# CIS 4004: Web Based Information Technology Fall 2013

## JavaScript – Part 2

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cis4004/fall2013

Department of Electrical Engineering and Computer Science
University of Central Florida

# JavaScript – Part 2

- JavaScript is a web programming language that you can use with HTML5.
- Although many new HTML5 elements support features that no longer require the developer to employ JavaScript, it can used to access certain parts of your web pages written in HTML5 and do other things that simply cannot be done without JavaScript.
- HTML5 defines your web pages content. CSS defines the presentation of your web pages. JavaScript defines special behavior for the page.

## JavaScript – Part 2

- JavaScript has nothing to do with Java. They are two completely different and unrelated languages.
- JavaScript is considered a scripting language because it is interpreted by the browser at runtime (when the web page is actually opened) rather than compiled and stored on your computer.
- Slightly different versions of JavaScript are present and lead to different implementations of the language on different browsers. Because JavaScript meets an ECMAScript standard (ECMA-262), these differences are slight, and we'll discuss only aspects of JavaScript that you can use with HTML5.

# JavaScript – Part 2

- JavaScript has quite a range of possibilities, and its use has exploded in recent years.
- This explosion, in part, is due to JavaScript libraries like jQuery, MooTools, and YUI, that have made it easier to add both simple interactivity and sophisticated behavior to pages, while helping them to behave more consistently across various browsers.
- Of these, jQuery enjoys the most widespread use, largely because beginners find it easier to learn, it has good online documentation, and it has a very large community behind it.
- We’ll look at jQuery in more detail later.

## JavaScript – Part 2

- Browser vendors have spent considerable resources making their browsers process JavaScript significantly faster than their versions of even just a few years ago.

- JavaScript also works in tablet and modern mobile browsers, though for performance reasons, you'll want to be smart about how much you load in pages for these devices (much more later).

- There are two primary kinds of scripts – those that you load from an external file (in text-only format) and those that are embedded in your web page. It's the same concept as external and embedded style sheets.

## JavaScript – Part 2

- As with CSS, its generally better to load scripts from an external file than to embed them in your markup.
- You'll reap some of the same benefits, in that a single JavaScript file can be loaded by any number of pages that need it. You can edit one script rather than updating similar scripts in a number of individual pages of markup.
- Whether loading an external or an embedded script, the `script` element is used. The `src` attribute of the script element references the script's URL.
- First, let's look at how a browser handles scripts.

## How A Browser Handles Scripts

- As a page loads, by default the browser downloads (for external scripts), parses, and executes each script in the order in which it appears in the markup.
- As its processing, the browser neither downloads nor renders any content that appears *after* the `script` element – not even text. This is known as blocking behavior.
- This is true for both embedded and external scripts. As you can imagine, it can really impact the rendering speed of your page, depending on the size of the script and what actions it performs.

## How A Browser Handles Scripts

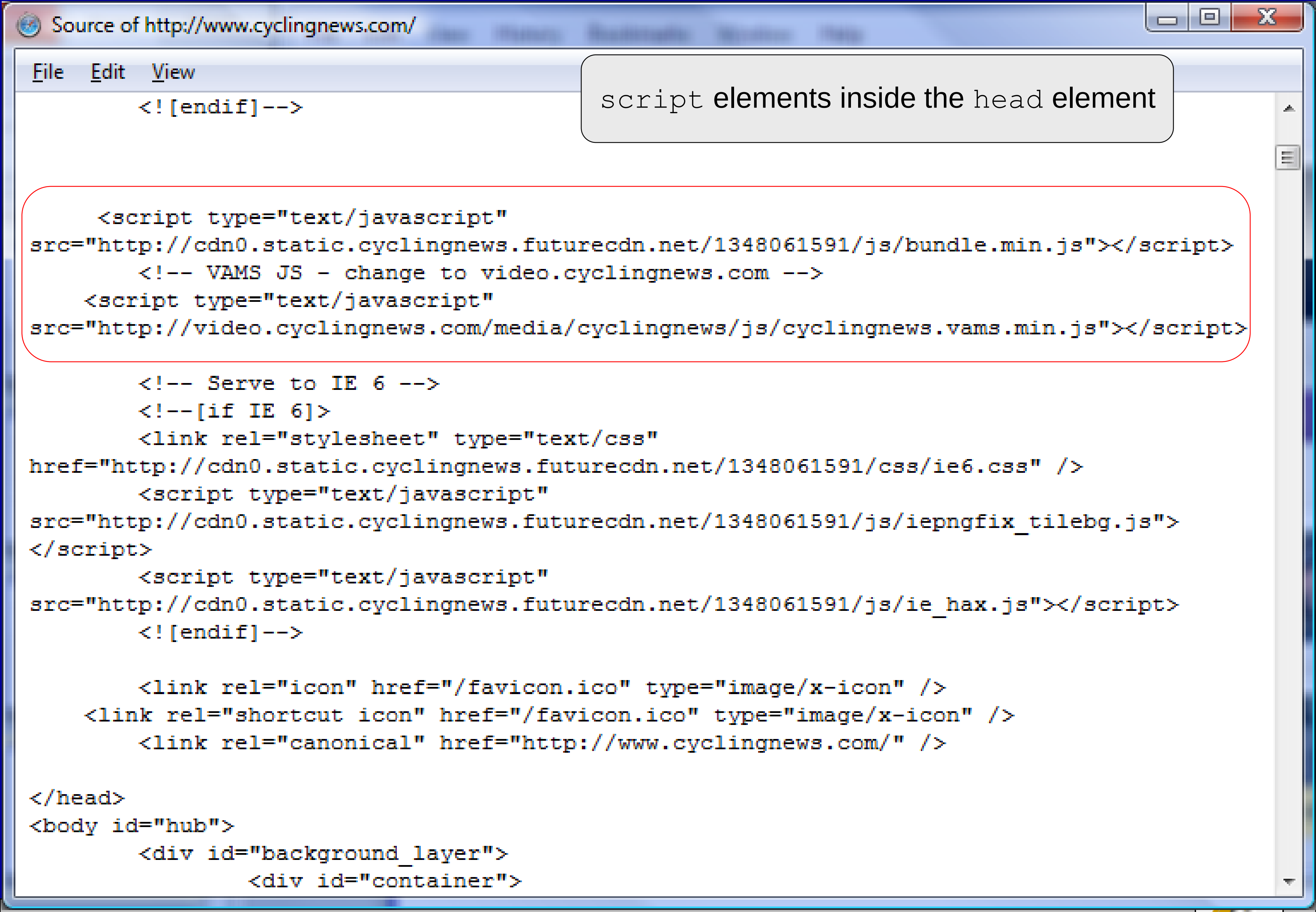
- Most browsers do this because your JavaScript may include code on which another script relies, code that generates content immediately, or code that otherwise alters your page.

- Browsers need to take all of that into account *before* they finish rendering your page.

- So how do you avoid this? The easiest technique to make your JavaScript non-blocking is to put all `script` elements at the end of your markup, right before the `</body>` end tag.

## How A Browser Handles Scripts

- If you’ve ever spent any time viewing source code on various web sites, you’ve no doubt seen scripts loaded in the head element (see next page for an example).
- Outside of the occasional instance where it might be necessary, it is considered an outdated practice to do this and you should avoid it whenever possible. (One case where it is necessary is loading the HTML5 `shiv` which is a collection of JavaScript functions that allow outdated browsers to support some of the new HTML5 elements – more later).
- If you do load scripts from the `head`, they should be placed after all `link` elements that load CSS files.



`script` elements inside the `head` element

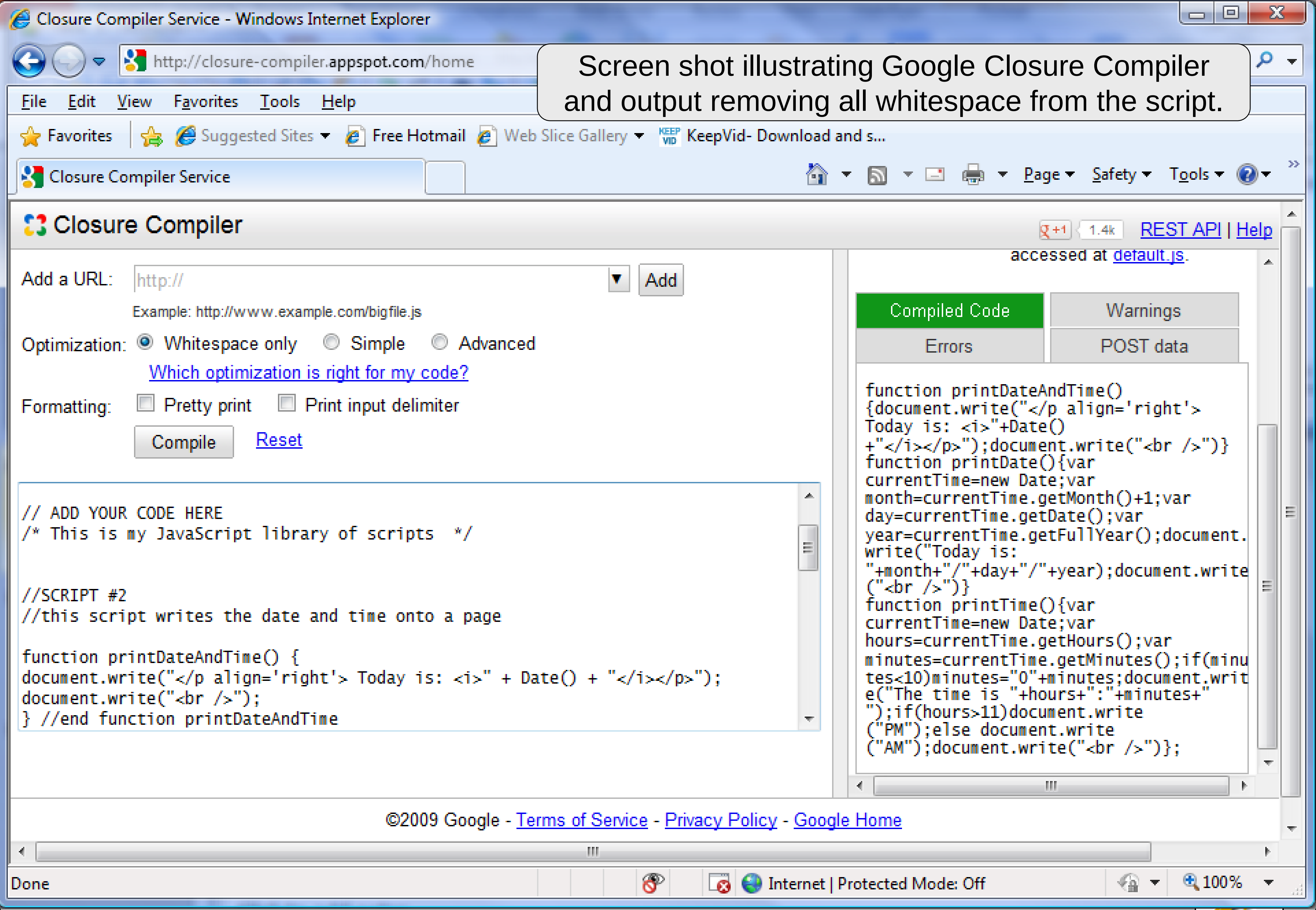
```
        <![endif]-->


     <script type="text/javascript"
src="http://cdn0.static.cyclingnews.futurecdn.net/1348061591/js/bundle.min.js"></script>
        <!-- VAMS JS - change to video.cyclingnews.com -->
    <script type="text/javascript"
src="http://video.cyclingnews.com/media/cyclingnews/js/cyclingnews.vams.min.js"></script>

        <!-- Serve to IE 6 -->
        <!--[if IE 6]>
        <link rel="stylesheet" type="text/css"
href="http://cdn0.static.cyclingnews.futurecdn.net/1348061591/css/ie6.css" />
        <script type="text/javascript"
src="http://cdn0.static.cyclingnews.futurecdn.net/1348061591/js/iepngfix_tilebg.js">
</script>
        <script type="text/javascript"
src="http://cdn0.static.cyclingnews.futurecdn.net/1348061591/js/ie_hax.js"></script>
        <![endif]-->

        <link rel="icon" href="/favicon.ico" type="image/x-icon" />
    <link rel="shortcut icon" href="/favicon.ico" type="image/x-icon" />
        <link rel="canonical" href="http://www.cyclingnews.com/" />

</head>
<body id="hub">
        <div id="background_layer">
                <div id="container">
```

## How A Browser Handles Scripts

- Another way to speed up script loading is to combine all your JavaScript into a single file, or as few as possible, and then minify the code.
- Typically, minified code contains no line breaks, comments, or any extra whitespace. This sets it apart from un-minified code. Imagine writing code in one long line without ever pressing return or enter!

If you'd like to try minifying your JavaScripts you can use the following:
Google Closure Compiler:
    download & documentation at: http://code.goggle.com/closure/compiler
    online version at: http://closure-compiler/appspot.com
YUI Compressor:
    download & documentation at: http://developer.yahoo.com/yui/compressor
    (unofficial) online version at: http://refresh-sf-com/yui

Screen shot illustrating Google Closure Compiler and output removing all whitespace from the script.


