case the content changes.
* **Determine the floater position.** Position is determined by the size of the browser window, the size of the floater, and any scrolling the user has done.
* **Update the floater's position.** At regular intervals the browser needs to make sure the floater is still in its assigned position and move it if it's not.

The balance of this tutorial details the concepts and code required to achieve the above.

Create the floater



The actual floater is a simple menu with links to the next section of the epic poem and to several other works, as seen on the left.

A single `div` tag contains the menu, allowing the script to simply refer to this container when moving the content. In order to apply style information (while keeping the code as clean as possible), the page also uses `div` tags to contain the actual content, as shown below:

```
...
 <style type="text/css">
 * { font-family: Arial, Helvetica, "sans serif" }
 #menubar { background-color: #BBBB99;
            width: 100%;
            text-align: center;
            border: 2px solid #BBBB99;
            padding-right: 10px;}
 #menuitems { background-color: #DDDDBB;
              width: 100%;
              border: 2px solid #BBBB99;
              padding-right: 10px;}
 </style>

 ...

 <div id="floater" style="position: absolute; width: 200">
    <div id="menubar"><b>Menu</b></div>
    <div id="menuitems">
       <a href="#">Beowulf: Next Section</a><br />
       <br />
       <a href="#">Homer's Illiad</a><br />
       <a href="#">Homer's Odyssey</a><br />
       <a href="#">Virgil's Aeneid</a>
    </div>
 </div>
 ...
```

As long as the outer `div` (`floater`) is intact, the content in between is irrelevant. It could be HTML text, an image, or even streaming media (though performance would likely suffer due to the constant repositioning the box.) The scripts refer to the outer `div`, and wherever the outer `div` goes, the inner content follows.

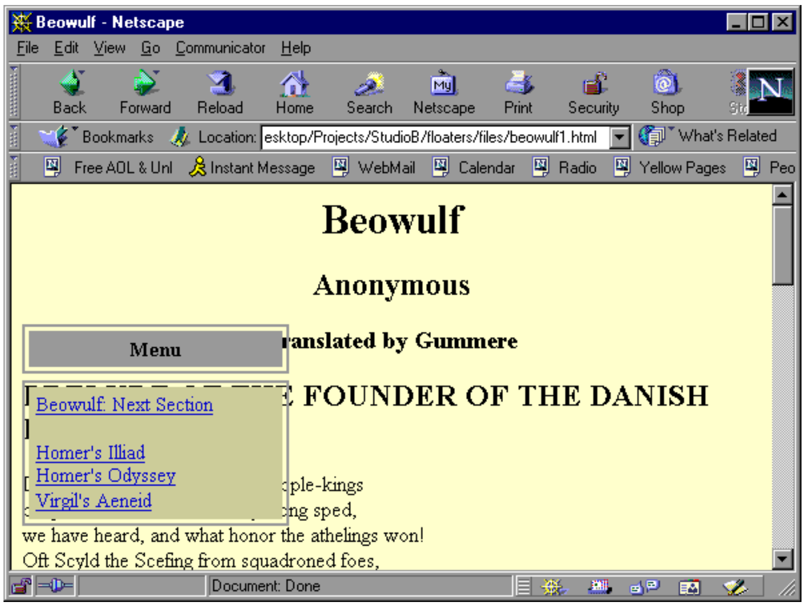
Reference the floater: pre-DOM

In order to manipulate the floater, the script needs a way to reference it. Any HTML element on a page can be referenced as an object that is a property of the `document` object, but the structure of those objects and properties differs depending on the version of the browser referencing it.

Recent browsers, such as Netscape 6.x and Internet Explorer 5.x and 6.x, adhere (to lesser and greater degrees depending on version and platform) to the Document Object Model (DOM) Recommendation from the World Wide Web Consortium (W3C). Older browsers such as Netscape 4.x use a different variation. Technically, this is also a Document Object Model, but to avoid confusion this tutorial refers to the older specification as "pre-DOM."

The pre-DOM variation allows for named objects -- such as the `div` tag -- to use a "dot" notation, whereby an object is named and the property follows separated by a dot. The example below demonstrates how to move the floating menu down 100 pixels in Netscape 4.7.

```
...
 <script type="text/javascript">

    document.floater.top = 100;

 </script>
 ...
```


Beowulf - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop
Bookmarks Location: esktop/Projects/StudioB/floaters/files/beowulf1.html What's Related
Free AOL & Unl Instant Message WebMail Calendar Radio Yellow Pages Peo
Menu
Beowulf: Next Section
Homer's Illiad
Homer's Odyssey
Virgil's Aeneid
Document: Done

Beowulf

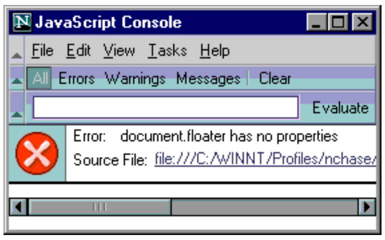
Anonymous

ranslated by Gummere

: FOUNDER OF THE DANISH

ple-kings
ng sped,
we have heard, and what honor the athelings won!
Oft Scyld the Scefing from squadroned foes,

Reference the floater: DOM



Error: document.floater has no properties
Source File: file:///C:/WINNT/Profiles/nchase/

Unfortunately, running the code shown in the previous panel in a DOM-based browser such as Netscape 6.x or IE 5.x results in an error because the browser doesn't recognize `document.floater` as an object.

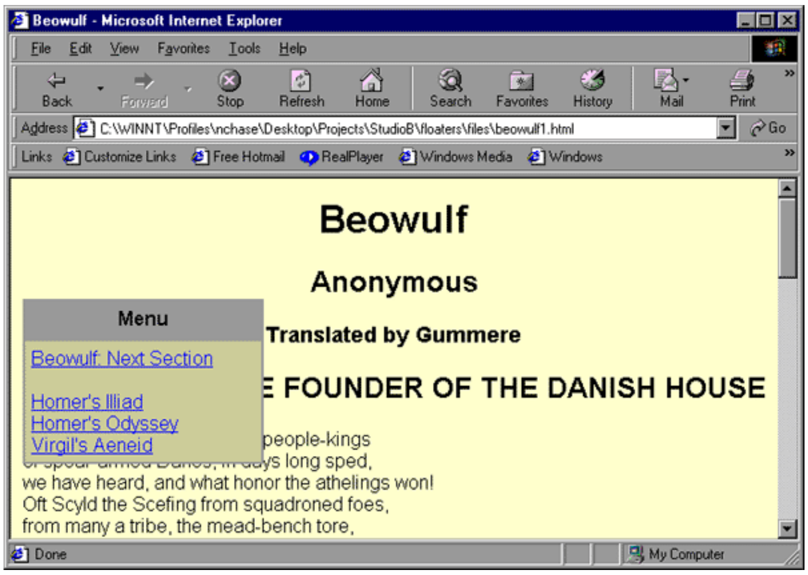
Instead, the `document` object has a method, `getElementById()`, that returns a reference to an object representing the `floater` element, so the following code represents the `div`:

```
document.getElementById("floater")
```

The story doesn't end here, however. In a DOM-based browser, the properties of an element are not part of the object itself, but instead are part of the `style` property of the object, so the same effect as the previous panel can be achieved with the following code:

```
...
 <script type="text/javascript">

    document.getElementById("floater").style.top = 100;

 </script>
 ...
```

Unfortunately, the above approach causes an error in non-DOM browsers because they don't recognize the `getElementById()` function.


Beowulf - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print
Address C:\WINNT\Profiles\nchase\Desktop\Projects\StudioB\floaters\files\beowulf1.html
Go
Links Customize Links Free Hotmail RealPlayer Windows Media Windows
Done
My Computer

Beowulf

Anonymous

Menu

Translated by Gummere

Beowulf: Next Section

E FOUNDER OF THE DANISH HOUSE

Homer's Illiad
Homer's Odyssey
Virgil's Aeneid

people-kings
ys long sped,
we have heard, and what honor the athelings won!
Oft Scyld the Scefing from squadroned foes,
from many a tribe, the mead-bench tore,

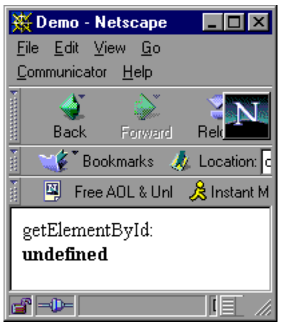
Determine the browser's object model

If code that is required in one browser causes an error in another, how can a single page serve both audiences?

