www.developer.com

- Home
- Languages
  - ASP
  - CSS
  - DHTML
  - HTML
  - JavaScript
  - Perl
  - PHP
  - RSS
  - XHTML
  - XML
- General
  - 3D Modeling
  - Design
  - Design Lab
  - Graphics
  - HTML Tools
  - Internet
  - JavaScript Tools
  - Multimedia
  - Programming
  - Promotion
  - Reviews
  - Search Engine Resources
  - Video
  - Web Servers
- Reference
  - Color Codes
  - HTML Characters
  - HTML Tags
  - JavaScript Core
  - Reference Section
  - Site Services
  - XML
- Newsletter

Search Site

Go

Login | Register

Enter your Email Address

Close

Home / Programming / javascript / rg30 / Comet Programming: the Hidden IFrame Technique

Comet Programming: the Hidden IFrame Technique

By Rob Gravelle

Tweet

Share

أعجبني ١١

#> [next]

As covered in Comet Programming: Using Ajax to Simulate Server Push, Comet is a Web application model that enables Web servers to send data to the client without having to explicitly request it. Hence, the Comet model provides an alternate mechanism to classical polling to update page content and/or data. In that article, we learned how to use XMLHttpRequest long polling to refresh page components and keep cached data in synch with the server. Another Comet strategy, sometimes referred to as the "Forever Frame" technique, is to use a hidden IFrame. As with XMLHttpRequest long polling, the "Forever Frame" technique also relies on browser-native technologies, rather than on proprietary plugins or other special technologies.

IFrames Technique Overview

IFrame stands for Inline Frame. It allows a Web page to embed one HTML document inside another HTML element. As you can imagine, it's a simple way to create a "mashup", whereby the page combines data from several sources into a single integrated document:

view plain | print | ?

```
<html>
```

view plain | print | ?

```
<head>
<title>IFrames Example</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF" id=body>
<h2 align="center">IFrames Example</h2>
    The content below comes from the website http://www.robgravelle.com
    <iframe src="http://www.robgravelle.com/" height="200">
      Your browsers does not support IFrames.
    </iframe>
</body>
</html>
```

Normally, data delivered in HTTP responses is sent in one piece. The length of the data is indicated by the `Content-Length` header field. With chunked encoding, the data is broken up into a series of blocks of data and transmitted in one or more "chunks" so that a server may start sending data before it knows the final size of the content that it's sending. Note that the size of the blocks may or may not be the same:

view plain | print | ?

```
HTTP/1.1 200 OK
Content-Type: text/plain
Transfer-Encoding: chunked
23
This is the data in the first chunk
1A
and this is the second one
0
```

In Forever Frame Streaming, a series of JavaScript commands is sent to a hidden IFrame as a chunked block. As events occur, the IFrame is gradually filled with script tags, containing JavaScript to be executed in the browser. Because browsers render HTML pages incrementally, each script tag is executed as it is received.

Two benefits of the IFrame method are that it's fairly easy to implement and it works in every common browser. On the cons side, there is no way to handle errors reliably nor is it possible to track the state of the request calling process. Therefore, if you want to track the progress of the request, you'd best stick with the XMLHttpRequest method.

The CD Sales Page Revisited

You'll remember this example from the Comet Programming: Using Ajax to Simulate Server Push article. We'll modify it in order to highlight the differences between the Ajax and IFrame techniques.

Minimal JavaScript Code

Many developers make the mistake of using Ajax to call the PHP script from the Web page. This is unnecessary and blurs the line between the two techniques. My view is that if you're going to go through the trouble of employing Ajax, you might as well forego the IFrame. One of the main benefits of the Hidden IFrame technique in my view is its simplicity and light use of client-side scripting. In fact, we only need one small JavaScript function:

view plain | print | ?

```
<script type="text/javascript">
   function updateCount(c) {
      document.getElementById('CD Count').innerHTML = c;
   }
</script>
```

The `updateCount()` function refreshes the CD count within the "CD Count" <SPAN> element. The innerHTML property is used because some browsers do not support innerText.

Integrated PHP Code

Another departure from the Ajax Comet page is that all the required functionality is contained within the same file. PHP, like ASP and JSP is ideal for inserting within the HTML markup. By integrating code in one file, the results of actions that take place on the server are then available to the rest of the page or can be included in place using the echo() function. Whereas the PHP and HTML were housed in separate files last time in order to clearly distinguish server-side components from client-side, the CometHiddenIFrame.php file contains all the necessary code. This was not done merely for the sake of convenience; but contributes to setting the execution order of the code. For example, at the top of the page, there is a short PHP script to send the headers to the browser. It is necessary to do this before sending any HTML to the browser, as by that time, it is too late. In fact, trying to send headers after the page content has already begun to download will result in a "Headers Already Sent" PHP error.

The `set_time_limit()` function is called with a value of `0` to turn off the maximum script execution time (which is 30 seconds by default). The headers that we are sending prevent browser caching - always important when sending dynamic content. Finally the `flush()` function is called to make sure that the header information is sent to the client:

view plain | print | ?

```
<?php
set_time_limit(0);
header("Cache-Control: no-cache, must-revalidate");
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
flush();
?>
```

The rest of the PHP code is contained in a script at the end of the page. It must be placed there so that the page has a chance to load, as the `do while` loop will prevent the rest of the page from displaying. The random order generation has also been moved from the `getCdCount()` function so that we can call `updateStock()` without a delay when the `Purchase` button is clicked. In fact, the `getCdCount()` function is now deprecated, and can be removed:

view plain | print | ?

```
$num = $_GET['num'];
if ( $num == "")
{
   //start the update service
   srand();
   do
   {
      $newOrder  = rand(1, 3);
      $sleeptime = rand(2, 10);
      sleep($sleeptime);
   } while(updateStock($newOrder) > 0);
}
else
{
   updateStock((int)$num);
}
```

#>

[next]

Degree: - Select All Degrees - ▼

○ Online ○ Campus ● Both

Submit

Top White Papers and Webcasts

It's Time to Rethink CRM

Today's CRM systems have the ability to deliver more than just lead generation. If leveraged correctly, they can be major drivers in loyalty and relationship management. But many companies have yet to unlock their true potential...

IT FOCUS ON ENTERPRISE APPLICATIONS
READY YOUR ENTERPRISE FOR THE API REVOLUTION

Ready Your Enterprise for the API Revolution

APIs are changing more than just software architectures. From planning through implementation and beyond, an API-driven business model brings a host of...

IT FOCUS ON ENTERPRISE APPS
PLANNING A SUCCESSFUL ERP IMPLEMENTATION

Planning a Successful ERP Implementation

Changes in the ERP landscape have made the deployment process increasingly complex. Post-deployment, implementations often fail to deliver on their...

WebReference

- Tip Archive
- About
- Sitemap
- Contact
- Subscribe

Topics

- Web Authoring
- The Internet and the Web
- Multimedia
- Web Programming
- Website Promotion and Marketing
- Contributors

developer.com

- Terms Of Service
- Licensing and Permissions
- Privacy Policy
- Advertising
- Newsletters