Closure Compiler Service - Windows Internet Explorer
http://closure-compiler.appspot.com/home
File Edit View Favorites Tools Help
Favorites Suggested Sites Free Hotmail Web Slice Gallery KeepVid- Download and s...
Closure Compiler Service
Page Safety Tools
Closure Compiler
REST API | Help
Compiled Code
Warnings
Errors
POST data
Done
Internet | Protected Mode: Off
100%

Add a URL: http://

Example: http://www.example.com/bigfile.js

Optimization: [✓ Whitespace only] Simple Advanced

Which optimization is right for my code?

Formatting: Pretty print Print input delimiter

Compile Reset

```
// ADD YOUR CODE HERE
/* This is my JavaScript library of scripts  */


//SCRIPT #2
//this script writes the date and time onto a page

function printDateAndTime() {
document.write("</p align='right'> Today is: <i>" + Date() + "</i></p>");
document.write("<br />");
} //end function printDateAndTime
```

accessed at default.js.

```
function printDateAndTime()
{document.write("</p align='right'>
Today is: <i>"+Date()
+"</i></p>");document.write("<br />")}
function printDate(){var
currentTime=new Date;var
month=currentTime.getMonth()+1;var
day=currentTime.getDate();var
year=currentTime.getFullYear();document.
write("Today is:
"+month+"/"+day+"/"+year);document.write
("<br />")}
function printTime(){var
currentTime=new Date;var
hours=currentTime.getHours();var
minutes=currentTime.getMinutes();if(minu
tes<10)minutes="0"+minutes;document.writ
e("The time is "+hours+":"+minutes+"
");if(hours>11)document.write
("PM");else document.write
("AM");document.write("<br />")};
```

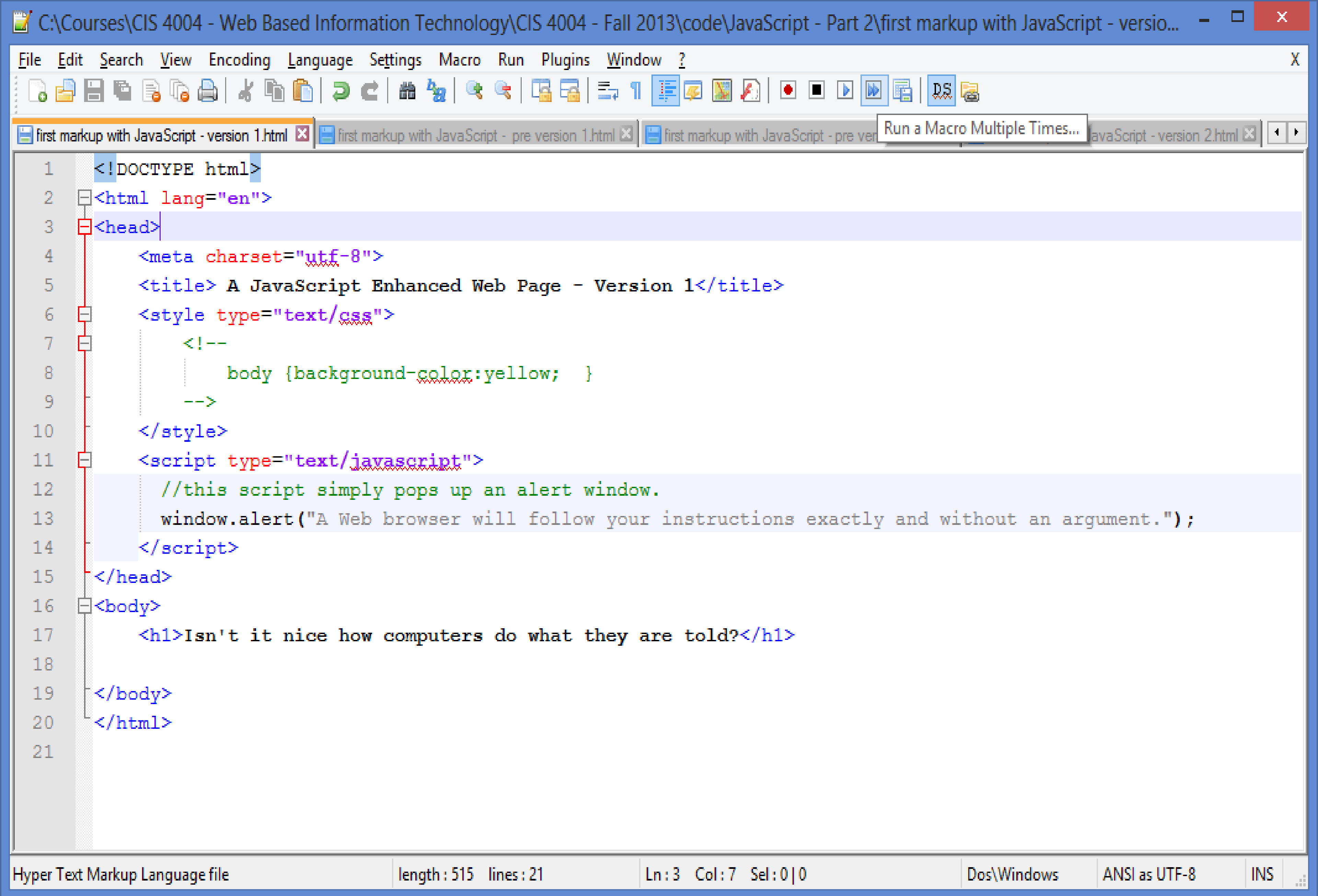
©2009 Google - Terms of Service - Privacy Policy - Google Home

## How A Browser Handles Scripts

- Either of these tools will reduce the file size of the script, but results will vary from script to script.
- Generally, it is faster for a browser to load one file than two (or more), even if the single file is larger than the combined size of the individual files (unless the one file is *much* larger).
- Browsers that do not understand JavaScript (these are rare nowadays) or ones that have it disabled by the user will ignore your JavaScript file. So be sure that your page doesn’t rely on JavaScript to provide users access to its content and basic experience.

# Adding An Embedded Script

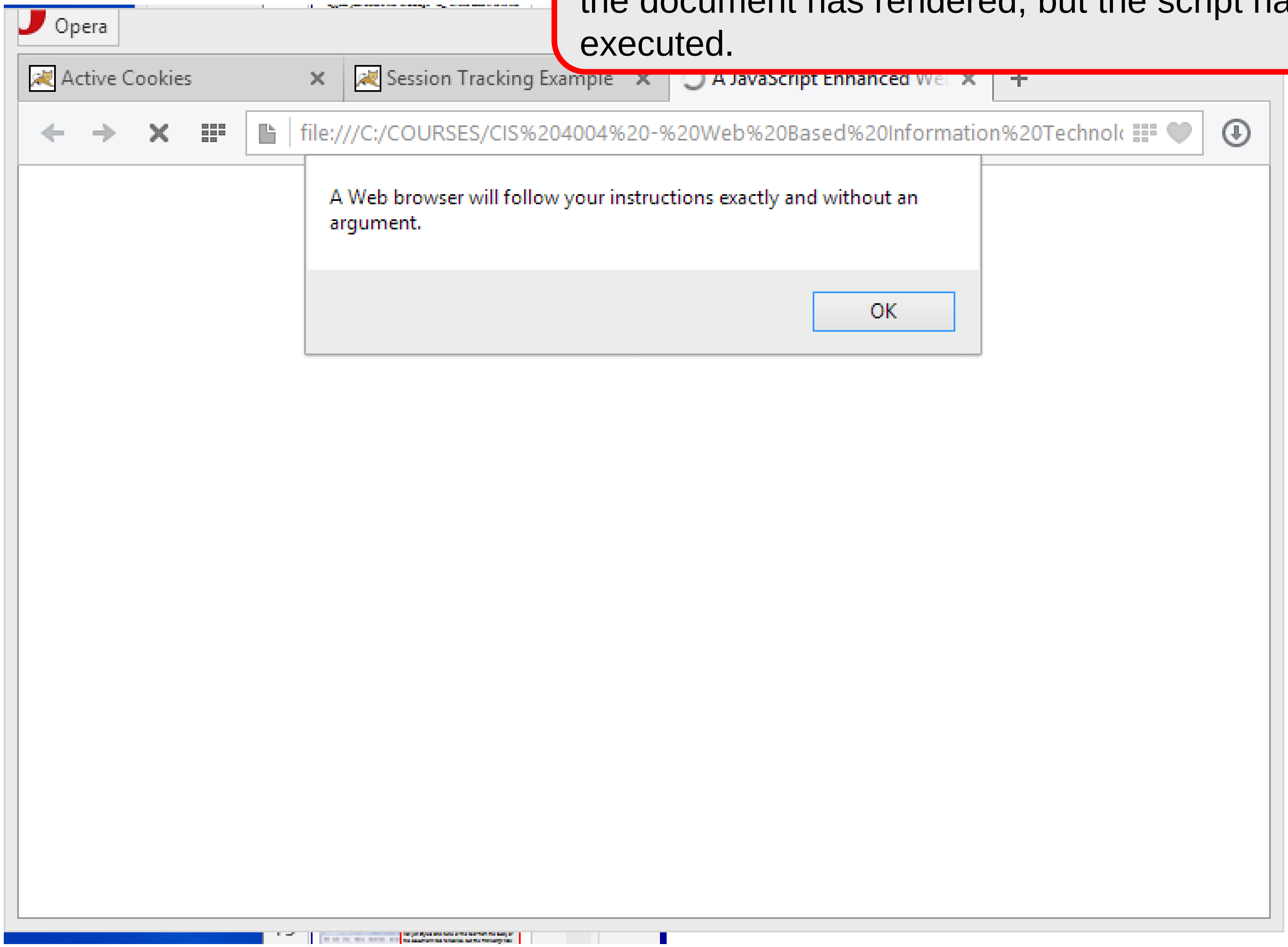
- Although it is not the preferred way of including a script in your markup, the first example, on the next page, illustrates embedding a very simple JavaScript script into the body of a page.
- Note, that although preferred practice would have the script appear just before the `</body>` tag, in this case it is in the `<head>` of the document.
- This simple script just pops up an alert message for the user to read.
- Notice the sequence of events as shown on page 16 when the page renders.


C:\Courses\CIS 4004 - Web Based Information Technology\CIS 4004 - Fall 2013\code\JavaScript - Part 2\first markup with JavaScript - versio...
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
first markup with JavaScript - version 1.html
first markup with JavaScript - pre version 1.html
first markup with JavaScript - pre ver
Run a Macro Multiple Times...
avaScript - version 2.html

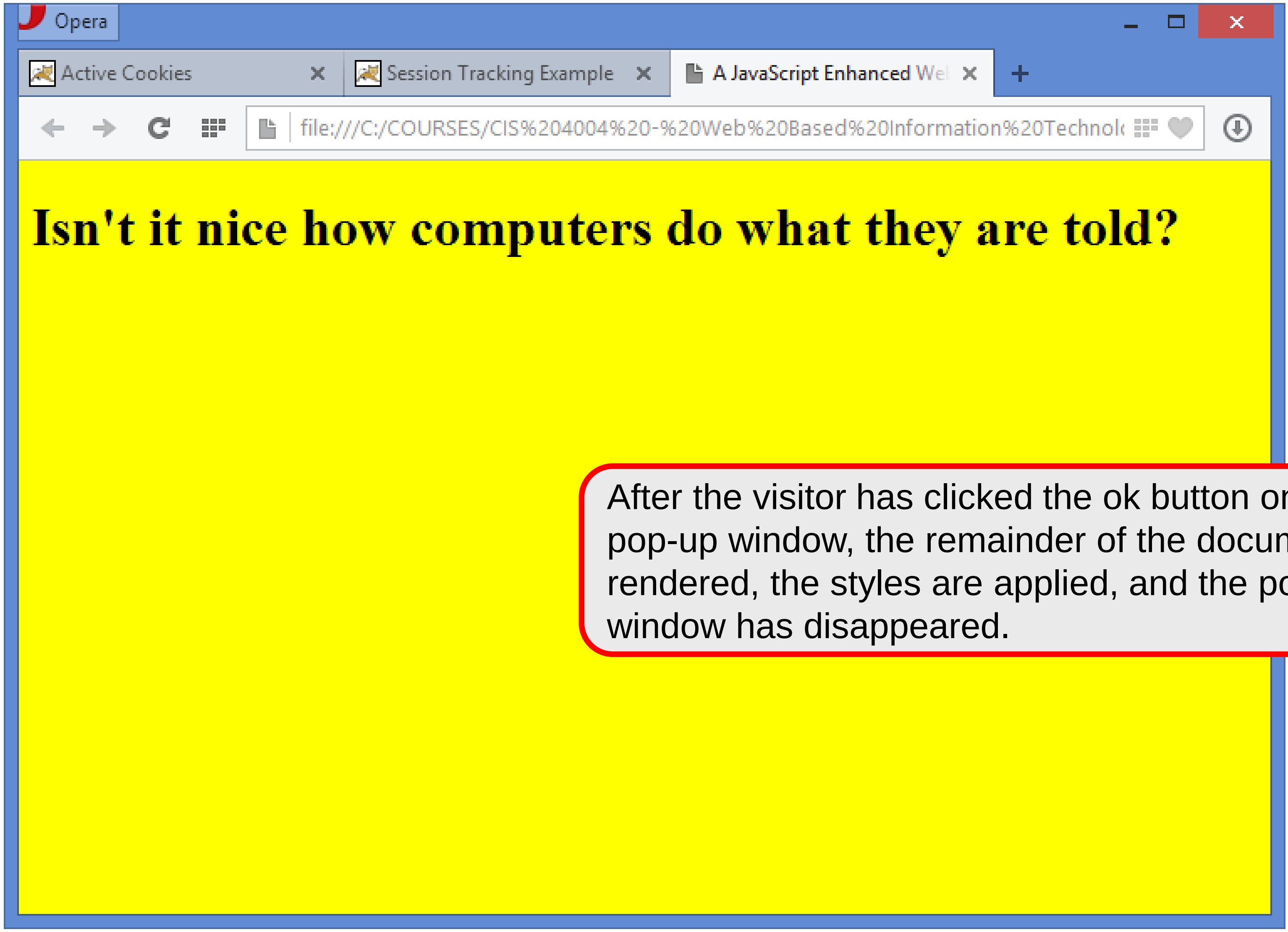
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title> A JavaScript Enhanced Web Page - Version 1</title>
    <style type="text/css">
        <!--
            body {background-color:yellow;   }
        -->
    </style>
    <script type="text/javascript">
      //this script simply pops up an alert window.
      window.alert("A Web browser will follow your instructions exactly and without an argument.");
    </script>
</head>
<body>
    <h1>Isn't it nice how computers do what they are told?</h1>