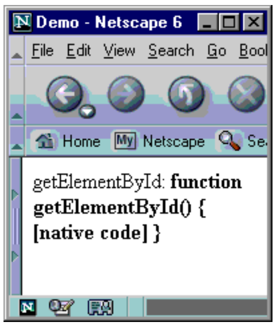
The secret lies in determining the object model in use by the browser accessing the page at any given time. Once the script makes that determination, it can set a variable that controls which code will be executed in any particular situation.

Determining whether a browser is pre-DOM or DOM-based is straightforward. In a DOM-based browser, the `getElementById()` method actually exists as code that's included in the `getElementId` property. Consider the following code executed in Netscape Navigator 4.7, Netscape 6.1, and Microsoft Internet Explorer 5.5, respectively:

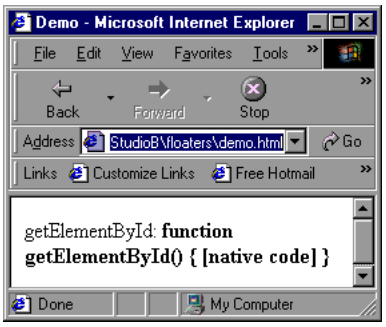
```
document.write('getElementById: <b>'+document.getElementById + '</b>')
```



getElementById: **undefined**



getElementById: **function getElementById() { [native code] }**


Demo - Microsoft Internet Explorer
File Edit View Favorites Tools
Back
Forward
Stop
Address
StudioB\floaters\demo.html
Go
Links
Customize Links
Free Hotmail
Done
My Computer
getElementById: **function getElementById() { [native code] }**

Because `getElementById` is undefined in a pre-DOM browser, it can determine which version of the code to execute:

```
...
 <script type="text/javascript">

    if (document.getElementById) {
       document.getElementById("floater").style.top = 100;
    } else {
       document.floater.top = 100;
    }

 </script>
 ...
```

In this way, the correct code is executed on each browser, and the menu is shifted down 100 pixels in all three.

Reference the floater dynamically

The previous panel showed how to determine which code to execute for a particular browser, but it would be extremely inconvenient to have to do that analysis every time the script needs to set a property for the floater.

A better solution is to determine the object model once. Instead of referencing the object directly, the script designates a variable to represent the floater:

```
...
       <a href="#">Virgil's Aeneid</a>
    </div>
 </div>

 <script type="text/javascript">
    var floaterObj, isDOM
    if (document.getElementById) {
       floaterObj = document.getElementById("floater").style;
       isDOM = true;
    } else {
       floaterObj = document.floater;
       isDOM = false;
    }

    floaterObj.top = 100;
 </script>

 <p>&nbsp;</p>
 <h1 align="center">Beowulf</h1>
 ...
```

This example declares the `floaterObj` variable in the top-level script block so it's available throughout. If `getElementById` returns a value, the script sets `floaterObj` to the DOM version of the object reference. If not, it sets `floaterObj` to the pre-DOM version.

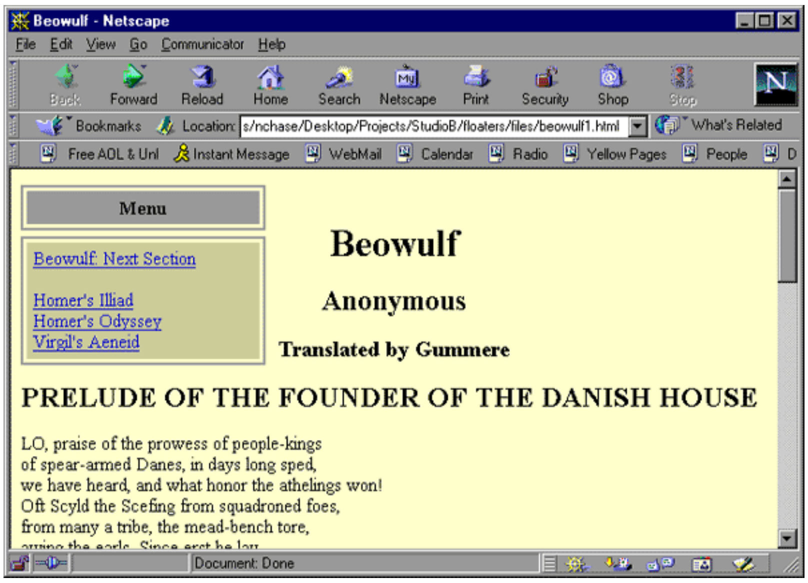
Once the object reference is set, its properties can be referenced as normal, so `floaterObj.top` refers to either `document.floater.top` or `document.getElementById("floater").style.top`, as appropriate. The script can reference all of the object's properties in this way.

The script also sets the `isDOM` flag so it's available later when other decisions need to be made.

Browser compatibility and graceful degradation

The previous panels demonstrate controlling the floater in the three most common browsers, but many others are still in use and may not be caught by this simple test. Obviously, it is advantageous to check for as many browsers as possible using this and similar techniques, but some browsers don't support dynamic positioning or even style sheets. What happens then?

That depends on the structure of the page. This page is specifically structured so that if dynamic positioning is not supported, the user won't know the difference. Of course, the menu won't float in the lower right-hand corner of the page, but it won't obscure the text, either. It simply seems like a menu in the upper left:


Beowulf - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: s/nchase/Desktop/Projects/StudioB/floaters/files/beowulf1.html What's Related
Free AOL & Unl Instant Message WebMail Calendar Radio Yellow Pages People D
Document: Done

Menu

Beowulf: Next Section

Homer's Illiad
Homer's Odyssey
Virgil's Aeneid

Beowulf

Anonymous

Translated by Gummere

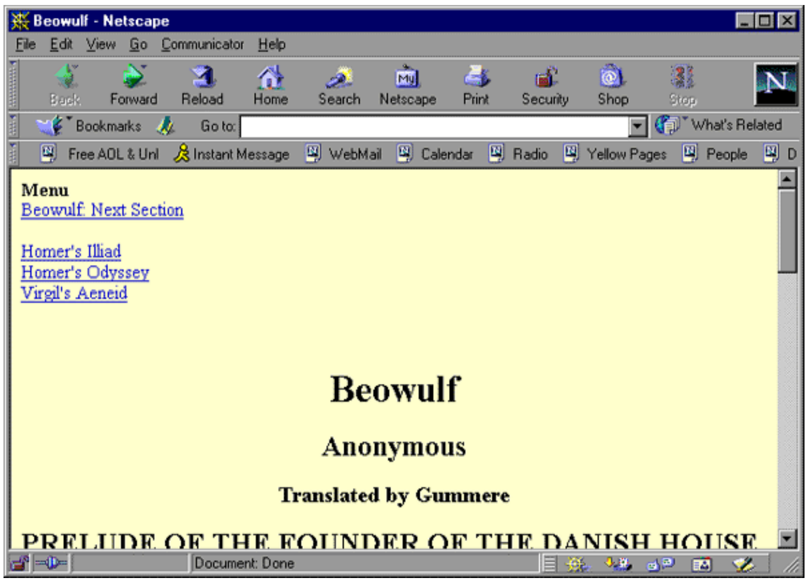
PRELUDE OF THE FOUNDER OF THE DANISH HOUSE

LO, praise of the prowess of people-kings
of spear-armed Danes, in days long sped,
we have heard, and what honor the athelings won!
Oft Scyld the Scefing from squadroned foes,
from many a tribe, the mead-bench tore,

Browser compatibility and graceful degradation (continued)

If neither dynamic positioning nor style sheets are supported at all, the menu sits at the top of the page.

In either case, the user's experience is not negatively affected by the floater. If the effect is supported, it will be seen. If not, the user will be none the wiser.


Beowulf - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Go to: What's Related
Free AOL & Unl Instant Message WebMail Calendar Radio Yellow Pages People D
Document: Done

Menu
Beowulf: Next Section

Homer's Illiad
Homer's Odyssey
Virgil's Aeneid

Beowulf

Anonymous

Translated by Gummere

PRELUDE OF THE FOUNDER OF THE DANISH HOUSE

Section 3. Positioning content

Positioning schemes

On a typical Web page, items are rendered on the page in accordance with the *normal flow*. The browser places elements on the page as it encounters them, either on separate lines or one after the other on a single line until the line is full (then it starts a new line), depending on the type of element.

Dynamic positioning allows programmers to alter the position of an element, either placing it in a specific location or adjusting it relative to its position within the relative flow, depending on the *positioning scheme*.

The positioning scheme determines how an item is placed on the page, and is set via the `position` property for an element. The position property can have four possible values:

* static: These elements are laid out according to their position within the normal flow.
* fixed: These elements are laid out in a specific position and remain there even if the user scrolls, like a floater. Unfortunately, this scheme isn't supported in current browsers so the effect must be created via scripting.
* relative: These elements are positioned relative to their normal place within the flow. For example, an element may be shifted up 5 pixels to simulate a superscript. Subsequent content is laid out as though the item were in position according to the normal flow.
* absolute: These elements are actually removed from the normal flow altogether, and placed relative to a reference point, as discussed in the next panel.

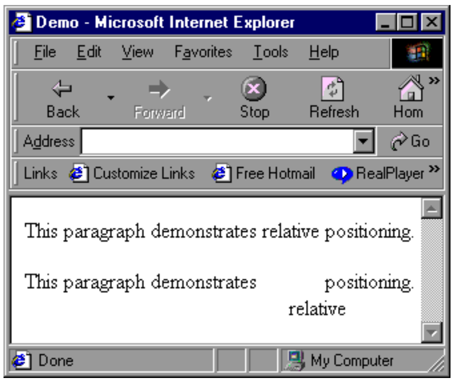
These positioning schemes, along with other CSS properties, enable dynamic positioning of elements.

Positioning and flow

All dynamic positioning is accomplished using the `top`, `left`, `bottom`, and `right` styling properties for an element. The `top` and `left` properties are the most commonly used, referring to the amount of space between the top of the element and the reference point (in other words, the distance the element is shifted downwards) and the distance between the left edge and the reference point (in other words, the distance the element is shifted to the right), respectively.

The reference point depends on the positioning scheme. For a relatively positioned element, the reference point is the position the element takes within the normal flow. Consider this example:

```
...
 <div>

 <p>This paragraph demonstrates relative positioning.</p>

 <p>This paragraph demonstrates
 <span style="position:relative;top:20px;left:20px">relative</span> positioning.</p>

 </div>
 ...
```

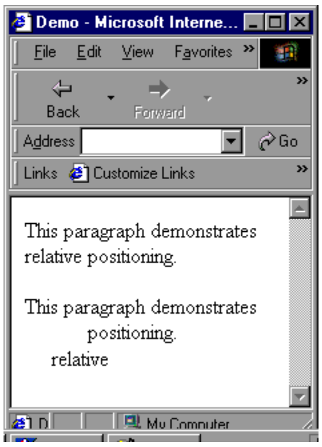

Demo - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Hom
Address Go
Links Customize Links Free Hotmail RealPlayer
Done My Computer

This paragraph demonstrates relative positioning.

This paragraph demonstrates positioning.
relative

The span containing the word "relative" is repositioned relative to its normal position within the flow. Because the value of the top property is 20 pixels, the span is shifted downwards 20 pixels. Similarly, the value of 20 pixels for the left property shifts it 20 pixels to the right. To shift the element up and to the left -- rather than down and to the right -- use negative values.

Positioning and flow (continued)



This paragraph demonstrates relative positioning.

This paragraph demonstrates
positioning.
relative

It's important to notice that when using relative positioning, the rest of the page is laid out as though the element were still in its original position, as illustrated by the word "positioning" in the image to the left. Also, if the layout of the page were to change -- if the page were resized, for example -- the position of the element may change, as seen here.

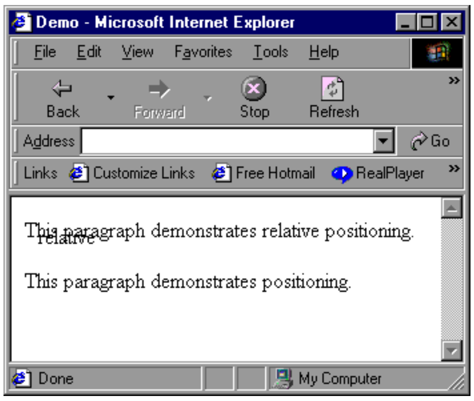
Both of these aspects of relative positioning make it unsuitable for positioning the floater.

Absolute positioning

With absolute positioning, an element can be placed in a specific location, and in most cases, no matter what happens to the page it's not going to move. For example:

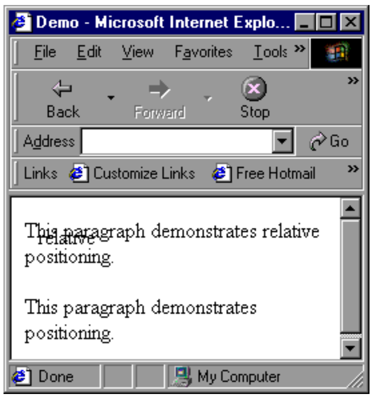
```
...
 <div>

 <p>This paragraph demonstrates relative positioning.</p>

 <p>This paragraph demonstrates
```

```
<span style="position: absolute; top: 20px; left: 20px">relative</span>
positioning.</p>

</div>
...
```


Demo - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address Go
Links Customize Links Free Hotmail RealPlayer
Done My Computer

This paragraph demonstrates relative positioning.

This paragraph demonstrates positioning.


Demo - Microsoft Internet Explo...
File Edit View Favorites Tools
Back Forward Stop
Address Go
Links Customize Links Free Hotmail
Done My Computer

This paragraph demonstrates relative positioning.

This paragraph demonstrates positioning.

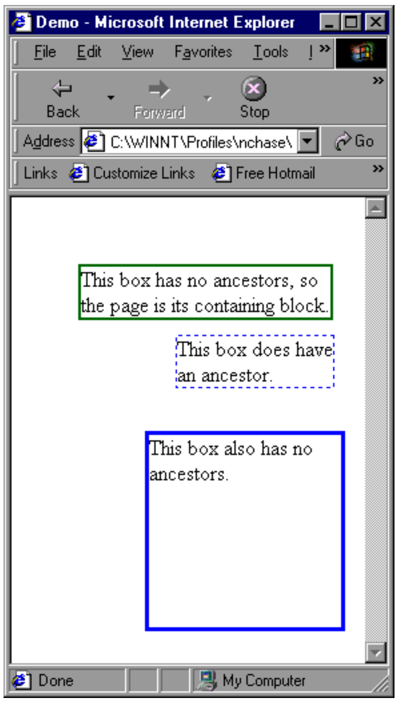
Notice that in both cases, the word "relative" was placed 20 pixels down from the upper right-hand corner of the window, even though other text was already there. Notice also that the `span` was completely removed from the flow of the page. The word "positioning" that follows it is placed immediately next to the word "demonstrates", as though the span wasn't ever there -- because it wasn't.

Determining the reference point

The `top` and `left` properties determine the distance of an element from the reference point, but what determines the reference point?

For an absolutely positioned element, the reference point is always the top left corner of its *containing block*. The containing block for an element is the closest block level element (such as a `div` or p) enclosing it that is itself dynamically positioned. If there is no such block, the page itself acts as the containing block.