</body>
</html>
```

Hyper Text Markup Language file
length : 515    lines : 21
Ln : 3    Col : 7    Sel : 0 | 0
Dos\Windows
ANSI as UTF-8
INS

Initial screen shot.  Notice that the background is not yet styled and none of the text from the body of the document has rendered, but the script has executed.


Opera
Active Cookies
Session Tracking Example
A JavaScript Enhanced We
file:///C:/COURSES/CIS%204004%20-%20Web%20Based%20Information%20Technol
OK

A Web browser will follow your instructions exactly and without an argument.

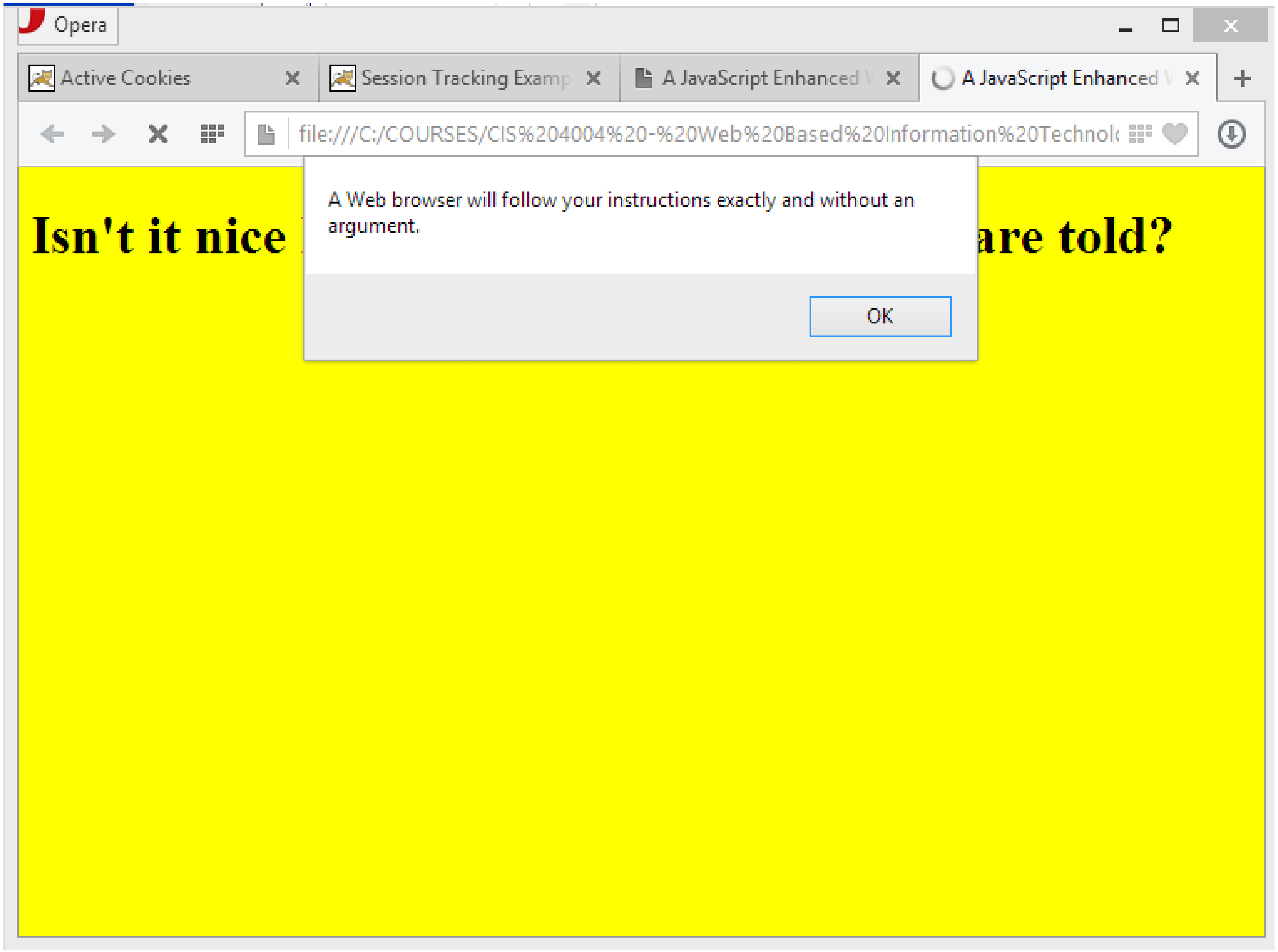

Opera
Active Cookies
Session Tracking Example
A JavaScript Enhanced We
file:///C:/COURSES/CIS%204004%20-%20Web%20Based%20Information%20Technol

Isn't it nice how computers do what they are told?

After the visitor has clicked the ok button on the pop-up window, the remainder of the document is rendered, the styles are applied, and the pop-up window has disappeared.

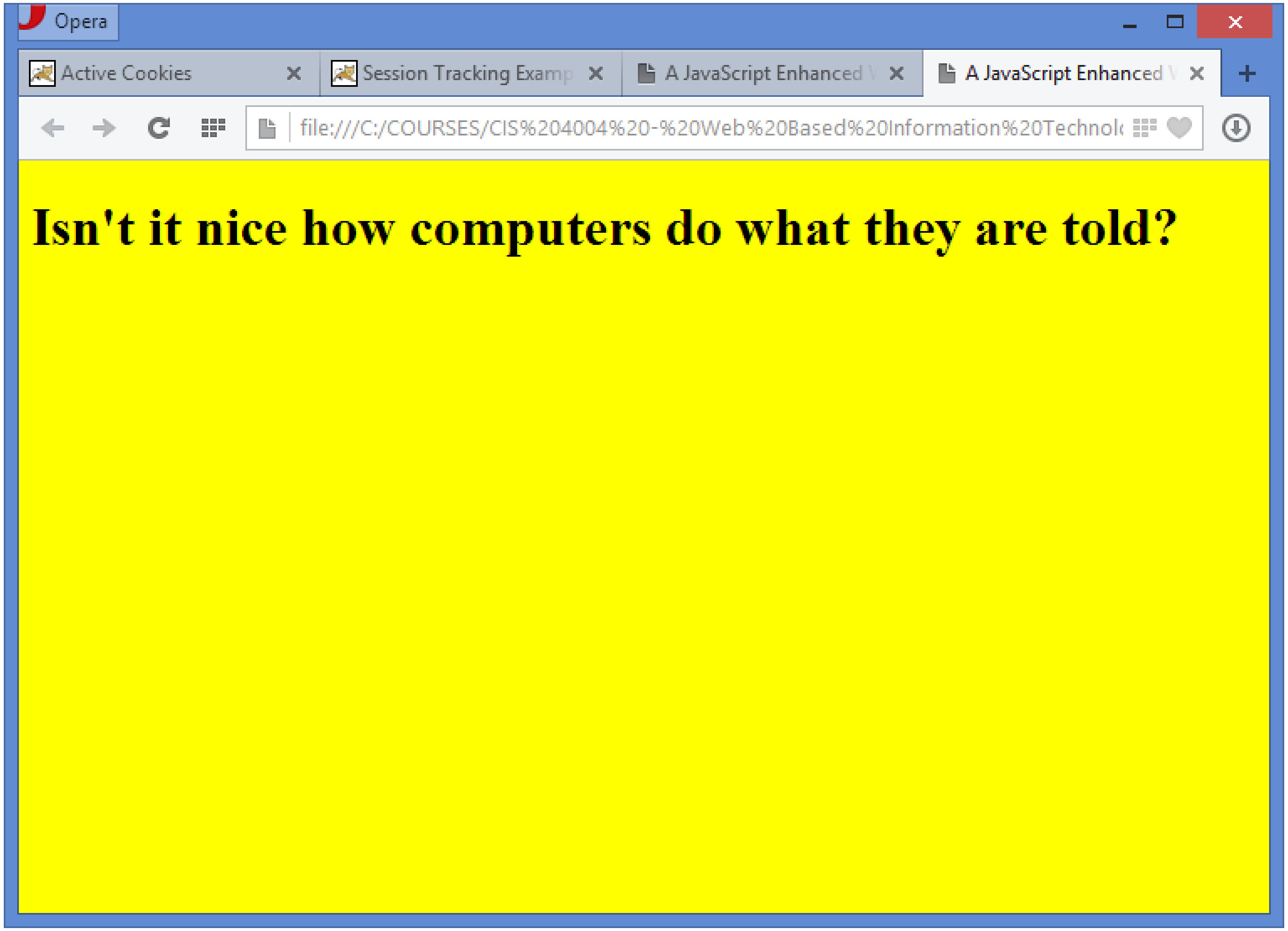
## Adding An Embedded Script

- Now look at a second version of the original markup.
- This markup places the script just before the `</body>` tag.
- Notice again in the following sequence of screen shots how the page is rendered.



Isn't it nice … are told?

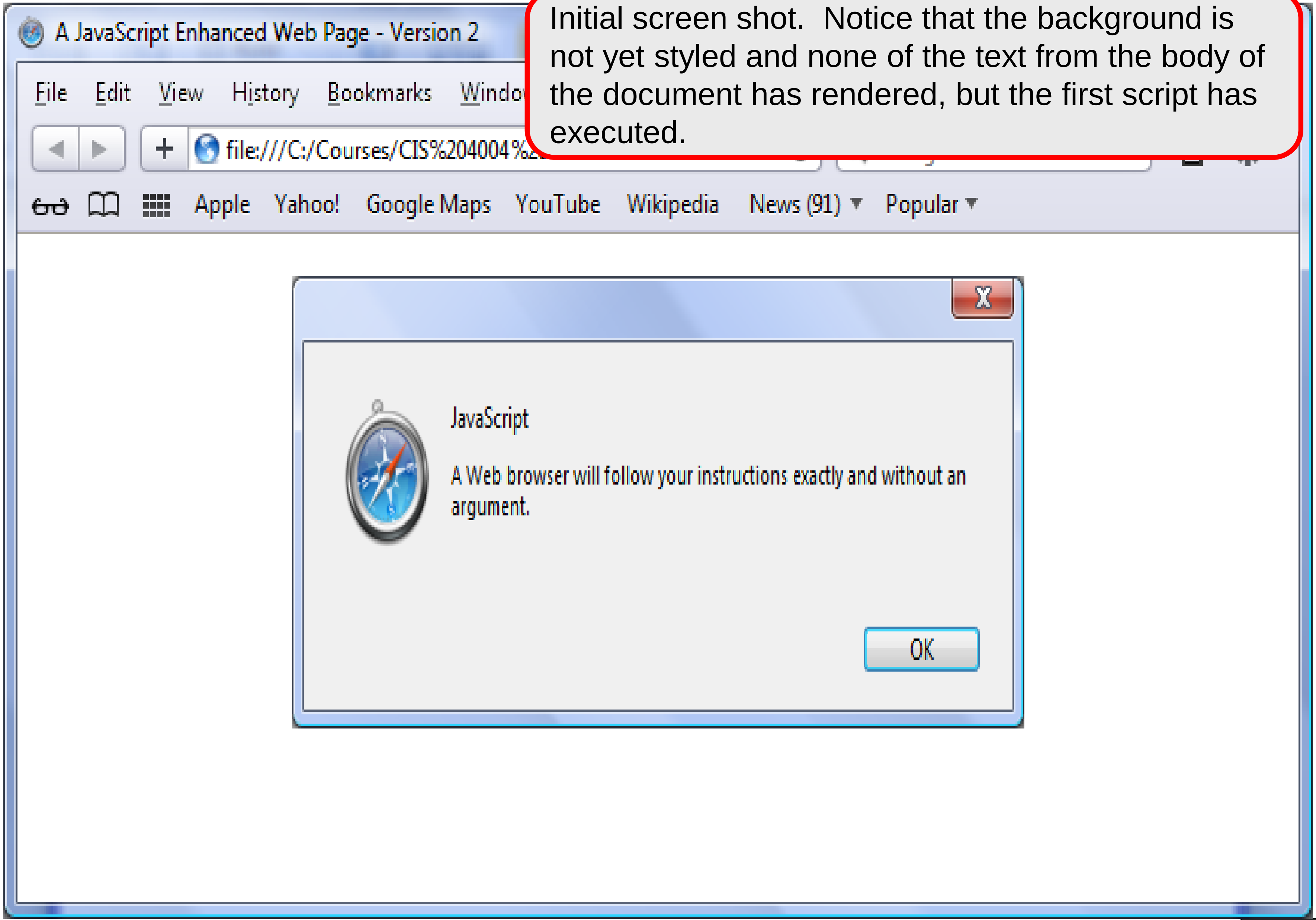
A Web browser will follow your instructions exactly and without an argument.



Isn't it nice how computers do what they are told?

## Adding An Embedded Script

- Now look at a second version of the original markup.
- This markup contains two different scripts.
- Notice again in the following sequence of screen shots that both of the scripts are executed before the body of the document is rendered or the styles applied to the document.

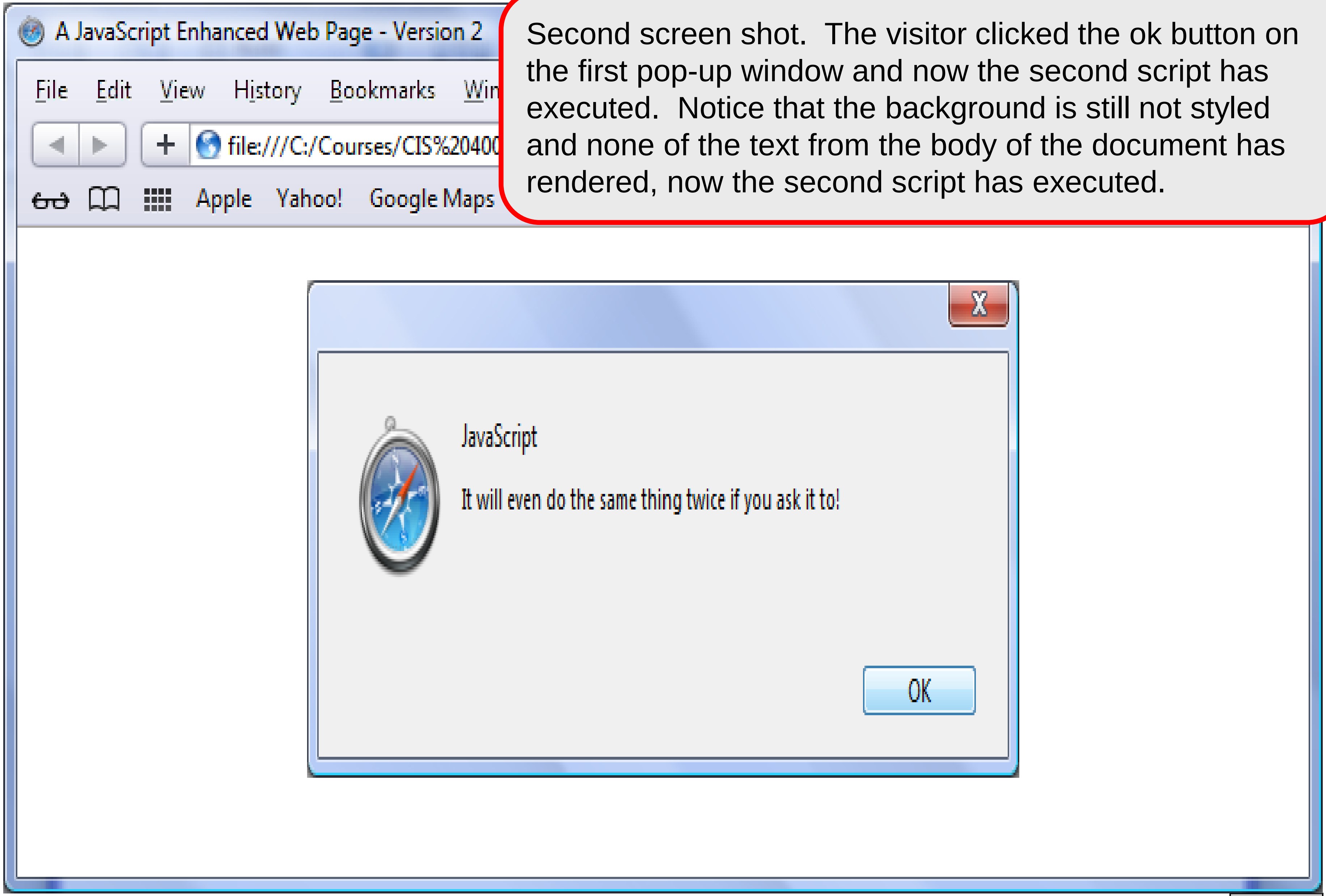


A JavaScript Enhanced Web Page - Version 2
File Edit View History Bookmarks Windo
file:///C:/Courses/CIS%204004%2
Apple Yahoo! Google Maps YouTube Wikipedia News (91) Popular
OK

JavaScript

A Web browser will follow your instructions exactly and without an argument.

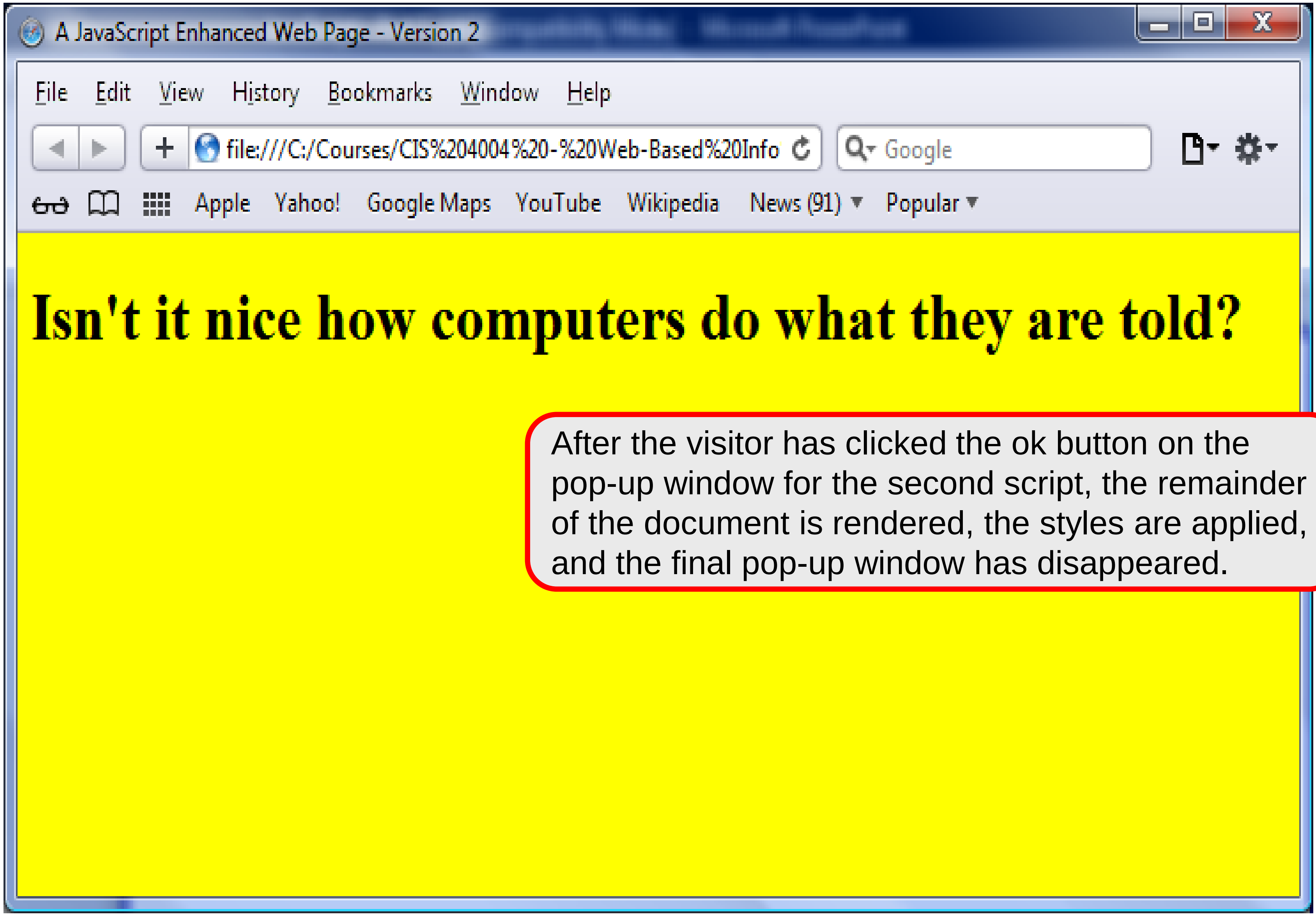
Initial screen shot. Notice that the background is not yet styled and none of the text from the body of the document has rendered, but the first script has executed.


A JavaScript Enhanced Web Page - Version 2
File Edit View History Bookmarks Win
file:///C:/Courses/CIS%20400
Apple Yahoo! Google Maps
OK

JavaScript

It will even do the same thing twice if you ask it to!

Second screen shot. The visitor clicked the ok button on the first pop-up window and now the second script has executed. Notice that the background is still not styled and none of the text from the body of the document has rendered, now the second script has executed.

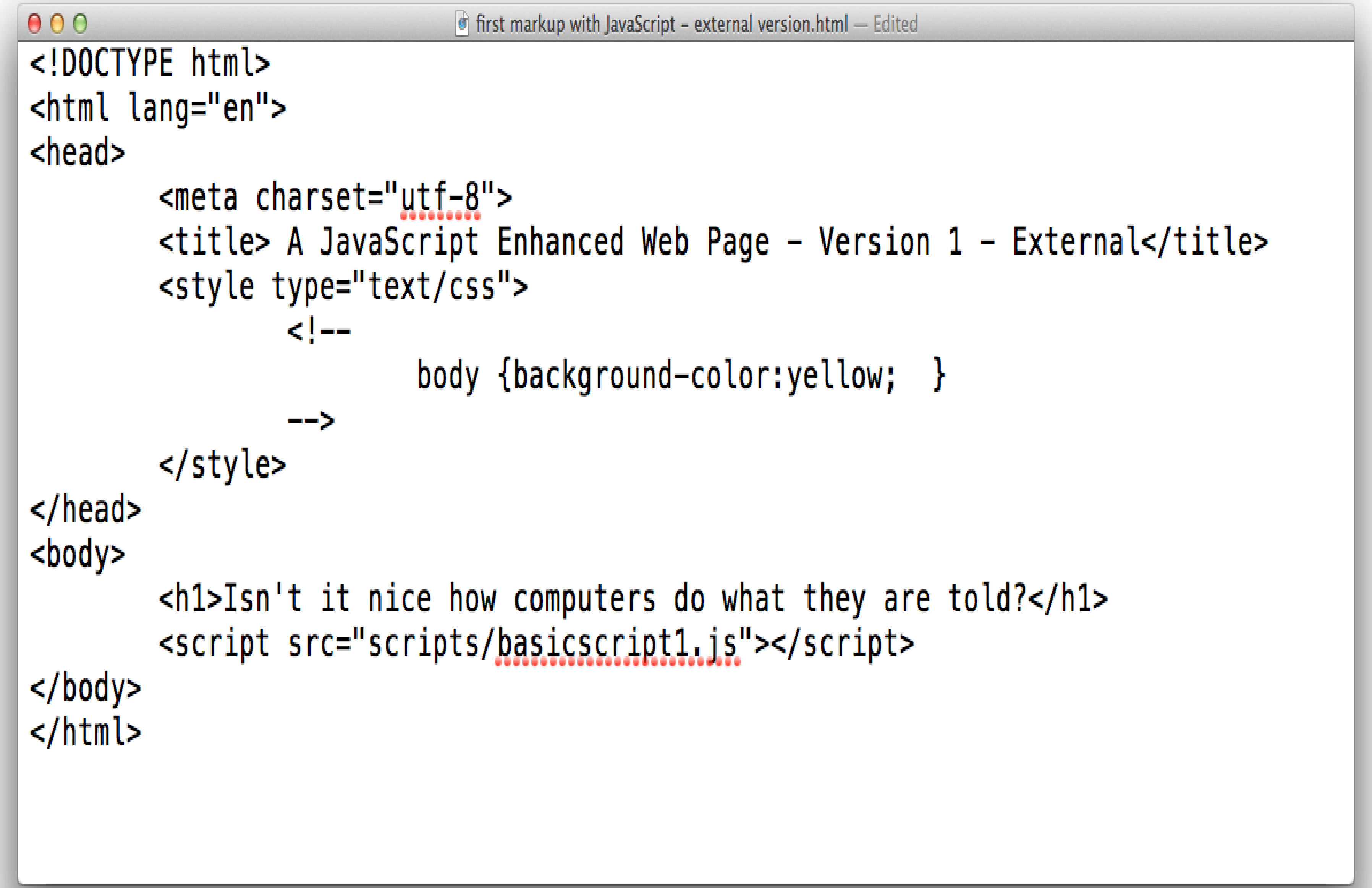

A JavaScript Enhanced Web Page - Version 2
File Edit View History Bookmarks Window Help
file:///C:/Courses/CIS%204004%20-%20Web-Based%20Info
Google
Apple Yahoo! Google Maps YouTube Wikipedia News (91) Popular

Isn't it nice how computers do what they are told?

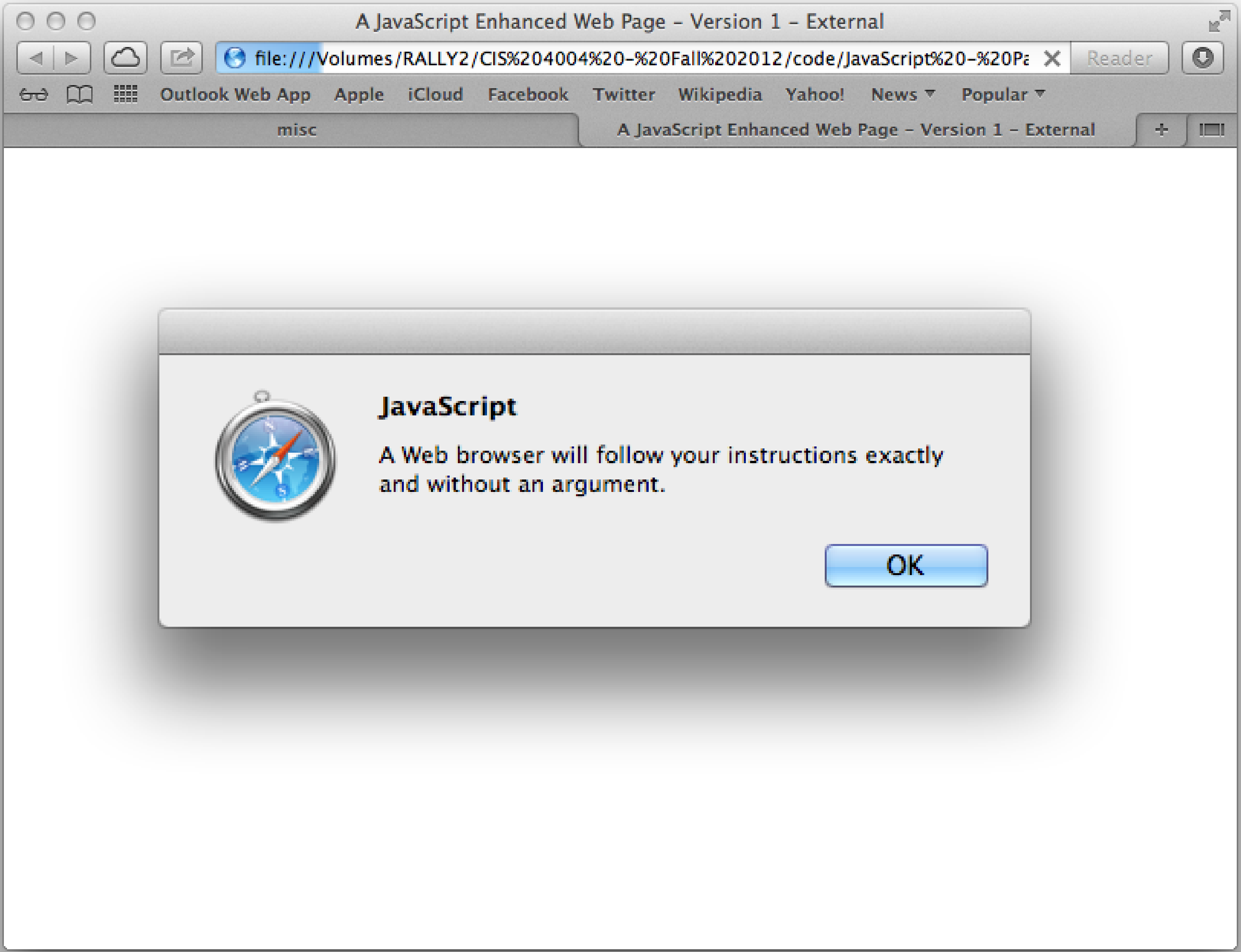
After the visitor has clicked the ok button on the pop-up window for the second script, the remainder of the document is rendered, the styles are applied, and the final pop-up window has disappeared.

## Adding An Embedded Script

- The next example uses the exact same script as the first example. The difference is that the script is external to the markup.
- In this case, I've created a folder named `scripts` and all of the external scripts will reside in this folder. The script file that contains the same script as the first example is named `basicscript1.js`.
- JavaScript files should use a `.js` file extension.
- Notice that the behavior of the page is exactly the same as with the first example.

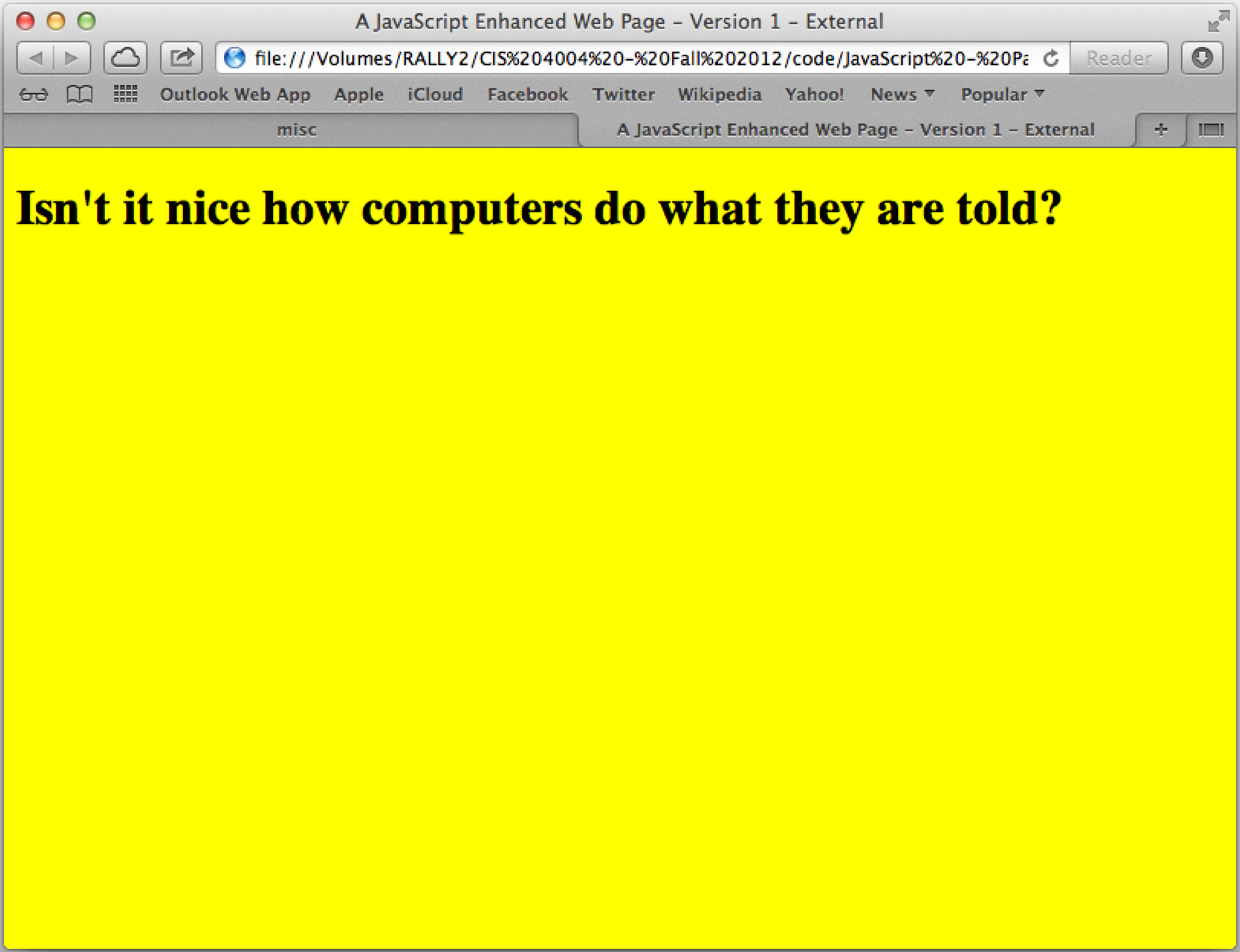

first markup with JavaScript – external version.html — Edited