Consider this example:


Demo - Microsoft Internet Explorer
File Edit View Favorites Tools
Back Forward Stop
Address C:\WINNT\Profiles\nchase\ Go
Links Customize Links Free Hotmail
Done My Computer

This box has no ancestors, so the page is its containing block.

This box does have an ancestor.

This box also has no ancestors.

```
<div style="position: absolute; top: 50px;
                left: 50px; border: 2px solid green">
 This box has no ancestors, so the page is
 its containing block.
 </div>

 <div style="position: absolute; top:175px;
             left: 100px;
             border: 3px solid blue;
             width: 150px; height: 150px">
    This box also has no ancestors.

    <div style="position: absolute; top: -75px;
                left: 20px;
                border: 1px dashed blue">
       This box does have an ancestor.
    </div>
 </div>
```

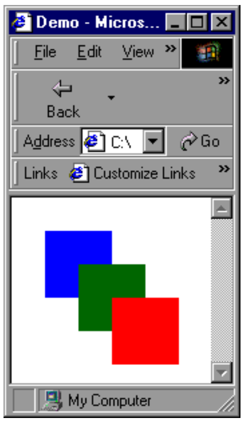
In this example, both the solid green and the solid blue boxes use the page as their containing block, so their measurements are taken from the top left-hand corner of the page. The dashed box, on the other hand, is contained within the solid blue box, and its measurements are taken from the upper left-hand corner of the solid blue box.

Determining the containing block (and thus the reference point) can get a good deal more complicated, so be aware of where the floater is located within the page code in order to prevent problems later. If the dashed blue box were moved out from within the div to the page itself, it would disappear altogether because it would be 75 pixels above the top of the window.

For more links to more information on determining the containing block, see the Dynamic Positioning tutorial listed in the Resources on page 35.

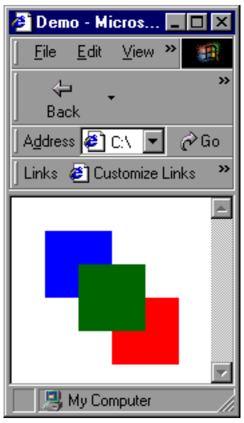
Layering elements

Once elements are removed from the normal flow, they have the potential to overlap with the most recently placed element on top:



```
<div style="position: absolute;
             width: 50px; height: 50px;
             top: 25px; left: 25px;
             background-color: blue"></div>
 <div style="position: absolute;
             width: 50px; height: 50px;
             top: 50px; left: 50px;
             background-color: green"></div>
 <div style="position: absolute;
             width: 50px; height: 50px;
             top: 75px; left: 75px;
             background-color: red"></div>
```

To control the layering, add the z-index property. The higher the z-index, the closer to the "front" the box appears:



```
<div style="position: absolute;
             width: 50px; height: 50px;
             top: 25px; left: 25px;
             background-color: blue;
             z-index: 1"></div>
 <div style="position: absolute;
             width: 50px; height: 50px;
             top: 50px; left: 50px;
             background-color: green;
             z-index: 3"></div>
 <div style="position: absolute;
             width: 50px; height: 50px;
             top: 75px; left: 75px;
             background-color: red;
             z-index: 2"></div>
```

The floater should always appear "in front of" the rest of the page, so be sure to assign it a high `z-index`:

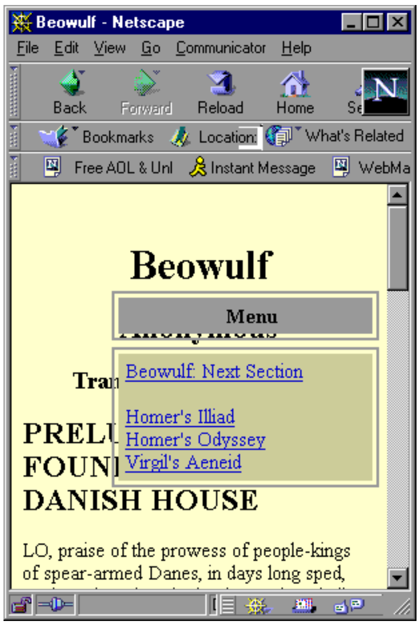
```
<div id="floater" style="position: absolute; width: 200; z-index: 1000">
    <div id="menubar"><b>Menu</b></div>
 ...
 </div>
 ...
```

This value ensures that it is rendered in front of not only elements with a lower `z-index`, but also elements with no `z-index` at all.

Control the position via scripting

Because positioning is controlled through the `top` and `left` styling properties of an element, scripting the floater position is straightforward.

Reference the floater dynamically on page 9 shows the creation of a common reference to the floater, `floaterObj`. Set the `top` and `left` properties on the object to move it around the page:


Beowulf - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home
Bookmarks Location: What's Related
Free AOL & Unl Instant Message WebMa
Beowulf
Menu
Beowulf: Next Section
Homer's Illiad
Homer's Odyssey
Virgil's Aeneid
PREL
FOUN
DANISH HOUSE
LO, praise of the prowess of people-kings
of spear-armed Danes, in days long sped,

```
...
       <a href="#">Virgil's Aeneid</a>
    </div>
 </div>

 <script type="text/javascript">
 var floaterObj, isDOM
 if (document.getElementById) {
    floaterObj =
    document.getElementById("floater").style;
    isDOM = true;
 } else {
    floaterObj = document.floater;
    isDOM = false;
 }

 floaterObj.top = 75;
 floaterObj.left = 75;
 </script>
```

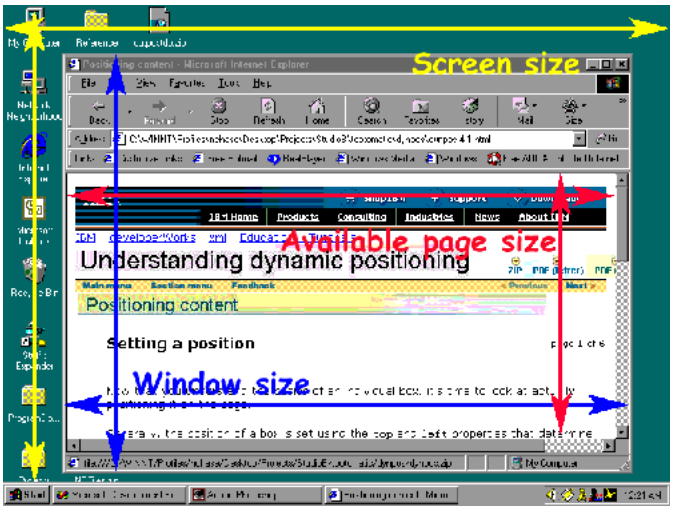
```
<p>&nbsp;</p>
<h1 align="center">Beowulf</h1>
...
```

Positioning the floater is simple; the difficult part is knowing *where* to position it. The next section explores the code necessary to adjust the position of the floater based on the size of the window.

Section 4. Detecting screen position

Types of screen measurements

In placing any content on a page, various measurements come into play.



Some, like the size of the screen itself (the yellow dimensions in the screenshot above), are not necessarily important for the actual placement of content, but can give developers an idea of the user's available screen real estate. From there, developers can make decisions about window size, allowing for choices between different layouts or, in the case of this tutorial, floater position.

Others, like the available page size, are much more useful in placing the floater. The available page size (the red dimensions) consists of the window size (blue dimensions) minus the actual interface, including buttons, status bars, and scroll bars.

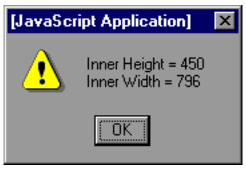
To place the floater in the lower right-hand corner, the script needs to know the available page width and height, as well as the size of the floater itself. Subtracting the floater height and width from the page height and width yields the coordinates for the floater.

Built-in objects such as `window`, `screen`, and `document` make these measurements available to scripts, but like the pre-DOM/DOM variation discussed in Section 2, the way in which they're accessed differs depending on the browser implementation.

Determine page size: Netscape

While the `floater` object is referenced according to the DOM/pre-DOM distinction, available page size is determined by browser type.

In Netscape browsers, the `window` object makes the `innerHeight` and `innerWidth` properties available. These measurements include the actual content area of the window, as well as the width of any scroll bars:


[JavaScript Application]

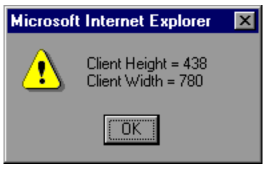
Inner Height = 450
Inner Width = 796

OK

```
...
    alert('Inner Height = ' + window.innerHeight
          + '\n' +
          'Inner Width = ' + window.innerWidth);
 ...
```

Determine page size: Internet Explorer

Determining the page size in IE is a bit more precise due to the `clientHeight` and `clientWidth` properties of the `document.body` object, which measure only the actual content area of the page:


Microsoft Internet Explorer

Client Height = 438
Client Width = 780

OK

```
...
    alert('Client Height = ' + document.body.clientHeight
          + '\n' +
          'Client Width = ' + document.body.clientWidth);
 ...
```

The `clientHeight` and `clientWidth` properties take the scroll bars into account, so they need not be considered when determining placement.

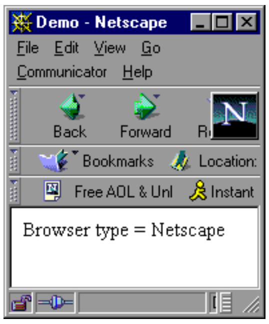
Determine the browser type

In order to correctly configure the float position for the variances in browser type, another test is needed. The `isDOM` variable won't help, because Netscape 6 is DOM-based, but uses the Netscape format.

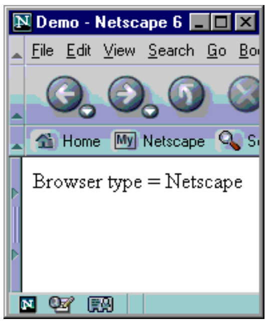
Instead, the `navigator` object provides information as to the type of browser, using the `appName` property. For example:

```
   document.write('Browser type = '+navigator.appName)
```

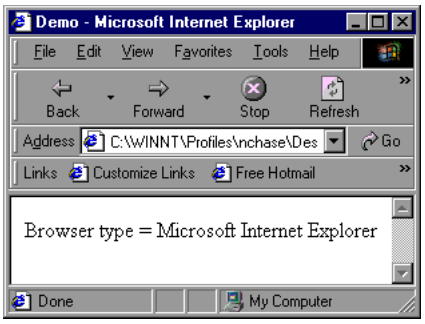
produces:



Browser type = Netscape



Browser type = Netscape


Demo - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Forward
Stop
Refresh
Address
C:\WINNT\Profiles\nchase\Des
Go
Links
Customize Links
Free Hotmail
Done
My Computer

Browser type = Microsoft Internet Explorer

Using this information, determine which measures to take for the page height and width. Remember to take into account possible scroll bars in Netscape browsers:

```
...
    floaterObj.left = 400;

    var isNS, pageHeight, pageWidth

    if (navigator.appName == 'Netscape') {
       isNS = true;
       pageHeight = window.innerHeight - 20;
       pageWidth = window.innerWidth - 20;
    } else {
       isNS = false;
       pageHeight = document.body.clientHeight;
       pageWidth = document.body.clientWidth;
    }

 </script>

 <p>&nbsp;</p>
 <h1 align="center">Beowulf</h1>
 ...
```

As with the `isDOM` flag, the script sets the `isNS` flag for later use.

Adjust for floater size

The script used in this tutorial positions the floater in the lower right-hand corner of the browser window, so in order to determine the exact placement coordinates, the size of the floater itself must be taken into consideration.

In an ideal world, the height and width of the `floater` object could be determined dynamically using `floaterObj.height` and `floaterObj.width`. Unfortunately, unless the `style` attribute specifies these values (or they're specified in a script), Netscape 6.x and IE 5.x won't return a value for them. A developer could work around this problem by simply setting those values, but even then Netscape 4.x won't return any values.

The only answer is to set the values from within the script:

```
...
       pageWidth = document.body.clientWidth;
    }

    var floaterWidth, floaterHeight
    floaterWidth = 200;
    floaterHeight = 175;

 </script>

 <p>&nbsp;</p>
 <h1 align="center">Beowulf</h1>
 ...
```

Place the floater

Finally, the script has all of the information it needs to initially place the floater:

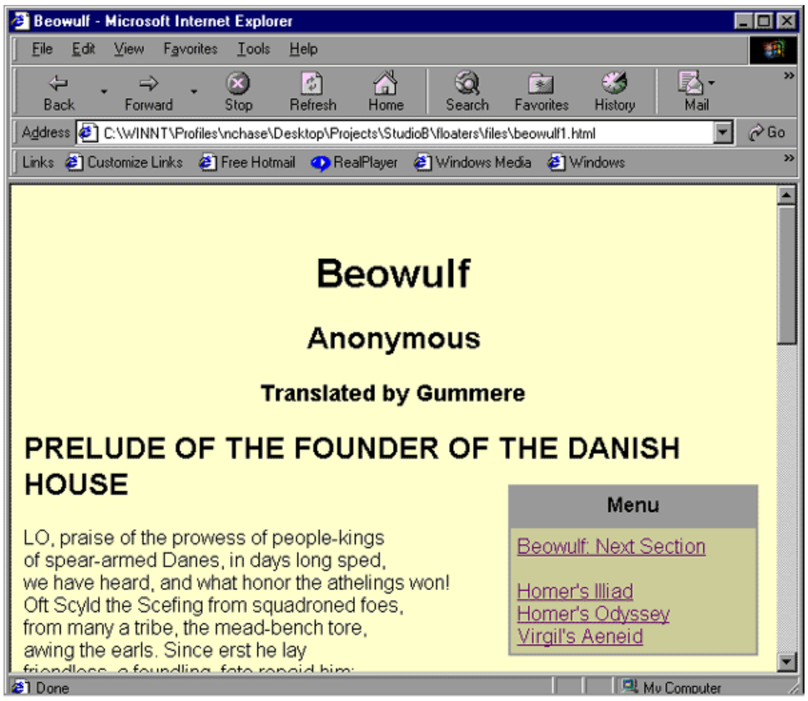
```
...
    floaterWidth = 200;
    floaterHeight = 135;

    var paddingX, paddingY, floaterX, floaterY

    paddingX = 15;
    paddingY = 15;

    floaterX = pageWidth - floaterWidth - paddingX;
    floaterY = pageHeight - floaterHeight - paddingY;

    floaterObj.top = floaterY;
    floaterObj.left = floaterX;

 </script>

 <p>&nbsp;</p>
 <h1 align="center">Beowulf</h1>
 ...
```

First, the script sets values for the padding that sits between the edge of the floater and the edge of the window. It then determines the actual placement by subtracting the floater width and height, and the padding, from the available space. Finally, the script sets the actual properties of the object to match these positions.

Initial results

At this point, the initial script is complete. Save the file and refresh the browser. The floater should appear in the lower right-hand corner.

Of course, resizing or scrolling the browser results in the floater winding up out of position. The next section details the scripting necessary to keep the floater in the correct position no matter what the user does to the window.


Beowulf - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail
Address C:\WINNT\Profiles\nchase\Desktop\Projects\StudioB\floaters\files\beowulf1.html
Go
Links Customize Links Free Hotmail RealPlayer Windows Media Windows
Done
My Computer

Beowulf

Anonymous

Translated by Gummere

PRELUDE OF THE FOUNDER OF THE DANISH HOUSE

LO, praise of the prowess of people-kings
of spear-armed Danes, in days long sped,
we have heard, and what honor the athelings won!
Oft Scyld the Scefing from squadroned foes,
from many a tribe, the mead-bench tore,
awing the earls. Since erst he lay
friendless, a foundling, fate repaid him:

Menu

Beowulf: Next Section

Homer's Illiad
Homer's Odyssey
Virgil's Aeneid

Section 5. Reacting to the user

Create functions

The current page places the floater in the correct initial position, but does not keep the floater in place if the user scrolls or resizes the window. This section demonstrates how to take care of those issues.