```html
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="utf-8">
        <title> A JavaScript Enhanced Web Page - Version 1 - External</title>
        <style type="text/css">
                <!--
                        body {background-color:yellow;  }
                -->
        </style>
</head>
<body>
        <h1>Isn't it nice how computers do what they are told?</h1>
        <script src="scripts/basicscript1.js"></script>
</body>
</html>
```


A JavaScript Enhanced Web Page – Version 1 – External
file:///Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Pa
Reader
Outlook Web App
Apple
iCloud
Facebook
Twitter
Wikipedia
Yahoo!
News
Popular
misc
A JavaScript Enhanced Web Page – Version 1 – External

JavaScript

A Web browser will follow your instructions exactly and without an argument.


A JavaScript Enhanced Web Page - Version 1 - External
file:///Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Pa
Reader
Outlook Web App
Apple
iCloud
Facebook
Twitter
Wikipedia
Yahoo!
News
Popular
misc
A JavaScript Enhanced Web Page - Version 1 - External

Isn't it nice how computers do what they are told?

## How To Really Use JavaScript

- The previous two examples illustrate very basic use of JavaScript.
- Notice too, that in both of those examples, the execution of the script was automatic…the visitor didn’t do anything to cause the script to execute (other than visiting the page).
- The real power of JavaScript in HTML5 can be better seen when the script waits until the visitor does something to launch the script. For example, if the visitor clicks something, you can launch any script you want.
- Do to this you use an HTML5 event handler.

## How To Really Use JavaScript

- An HTML5 event handler allows the page to detect that some kind of action (an event) as occurred and has a built-in function that recognizes the event.
- HTML5 recognizes a lot of different events. Some of the events occur automatically – such as when the page loads. Other events occur when visitors do something with the mouse or keyboard.
- The table on the next page illustrates some of the more common event handlers in HTML5.

## How To Really Use JavaScript

| | | | | |
|---|---|---|---|---|
| onchange | onclick | ondbleclick | ondrag | ondragend |
| ondrageneter | ondragleave | ondrageover | ondragstart | ondrop |
| onkeydown | onkeypress | onkeyup | onmousedown | onmousemove |
| onmouseout | onmouseover | onmouseup | onmousewheel | onpause |
| onplay | onplaying | onprogress | onloadstart | onload |

A sample of HTML5 event handlers

## How To Really Use JavaScript

- The general format of all events linked to elements is:

```
<element onEvent = “javascriptFunction()”>
```

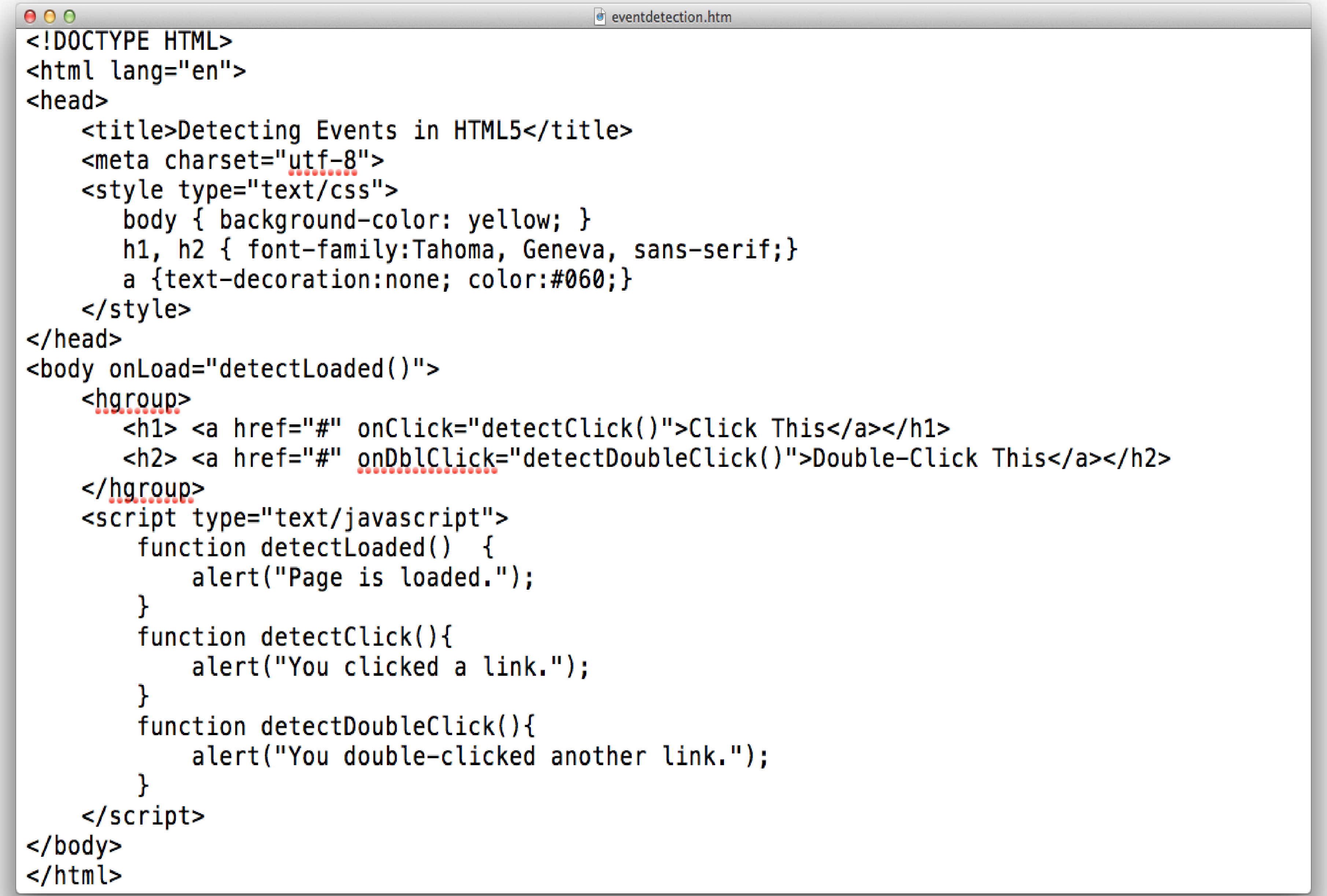
- An example might be:

```
<body onLoad = “announceSomething()”>
```

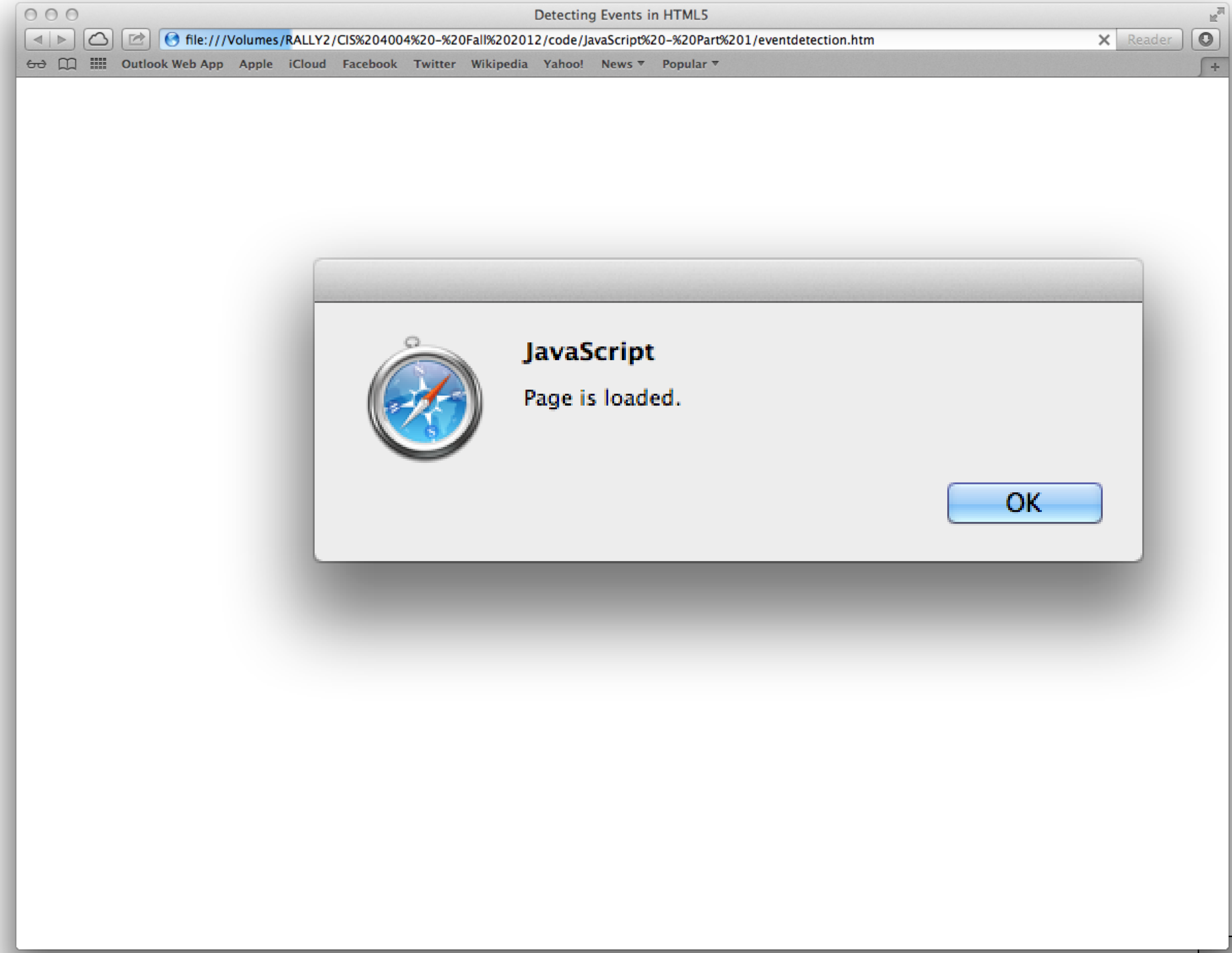
- The example above would use the `body` element with an `onLoad` event handler to fire a JavaScript function named `announceSomething()`.

## Detecting Events

- To see how event handlers work with JavaScript, look at the example markup on the following page.
- This example has three different event handlers and three different JavaScript functions that are launched by the events.
- The first one sends out an alert when the page loads, the second fires when the top link is clicked, and the third event launches an alert when the second link is double-clicked.
- In general, the JavaScript function can be whatever you want there to be, which allows you to interact far more with the visitor.  You can provide instructions, options, cautions, etc..

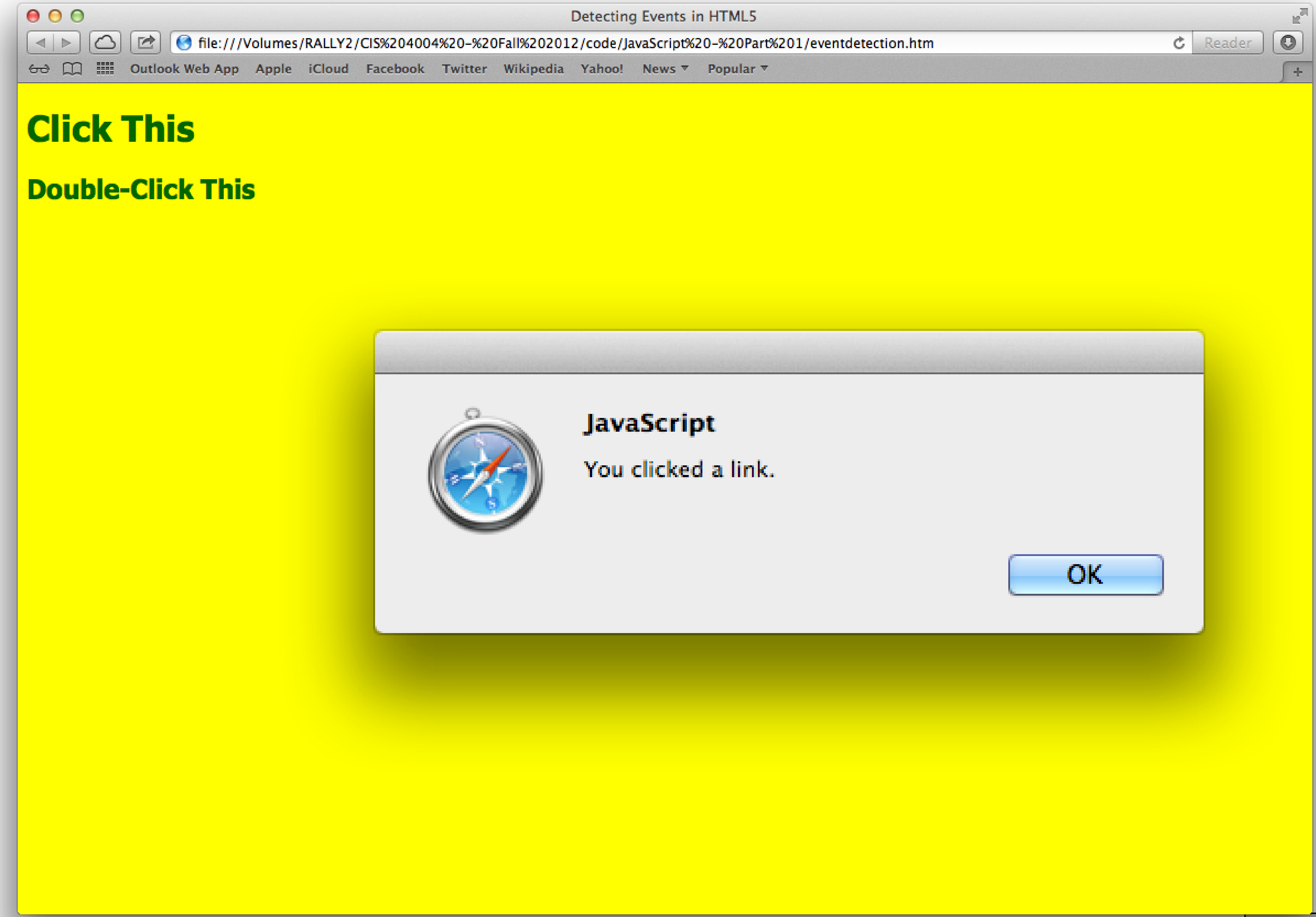