All of the code so far is included in the main script block, which means it's executed when the page is loaded. While that has been convenient up to now, it makes it impossible to attach actions to events such as the resizing of the window. The first step in reacting to the user, therefore, is to modularize the code into functions:

```
...
 #menuitems { background-color: #DDDDBB;
              width: 100%;
              border: 2px solid #BBBB99;
              padding-right: 10px;}
 </style>

 </head>
 <body style="background-color: #FFFFDD" onload="setUp()">

 <div id="floater" name="floater"
      style="position: absolute; width: 200; z-index: 1000">
    <div id="menubar"><b>Menu</b></div>
    <div id="menuitems">
       <a href="#">Beowulf: Next Section</a><br />
       <br />
       <a href="#">Homer's Illiad</a><br />
       <a href="#">Homer's Odyssey</a><br />
       <a href="#">Virgil's Aeneid</a>
    </div>
 </div>

 <script type="text/javascript">

    var isDOM, floaterObj
    var isNS, pageHeight, pageWidth
    var floaterWidth, floaterHeight
    var paddingX, paddingY, floaterX, floaterY

    function setUp() {

       if (document.getElementById) {
          isDOM = true;
          floaterObj = document.getElementById("floater").style;
       } else {
          isDOM = false;
          floaterObj = document.floater;
       }

       if (navigator.appName == 'Netscape') {
          isNS = true;
       } else {
          isNS = false;
       }

       floaterWidth = 200;
       floaterHeight = 135;

       paddingX = 15;
       paddingY = 15;

       refreshValues();
    }

    function refreshValues() {
       if (isNS) {
          pageHeight = window.innerHeight - 20;
          pageWidth = window.innerWidth - 20;
       } else {
          pageHeight = document.body.clientHeight;
          pageWidth = document.body.clientWidth;
       }

       floaterX = pageWidth - floaterWidth - paddingX;
       floaterY = pageHeight - floaterHeight - paddingY;

       placeFloater();

    }

    function placeFloater() {

       floaterObj.top = floaterY;
       floaterObj.left = floaterX;

    }

 </script>

 <p>&nbsp;</p>
 <h1 align="center">Beowulf</h1>
 <h2 align="center">Anonymous</h2>
 ...
```

First, notice that all variables are declared globally so they can be used between functions. The rest of the code has been broken into three functions. The first, `setUp()`, is called when the page is loaded, and initializes the flags and the `floaterObj` object. It also provides a place to set the floater size and padding values used later. Finally, it calls the `refreshValues()` function.

The `refreshValues()` function takes care of checking the size of the window and performing calculations to determine the position of the floater. Once it determines these values, it calls the `placeFloater()` function.

The `placeFloater()` function actually moves the floater into position. Because this function will be called most often, the fewer actions it performs, the better.

Now the page is ready to adapt to the user.

Resize the window

The floater is now placed in the proper position, but if the window is resized or scrolled, the floater doesn't adapt. Ideally, the page would refresh the values (and thus move the floater) if the window is resized.

The `window` object provides a way to make this happen. The `onresize` property allows a developer to specify a function that executes when the page is resized (thus firing the `onresize` event):

```
...
    function setUp() {
 ...
       paddingX = 15;
       paddingY = 15;

       window.onresize = refreshValues
```

```
      refreshValues();
   }

   function refreshValues() {

      if (isNS) {
         pageHeight = window.innerHeight - 20;
         pageWidth = window.innerWidth - 20;
      } else {
         pageHeight = document.body.clientHeight;
         pageWidth = document.body.clientWidth;
      }

      floaterX = pageWidth - floaterWidth - paddingX;
      floaterY = pageHeight - floaterHeight - paddingY;

      placeFloater();

   }
...
```

Now when the window is resized, the `refreshValues()` function is called. That function notes the new size of the window and resets the position variables. It then calls `placeFloater()` to move the floater to the new position.

Scrolling: Netscape

Much more common than resizing the window is scrolling the page up and down or (to a lesser extent) left and right. Remember, the floater uses the top left corner of the page, and not the top left corner of the window, as its reference point. This mean that when the user scrolls, the floater (currently) moves with the page.

The `placeFloater()` function needs to compensate for scrolling if the floater is going to remain in its original position relative to the window. Once again, however, implementation is split between the Netscape and Microsoft camps.

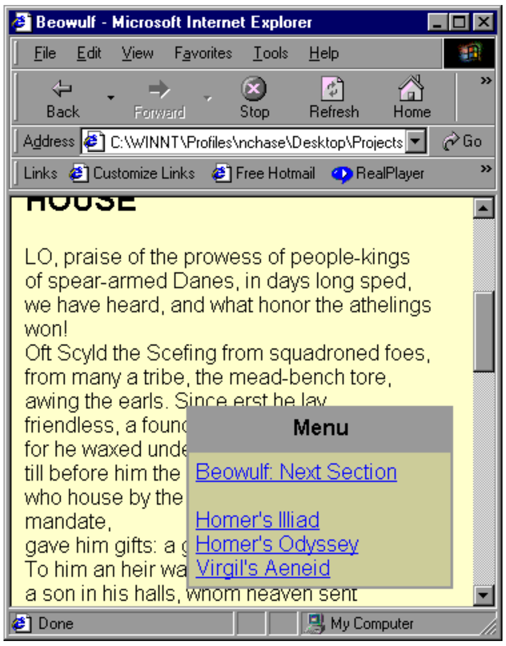
Netscape browsers use the `window.pageXOffset` and `window.pageYOffset` properties:

```
...
   var paddingX, paddingY, floaterX, floaterY
   var xOffset, yOffset
...
   function placeFloater() {

      xOffset = window.pageXOffset;
      yOffset = window.pageYOffset;

      floaterObj.top = floaterY + yOffset;
      floaterObj.left = floaterX + xOffset;

   }
...
```

Place the offsets in the `placeFloater()` function because they need to be determined much more frequently than the other variables that are dependent on window size. This way, all of the variables won't have to be recalculated every time the floater is repositioned.

Scrolling: Internet Explorer

Internet Explorer stores the page offset information not in the `window` object, but in the `document.body` object:



HOUSE

LO, praise of the prowess of people-kings
of spear-armed Danes, in days long sped,
we have heard, and what honor the athelings
won!
Oft Scyld the Scefing from squadroned foes,
from many a tribe, the mead-bench tore,
awing the earls. Since erst he lay
friendless, a found
for he waxed unde
till before him the
who house by the
mandate,
gave him gifts: a
To him an heir wa
a son in his halls, whom heaven sent

```
...
 function placeFloater() {

    var xOffset, yOffset
    if (isNS) {
       xOffset = window.pageXOffset;
       yOffset = window.pageYOffset;
    } else {
       xOffset = document.body.scrollLeft;
       yOffset = document.body.scrollTop;
    }

    floaterObj.top = floaterY + yOffset;
    floaterObj.left = floaterX + xOffset;

 }
 ...
```

Both browsers now get accurate values, so scrolling and then resizing show the shift.

Auto-update the position

All that remains at this point is instructing the browser to automatically update the positioning so that when the page is scrolled, the floater moves automatically.

Because there is no universally supported "onscroll" event, the only option is to have the browser check the position periodically. The `window` object's `setInterval()` method makes this straightforward by enabling you to designate code to execute at specific intervals, which are measured in milliseconds:

```
...
    function setUp() {
 ...
       paddingX = 15;
       paddingY = 15;

       window.onresize = refreshValues
```

```
        window.setInterval("placeFloater()", 100)

        refreshValues();
    }
 ...
```

With this code in place, every tenth of a second the browser executes the `placeFloater()` function. Because that function includes the offset values, the floater always winds up in the right place.

It stands to reason that the shorter the interval, the smoother the positioning of the floater, but remember that with the increased execution of `placeFloater` comes increased overhead. No user appreciates a Web site that brings his or her computer to its knees.

Section 6. Moving the floater

The algorithm

This section details the code needed to allow the user to reposition the floater. Only IE 5.5 users will be able to move the floater, but other users will not be aware that they're missing anything.

The process of moving the floater involves the following steps:

1. The user clicks a button to turn on movement.
2. The button changes to indicate that movement is on.
3. The user clicks the new location.
4. A new offset is calculated for subsequent executions of the `placeFloater()` function.
5. The floater is repositioned.
6. Movement is turned off.
7. The button goes back to its original state.

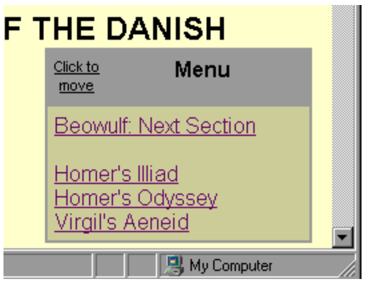
Create the button

The first step is to create a button that can only seen by users when their browser is capable of executing the underlying code. To do that, create an empty `div` element and then use scripting to add content.

```
...
 #menuitems { background-color: #DDDDBB;
              width: 100%;
              border: 2px solid #BBBB99;
              padding-right: 10px;}
 #movebutton { font-size: 8pt;
               float: left;
               text-decoration: underline }
 </style>

 </head>
 <body style="background-color: #FFFFDD" onload="setUp()" onclick="setNewPos()">

 <div id="floater" name="floater"
      style="position: absolute; width: 200; z-index: 1000">
    <div id="menubar">
       <div id="movebutton"></div>
       <b>Menu</b>
    </div>
    <div id="menuitems">
 ...
    </div>
 </div>

 <script type="text/javascript">
 ...
    function setUp() {
 ...
       window.onresize = refreshValues
       window.setInterval("placeFloater()", 100)

       refreshValues();

       if ((navigator.appName.indexOf('Internet Explorer') != -1)
            && (navigator.appVersion.indexOf('5.5') != -1)) {
          document.getElementById("movebutton").innerHTML =
                                           'Click to<br />move';
       }
    }

    function refreshValues() {
 ...
```

The logic behind the movebutton code

Because the `movebutton div` tag is created with no content, users do not see it unless the script changes the `innerHTML` property. The script checks the browser version after setting the initial values and placing the floater within the `setUp()` function. This version changes the content only if the browser's shows as Internet Explorer version 5.5. More extensive browser detection is possible, but should probably be broken out into a separate function, or at least a flag.



In this way, if the browser is capable of moving the floater, the user sees the button. If not, the user not only won't see it, but also won't have anything to click. Because browser detection ensures that only those browsers that can execute the code successfully will be allowed to execute it at all -- an inappropriate user won't be able to initiate the movement process -- further detection won't be necessary.

Turn on movement

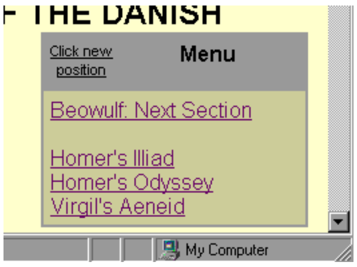
The button itself needs to provide a way to turn on scripting when it's clicked:

```
...
    <div id="menubar">
       <div id="movebutton" onclick="setMoveOn()"></div>
       <b>Menu</b>
    </div>
 ...
 <script type="text/javascript">
 ...
    var xOffset, yOffset
    var isMoving

    function setUp() {
 ...
       if ((navigator.appName.indexOf('Internet Explorer') != -1)
            && (navigator.appVersion.indexOf('5') != -1)) {
          document.getElementById("movebutton").innerHTML =
                                             'Click to<br />move';
       }

       isMoving = false;
    }
 ...
    function placeFloater() {
       if (isNS) {
          xOffset = window.pageXOffset;
```

```
         yOffset = window.pageYOffset;
      } else {
         xOffset = document.body.scrollLeft;
         yOffset = document.body.scrollTop;
      }

      floaterObj.top = floaterY + yOffset;
      floaterObj.left = floaterX + xOffset;
   }

   function setMoveOn() {
      isMoving = true;
      document.getElementById("movebutton").innerHTML=
                                   'Click new<br />position';
   }

 </script>
 ...
```

Create the isMoving variable

The user will (ideally) be clicking the page for many more reasons than moving the menu around. The script must know when to ignore those clicks, and when to reposition the floater. The global `isMoving` variable serves this purpose.



Non-moving browsers always see the flag set to `false`, because the assignment is made within the `setUp()` function. If, on the other hand, the user clicks the `movebutton`, the value of `isMoving` changes to `true`, and the content of the `movebutton` changes to reflect the new status. In addition, the changed text provides the user instructions on what to do next.

Register the new location

Once movement is turned on, the user needs to indicate the menu's new position. But what does he or she click? It's likely that the point the user clicks has no content.

To solve this problem, add the `onclick` event attribute to the `body` itself. This way, no matter where the user clicks, the script executes. The `clientX` and `clientY` properties of the `window.event` object hold the coordinates of that mouseclick:

```
...
 </head>
 <body style="background-color: #FFFFDD" onload="setUp()"
         onclick="setNewPos()">

 <div id="floater" name="floater"
      style="position: absolute; width: 200; z-index: 1000">
 ...
 function setMoveOn() {
    isMoving = true;
    document.getElementById("movebutton").innerHTML=
                              'Click new<br />position';

 }

 function setNewPos() {

    if (isMoving) {
       clickedX = window.event.clientX;
       clickedY = window.event.clientY;
    }

 }

 </script>
 ...
```

Unless the `isMoving` variable has been set to `true` by the `setMoveOn()` function, the `setNewPos()` function does nothing. If `isMoving` has been set to `true`, the script records the clicked location. Note that these are local variables. They are not required anywhere else in the script.

Create the new offset

It would be simple to move the floater to the new location using the values of `clickedX` and `clickedY`, but 100 milliseconds later the floater would be placed right back in the lower right-hand corner by the `placeFloater()` function.

To resolve this problem, create a pair of offsets representing the distance between the floater and the right and bottom edges of the window:

```
...
    var isMoving
    var newXOffset, newYOffset

    function setUp() {
 ...
       paddingX = 15;
       paddingY = 15;

       newXOffset = 0;
       newYOffset = 0;

       window.onresize = refreshValues
       window.setInterval("placeFloater()", 100)

       refreshValues();

       if ((navigator.appName.indexOf('Internet Explorer') != -1)
            && (navigator.appVersion.indexOf('5') != -1)) {
          document.getElementById("movebutton").innerHTML =
                                             'Click to<br />move';
       }

       isMoving = false;
    }

 ...
    function placeFloater() {
       if (isNS) {
          xOffset = window.pageXOffset;
          yOffset = window.pageYOffset;
       } else {
          xOffset = document.body.scrollLeft;
          yOffset = document.body.scrollTop;
       }
```

```
      floaterObj.top = floaterY + yOffset - newYOffset;
      floaterObj.left = floaterX + xOffset - newXOffset;

   }
...
function setNewPos() {

   if (isMoving) {
      clickedX = window.event.clientX;
      clickedY = window.event.clientY;

      newXOffset = pageWidth - clickedX - floaterWidth;
      newYOffset = pageHeight - clickedY - floaterHeight;

      placeFloater();
   }

}
...
```

The offsets are global, as several functions use them. Set their values as part of the `setNewPos()` function, calculating it as the width of the page minus the number of pixels from the left (or top) the user clicked, minus the size of the floater. What remains is the number of pixels between the right and bottom edges of the floater and the right and bottom edges of the window.

Subtract this value from the position `placeFloater()` would normally use and the floater will move to its new position.

The bubble up effect

There's just one flaw with this plan: The event model for IE registers the original "Click to move" and executes the `setMoveOn()` function, but that click is also over the body of the document, which means that same event also triggers the execution of the `setNewPos()` function. Unfortunately, this function then moves the floater just slightly, as though the user had intended to move it to the location of the button. This effect is known as *bubbling up*.