```
<!DOCTYPE HTML>
<html lang="en">
<head>
    <title>Detecting Events in HTML5</title>
    <meta charset="utf-8">
    <style type="text/css">
       body { background-color: yellow; }
       h1, h2 { font-family:Tahoma, Geneva, sans-serif;}
       a {text-decoration:none; color:#060;}
    </style>
</head>
<body onLoad="detectLoaded()">
    <hgroup>
       <h1> <a href="#" onClick="detectClick()">Click This</a></h1>
       <h2> <a href="#" onDblClick="detectDoubleClick()">Double-Click This</a></h2>
    </hgroup>
    <script type="text/javascript">
        function detectLoaded()  {
            alert("Page is loaded.");
        }
        function detectClick(){
            alert("You clicked a link.");
        }
        function detectDoubleClick(){
            alert("You double-clicked another link.");
        }
    </script>
</body>
</html>
```


Detecting Events in HTML5
file:///Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%201/eventdetection.htm
Reader
Outlook Web App
Apple
iCloud
Facebook
Twitter
Wikipedia
Yahoo!
News
Popular

JavaScript

Page is loaded.

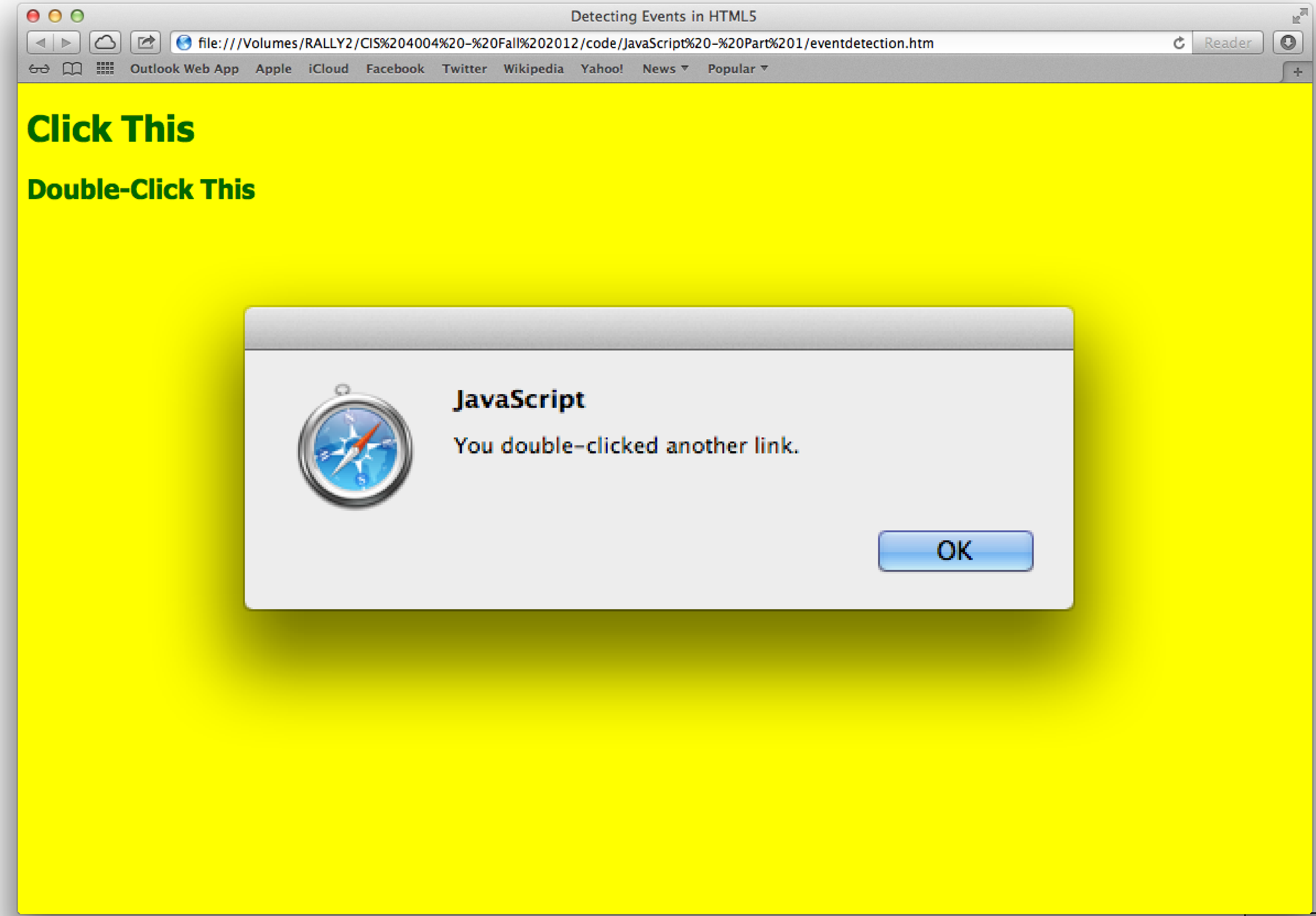

Detecting Events in HTML5
file:///Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%201/eventdetection.htm
Reader
Outlook Web App
Apple
iCloud
Facebook
Twitter
Wikipedia
Yahoo!
News
Popular

Click This

Double-Click This

JavaScript

You clicked a link.


Detecting Events in HTML5
file:///Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%201/eventdetection.htm
Reader
Outlook Web App
Apple
iCloud
Facebook
Twitter
Wikipedia
Yahoo!
News
Popular
OK

Click This

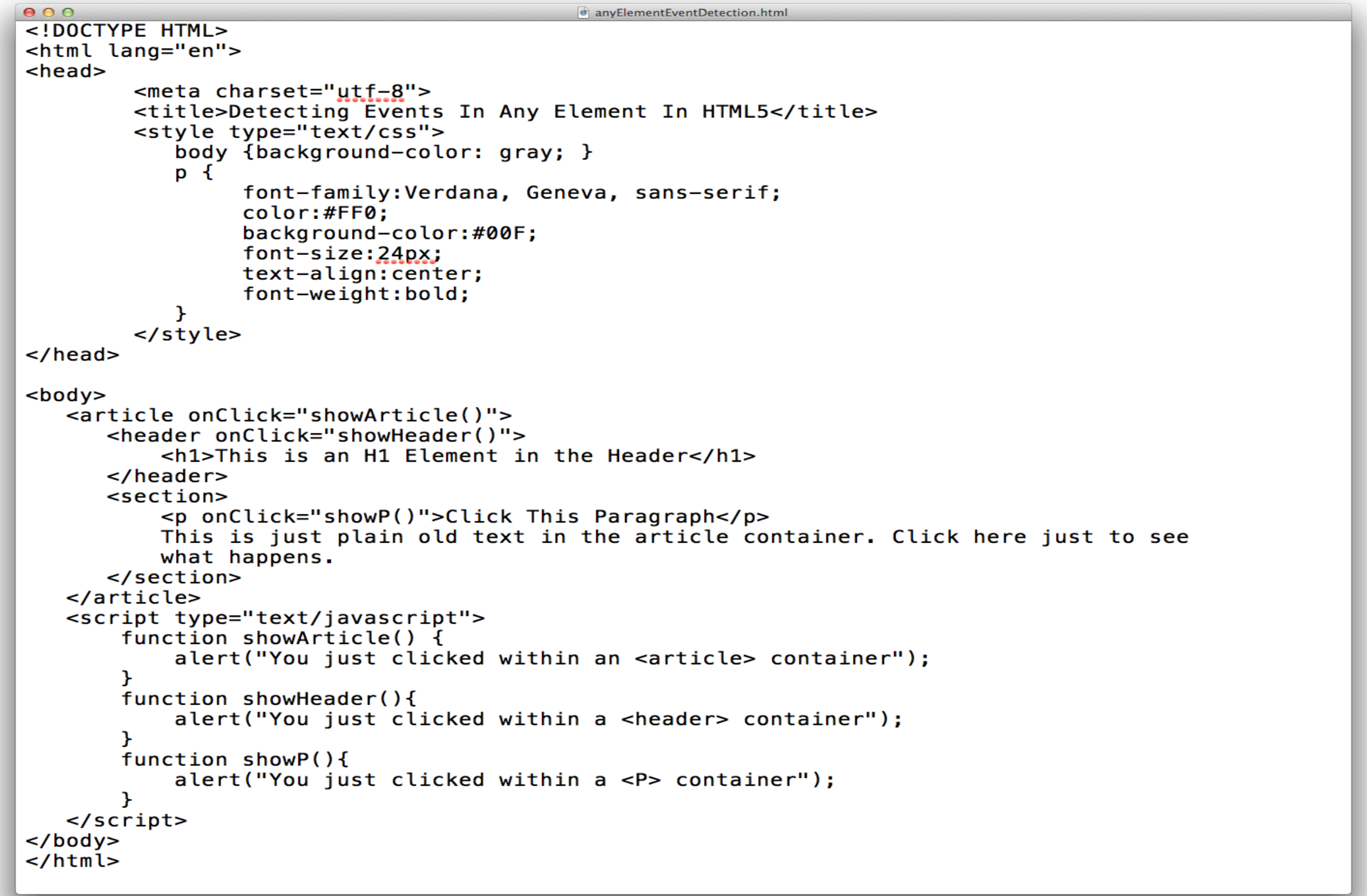
Double-Click This

JavaScript

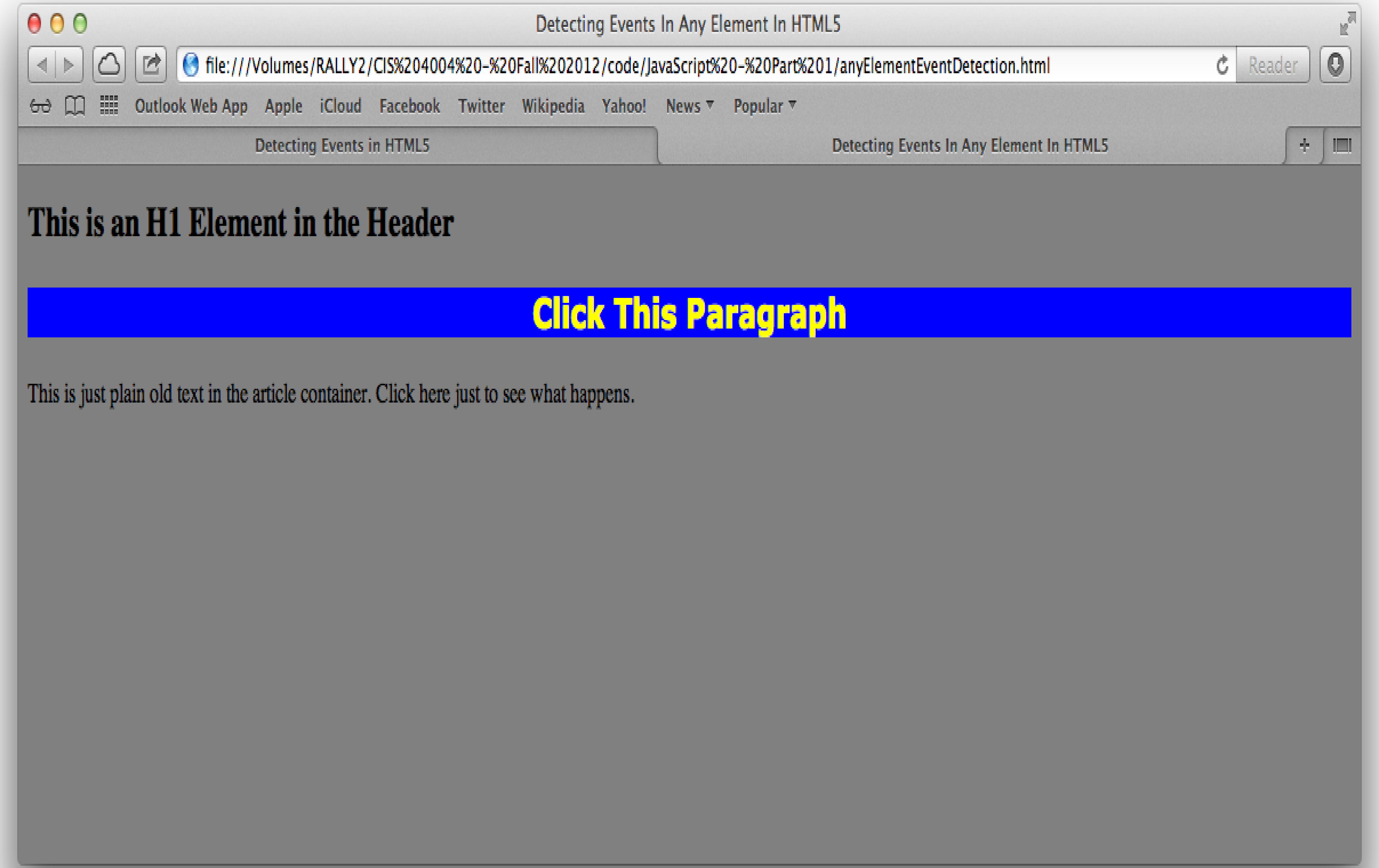
You double-clicked another link.

## Detecting Events

- The previous example doesn't use anything that is new to HTML5. I used it in this example for the simple reason that when the visitor moves the mouse over the anchor elements , the cursor changes shape so that the visitor know that they've moved the cursor over linked text.
- In HTML5 you can set up an event handler in any element.
- The following example illustrates events in `<p>`, `<header>`, and `<article>` elements.



```
<!DOCTYPE HTML>
<html lang="en">
<head>
        <meta charset="utf-8">
        <title>Detecting Events In Any Element In HTML5</title>
        <style type="text/css">
           body {background-color: gray; }
           p {
                font-family:Verdana, Geneva, sans-serif;
                color:#FF0;
                background-color:#00F;
                font-size:24px;
                text-align:center;
                font-weight:bold;
           }
        </style>
</head>

<body>
   <article onClick="showArticle()">
      <header onClick="showHeader()">
          <h1>This is an H1 Element in the Header</h1>
      </header>
      <section>
          <p onClick="showP()">Click This Paragraph</p>
          This is just plain old text in the article container. Click here just to see
          what happens.
      </section>
   </article>
   <script type="text/javascript">
       function showArticle() {
           alert("You just clicked within an <article> container");
       }
       function showHeader(){
           alert("You just clicked within a <header> container");
       }
       function showP(){
           alert("You just clicked within a <P> container");
       }
   </script>
</body>
</html>
```

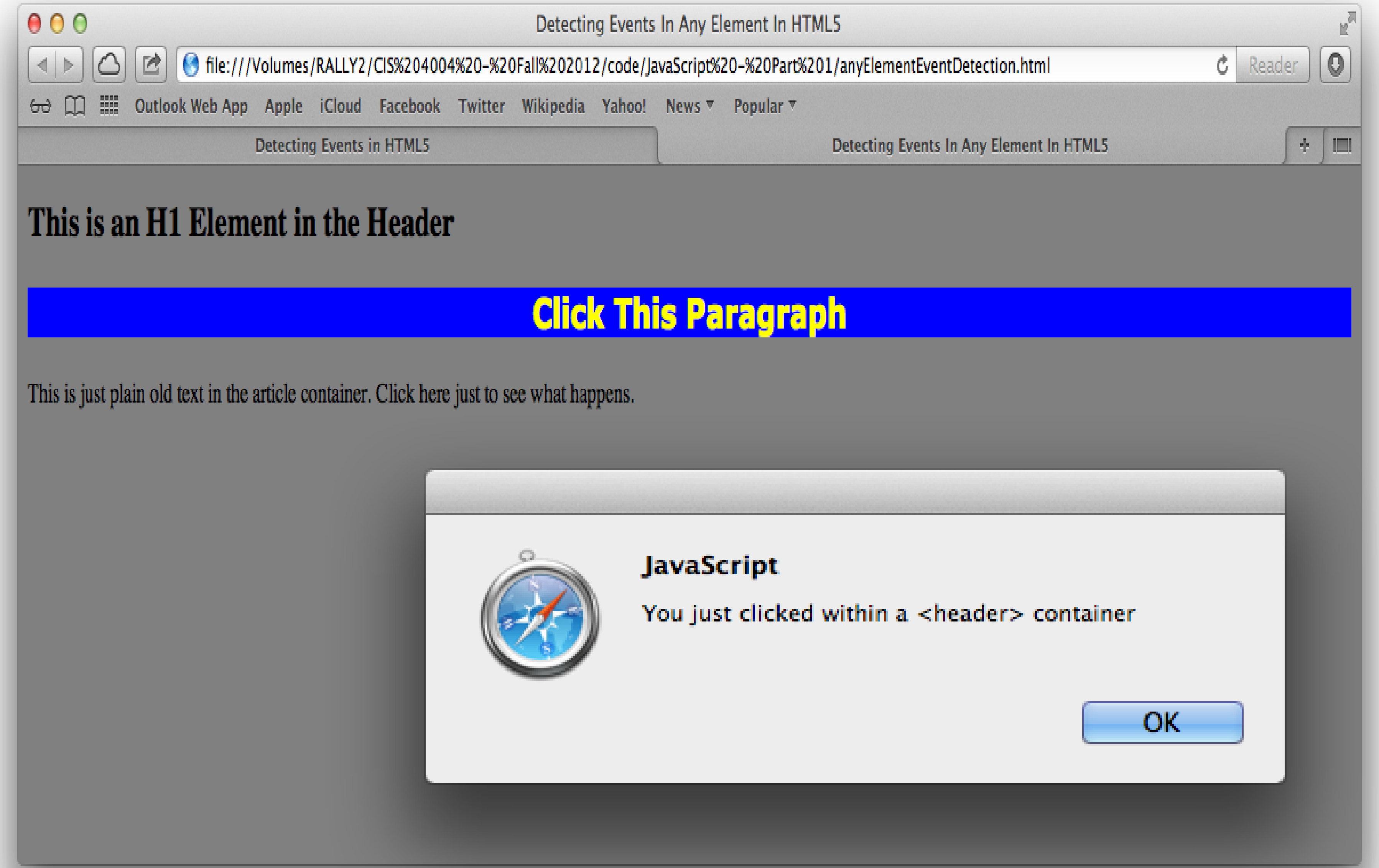

Detecting Events In Any Element In HTML5

file:///Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%201/anyElementEventDetection.html

This is an H1 Element in the Header

Click This Paragraph

This is just plain old text in the article container. Click here just to see what happens.


Detecting Events In Any Element In HTML5
file:///Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%201/anyElementEventDetection.html
Reader
Outlook Web App
Apple
iCloud
Facebook
Twitter
Wikipedia
Yahoo!
News
Popular
Detecting Events in HTML5
Detecting Events In Any Element In HTML5

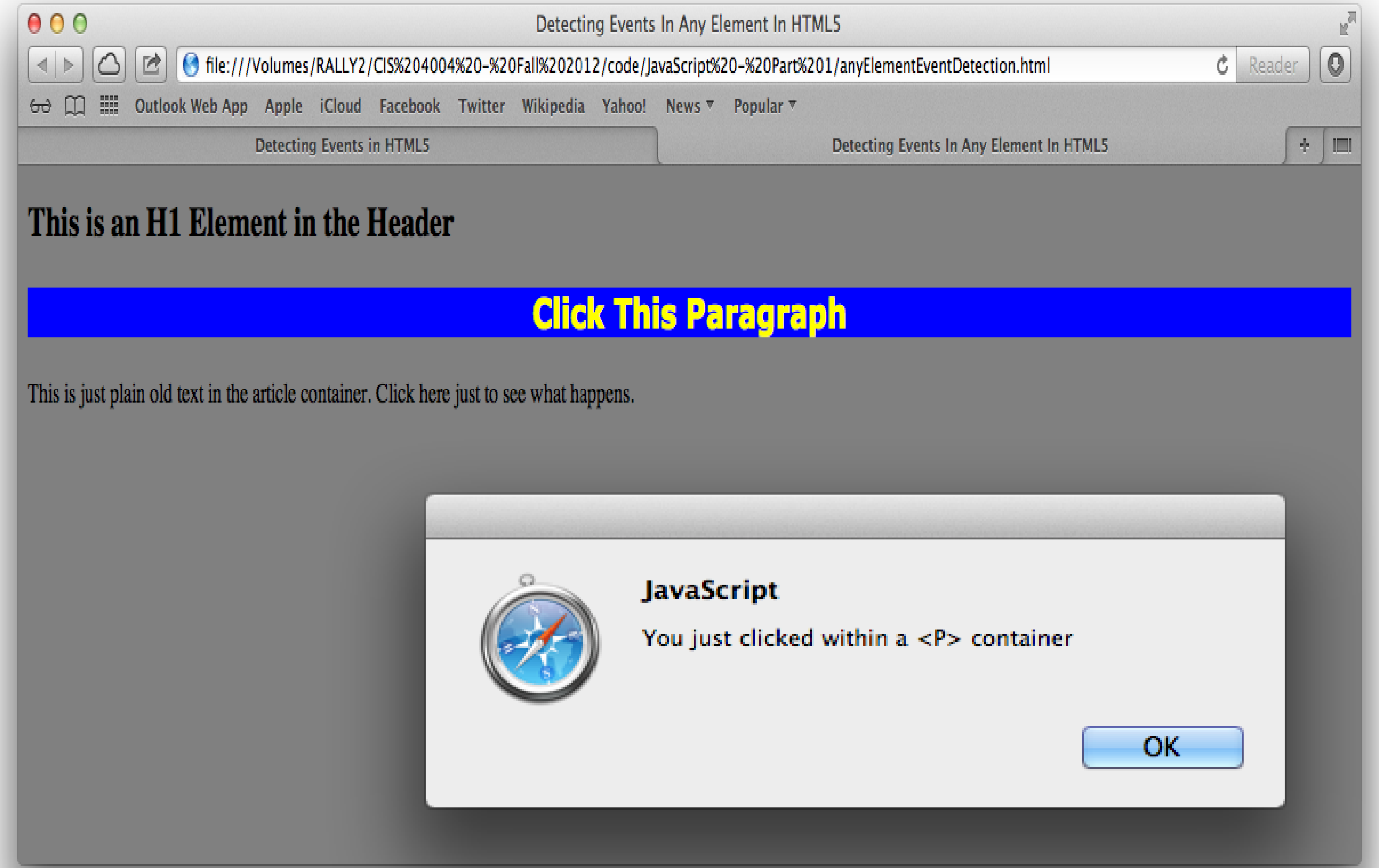
This is an H1 Element in the Header

Click This Paragraph

This is just plain old text in the article container. Click here just to see what happens.

JavaScript

You just clicked within a <header> container


Detecting Events In Any Element In HTML5
file:///Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%201/anyElementEventDetection.html
Reader
Outlook Web App Apple iCloud Facebook Twitter Wikipedia Yahoo! News Popular
Detecting Events in HTML5
Detecting Events In Any Element In HTML5
OK

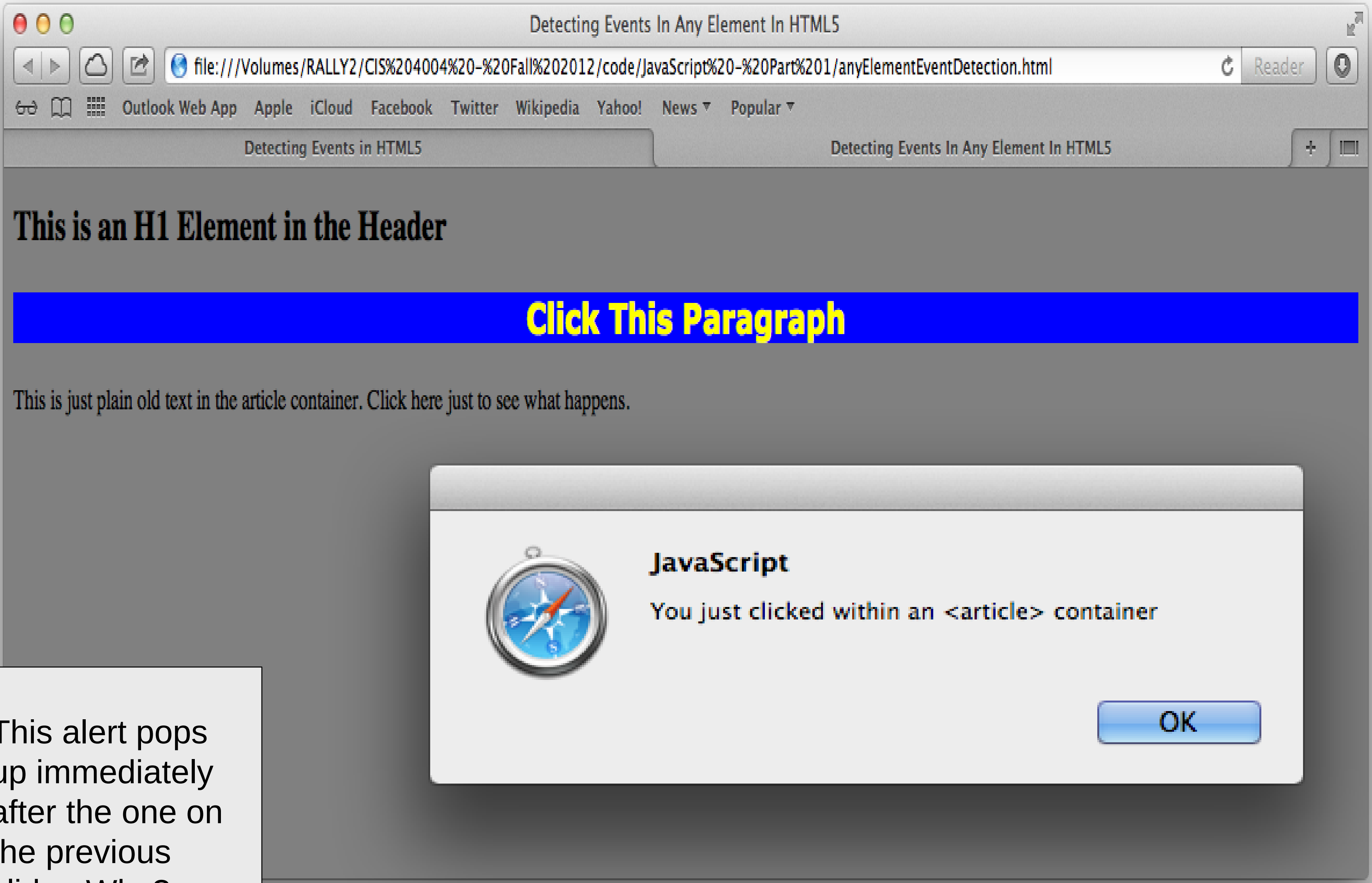
This is an H1 Element in the Header

Click This Paragraph

This is just plain old text in the article container. Click here just to see what happens.

JavaScript

You just clicked within a <P> container



Detecting Events In Any Element In HTML5

file:///Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%20Part%201/anyElementEventDetection.html

This is an H1 Element in the Header

Click This Paragraph

This is just plain old text in the article container. Click here just to see what happens.

JavaScript

You just clicked within an <article> container

This alert pops up immediately after the one on the previous slide. Why?

## Detecting Events

- If you examine the previous example’s markup more carefully, you’ll notice that some events are embedded inside other elements that also have event handlers.
- For instance, all the elements are inside an `<article>` element.
- As you can see from the screen shots of the rendering, what happened when the visitor clicked on the paragraph element that had an event handler?
- Did the visitor see a reaction from the innermost or the outermost event? Look at the previous two slides again.

# Detecting Events

- As soon as the visitor clicked on the line "Click This Paragraph" the event is reported (they clicked in the `<p>` container) in the alert shown on page 39.
- When the visitor clicks the OK button in the JavaScript pop-up, the second alert appears (page 40) letting them know that they had clicked in the `<article>` container as well.
- One way of looking at the events is bubbling up, beginning in the lowest level in the hierarchy (nesting of elements) and then bubbling up to the topmost level of the hierarchy.
- In this example, the hierarchy, from lowest level to highest, is represented by `<p>, (<section>, <header>), <article>, <body>`

# The Document Object Model

- The Document Object Model (DOM) for HTML5 represents a hierarchy tree.
- At the root of every web page or document is the `<html>` element, and the rest of the elements in the page are a branch somewhere along the tree.
- JavaScript uses the DOM for addressing and manipulation a web page beyond what you can do with HTML5 alone.
- The entire DOM tree is a representation of the document that resides in your computer's memory.
- We'll explore the DOM in more detail in the next section of notes.