In the bubble up model, events work their way through the hierarchy from specific (such as the `div`) to the general (such as the `body`), giving each possible object a chance to act on them. In this case, however, the event must be stopped at the button for everything to work properly.

Future versions of IE may implement an `event.cancelbubble()` method, but for now the script needs a flag to keep track of whether the user is really clicking a new location, or the event has just bubbled up to cause the execution of the `setNewPos()` function:

```
...
    var newXOffset, newYOffset
    var justClicked

    function setUp() {
 ...
       isMoving = false;
       justClicked = false;
    }
 ...
    function setMoveOn() {

       isMoving = true;
       document.getElementById("movebutton").innerHTML=
                                'Click new<br />position';
       justClicked = true;

    }

    function setNewPos() {

       if (isMoving) {
          if (justClicked) {
             justClicked = false;
          } else {
             clickedX = window.event.clientX;
             clickedY = window.event.clientY;

             newXOffset = pageWidth - clickedX - floaterWidth;
             newYOffset = pageHeight - clickedY - floaterHeight;

             placeFloater();
          }

       }
    }
 ...
```

The `setMoveOn()` function sets `justClicked` to `true` when the user clicks the move button. The event immediately bubbles up, causing the execution of the `setNewPos()` function. That function sees the value of `true` and simply sets it back to `false`. Next, the user clicks the new location and the `setNewPos()` function sees the value of `justClicked` as false, so it moves the floater.

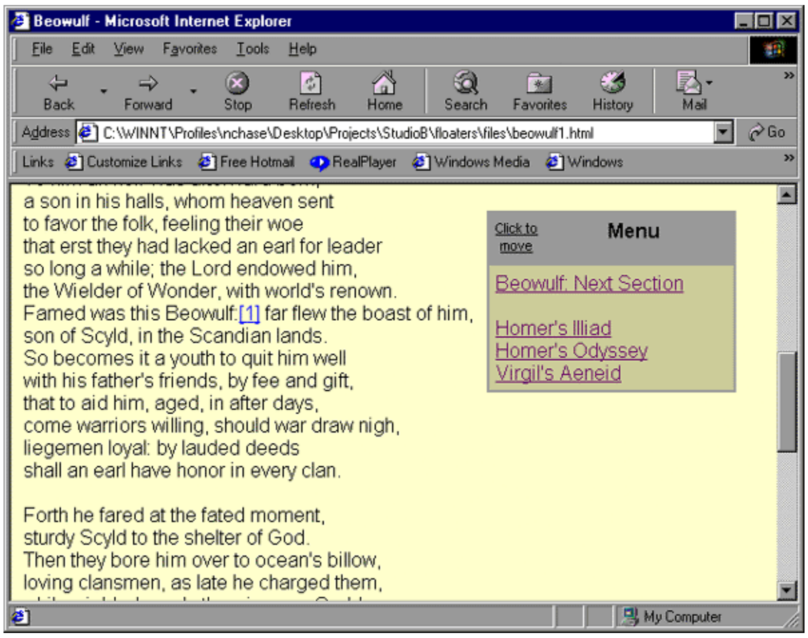
Turn off movement

All that remains is to turn off movement so the user can click the page as usual:

```
...
 function setNewPos() {

    if (isMoving) {
       if (justClicked) {
          justClicked = false;
       } else {
          clickedX = window.event.clientX;
          clickedY = window.event.clientY;

          newXOffset = pageWidth - clickedX - floaterWidth;
          newYOffset = pageHeight - clickedY - floaterHeight;

          placeFloater();

          isMoving = false;
          document.getElementById("movebutton").innerHTML=
                                            'Click to<br />move';

       }
    }
 }
 ...
```

After the script moves the floater, it sets `isMoving` to `false` and replaces the text of the move button.

The user can now reposition the floater and have it stay in position even if the page is scrolled or resized.


Beowulf - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail
Address C:\WINNT\Profiles\nchase\Desktop\Projects\StudioB\floaters\files\beowulf1.html
Go
Links Customize Links Free Hotmail RealPlayer Windows Media Windows
My Computer

a son in his halls, whom heaven sent
to favor the folk, feeling their woe
that erst they had lacked an earl for leader
so long a while; the Lord endowed him,
the Wielder of Wonder, with world's renown.
Famed was this Beowulf:[1] far flew the boast of him,
son of Scyld, in the Scandian lands.
So becomes it a youth to quit him well
with his father's friends, by fee and gift,
that to aid him, aged, in after days,
come warriors willing, should war draw nigh,
liegemen loyal: by lauded deeds
shall an earl have honor in every clan.

Forth he fared at the fated moment,
sturdy Scyld to the shelter of God.
Then they bore him over to ocean's billow,
loving clansmen, as late he charged them,

Click to move

Menu

Beowulf: Next Section

Homer's Illiad
Homer's Odyssey
Virgil's Aeneid

Section 7. JavaScript objects summary

Summary

Any HTML can be turned into a floating object by making it part of an identifiable `div` and positioning it via scripting. All you need to do is determine the size and scrolled offset of the window and dynamically place the floater at a set interval. The scripting necessary for this technique differs from browser to browser, but you can use browser detection techniques to give each browser its appropriate code.

Code that permits users of Internet Explorer 5.x to move the floater by checking their mouse clicks and creating a new offset to be used by the positioning routine can be added to the script.

This tutorial has offered techniques and code samples that demonstrate:

* Creating content to float
* Determining page size
* Positioning content
* Basic browser detection
* Attaching functions to page events
* Creating functions that are executed at regular time intervals

---

Resources

For a thorough grounding in dynamic positioning, take the IBM developerWorks *Understanding Dynamic Positioning* tutorial.

Creating floating objects relies on the use of JavaScript and dynamic positioning. For more information, see the following resources:

* Read *Stupid Browser Tricks #8: Add a Watermark* by Builder.com's Paul Anderson for another look at creating watermarks.
* Read *CSS Layout Techniques for Fun and Profit* for an excellent look at using dynamic positioning as an alternative to HTML tables.
* Read the complete *Cascading Style Sheets level 2 recommendation* .
* Follow the progress of work on *Cascading Style Sheets level 3* at the W3C.
* Explore additional CSS resources at *The CSS Pointers Group* .
* Read *Writing Cross-Browser Dynamic HTML* , by David Boles and Rachael Ann Siciliano, for a look at building your DHTML for all browsers.
* Explore additional DHTML resources at the *Web Developer's Virtual Library's Dynamic HTML pages* .
* Read *Creating Dynamic HTML in Internet Explorer 4+ using JavaScript* , an excerpt from Paul Wilton's *Beginning Javascript*, for a look at using JavaScript to change HTML elements.
* Explore *Danny Goodman's JavaScript Pages* for a look at what functions and tags are supported in which browsers.
* Read a *JavaScript Tutorial for Programmers* by Aaron Weiss.
* Read *A Primer on JavaScript Arrays* by Danny Goodman.
* Read *Creating Robust Functions* .

* Read *All About JavaScript* by Robert W. Husted for a look at how JavaScript on the client compares to JavaScript on the server.
* For a variety of JavaScript documentation, including reference manuals for Javascript 1.5, read *Netscape's JavaScript Documentation* .
* Explore a wealth of information at *Cetus Links: Object-Oriented Language: JavaScript / ECMAScript* .

Downloads

* Download a *zip archive of the sample code* presented in this tutorial.
* Download *IBM Web Browser for OS/2* .
* Download *Microsoft Internet Explorer 5.5* , *Internet Explorer 6* , or *Internet Explorer 5.0 for Macintosh* .
* Download *Netscape 6* , with improved compliance over earlier versions.
* Download *Opera* , a standards-compliant browser available for OS/2, BeOS, Linux/Solaris, Mac, QNX, Symbian OS, and more than 20 different localized versions for Windows.

Feedback

Was this tutorial helpful? Let us know what you think. We look forward to hearing from you!

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The Toot-O-Matic tool is a short Java program that uses XSLT stylesheets to convert the XML source into a number of HTML pages, a zip file, JPEG heading graphics, and PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML.