Front End Engineer

Events, AJAX y Communication Patterns

Topics

Events
AJAX/XHR
Common Communication Patterns

Globant
we are ready

Events

JavaScript events
- allow scripts to respond to user interactions and modify the page accordingly

Events and event handling
- help make web applications more responsive, dynamic and interactive

Now, you can see:

01 Event Basics

http://youtu.be/6Dd41Bt3fYY

Registering Event Handlers

Functions that handle events

Assigning an event handler to an event on a DOM node is called registering an event handler

Two models for registering event handlers

- Inline model treats events as attributes of XHTML elements
- Traditional model assigns the name of the function to the event property of a DOM node

Registering Event Handlers

In the inline model, the value of the XHTML attribute is a JavaScript statement to be executed when the event occurs

In the traditional model, the value of the event property of a DOM node is the name of a function to be called when the event occurs

Traditional registration of event handlers enables quick and easy assignment of event handlers to many elements using repetition statements, instead of adding an inline event handler to each XHTML element

Registering Event Handlers

```
<?xml version = "1.0" encoding = "utf - 8" ?>
<!DOCTYPE html PUBLIC " - //W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1 - strict.dtd" >

<! -- Fig. 13.1: registering.html -- >
<! -- Event registration models. -- >
<html xmlns = "http://www.w3.org/1999/xhtml" >
  <head>
    <title> Event Registration Models </title>
    <style type = "text/css" >
         div { padding: 5px ;
               margin: 10px ;
               border: 3px solid # 0000BB;
               width: 12em }
    </style>
    <script type = "text/javascript" >
      <! --
      // handle the onclick event regardless of how it was registered
         function handleEvent()
      {
        alert( "The event was successfully handled." );
      } // end function handleEvent

      // register the handler using the traditional model
         function registerHandler()
      {
           var traditional = document.getElementById( "traditional" );
           traditional.onclick = handleEvent;
      } // end function registerHandler
```

Function to handle the `onclick` event

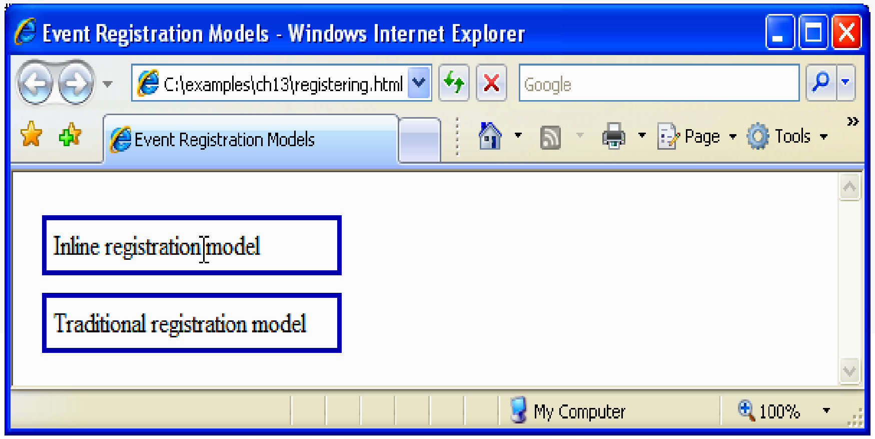
Registers the event handler using the traditional model

Registering Event Handlers

```
      //          -- >
    </script>
  </head>
  <body onload =      "registerHandler()"       >
    <!    --  The event handler is registered inline                -- >
    <div id =        "inline"      onclick =      "handleEvent()"     >
      Inline registration model                    </div>

    <!    --  The event handler is registered by function registerHandler                  -- >
       <div id =       "traditional"      >Traditional registration model            </div>
  </body>
</html>
```

Registers the event handler using the inline model

a) The user clicks the **div** for which the event handler was registered using the inline model.

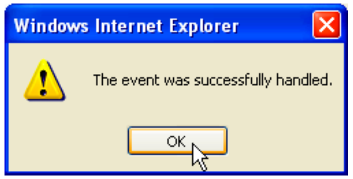

Event Registration Models - Windows Internet Explorer
C:\examples\ch13\registering.html
Google
Event Registration Models
Page
Tools
My Computer
100%

Inline registration model

Traditional registration model

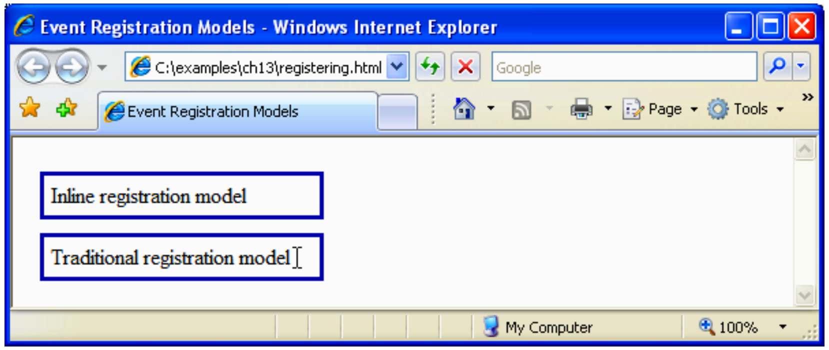
Registering Event Handlers

b) The event handler displays an alert dialog.


Windows Internet Explorer

The event was successfully handled.

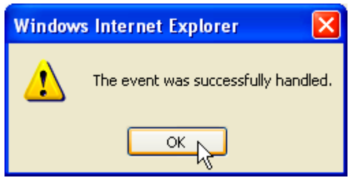
c) The user clicks the **div** for which the event handler was registered using the traditional model.


Event Registration Models - Windows Internet Explorer
C:\examples\ch13\registering.html
Google
Event Registration Models
Page
Tools
My Computer
100%

Inline registration model

Traditional registration model

d) The event handler displays an alert dialog..


Windows Internet Explorer

The event was successfully handled.

Common Programming Error

Putting quotes around the function name when registering it using the traditional model would assign a string to the onclick property of the node: a string cannot be called.

Putting parentheses after the function name when registering it using the traditional model would call the function immediately and assign its return value to the onclick property.

Events on load

onload event fires whenever an element finishes loading successfully

If a script in the head attempts to get a DOM node for an XHTML element in the body, getElementById returns null because the body has not yet loaded

Events on load

```
<?xml version = "1.0" encoding = "utf - 8" ?>
<!DOCTYPE html PUBLIC " - //W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1 - strict.dtd" >

<! -- Fig. 13.2: onload.html -- >
<! -- Demonstrating the onload event. -- >
<html xmlns = "http://www.w3.org/1999/xhtml" >
  <head>
    <title> onload Event </title>
    <script type = "text/javascript" >
          <! --
      var seconds = 0;

      // called when the page loads to begin the timer
      function startTimer()
      {
            // 1000 milliseconds = 1 second
            window.setInterval( "updateTime()" , 1000 ) ;
          } // end function startTimer

      // called every 1000 ms to update the timer
      function updateTime()
      {
            ++seconds;
            document.getElementById( "soFar" ).innerHTML = seconds;
          } // end function updateTime
          // -- >
    </script>
  </head>
```

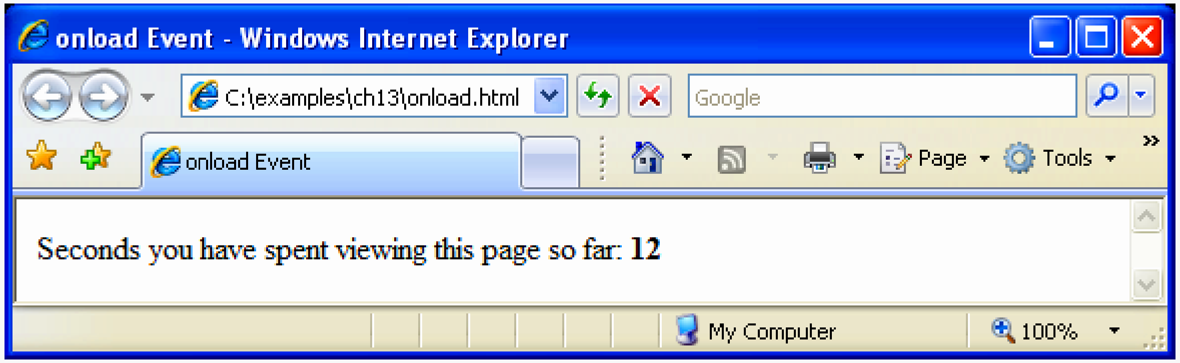
Calls function `updateTime` every second

Updates the timer display in the `soFar` element of the document

Events on load

```
<body onload =       "startTimer()"        >
   <p>      Seconds you have spent viewing this page so far:
   <strong id =             "soFar"    >0</strong></p>
 </body>
</html>
```

As soon as the `body` has loaded, `startTimer` is called


onload Event - Windows Internet Explorer
C:\examples\ch13\onload.html
Google
onload Event
Page
Tools
My Computer
100%

Seconds you have spent viewing this page so far: **12**

Common Programming Error

Trying to get an element in a page before the page has loaded is a common error. Avoid this by putting your script in a function using the onload event to call the function.

Event onmouseMove, the event object and this

onmousemove event fires whenever the user moves the mouse

event object stores information about the event that called the event-handling function

ctrlKey property contains a boolean which reflects whether the *Ctrl* key was pressed during the event

shiftKey property reflects whether the *Shift* key was pressed during the event

Event onmouseMove, the event object and this

In an event-handling function, this refers to the DOM object on which the event occurred

this keyword enables one event handler to apply a change to one of many DOM elements, depending on which one received the event

Example

```
<?xml version =        "1.0"    encoding =       "utf   - 8" ?>
<!DOCTYPE html PUBLIC      " - //W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1              - strict.dtd"       >

<! --   Fig.      13.3: draw.html         -- >
<! --   A simple drawing program.              -- >
<html x    mlns =    "http://www.w3.org/1999/xhtml"             >
  <head>
         <title>    Simple Drawing Program       </title>
        <style type =         "text/css"       >
             #canvas  {    width:     400px ;
                           border:      1px solid #999999       ;
                           borde  r - collapse:       collapse   }
             td       {    width:     4px ;
                           height:     4px  }
             th.key      {   font   - family:      arial, helvetica, sans             - serif    ;
                           font   - size:     12px ;
                           border   - bottom:     1px solid #      999999  }
       </style>
         <script type =         "text/javascript"          >
      <!        --
      //initialization function to insert cells into the table
             function     createCanvas ()
      {
                 var  side =      100 ;
                 var  tbody = document.getElementById(           "tablebody"      );

```

Sets the dimensions of a table of cells that will act as a canvas

Eliminates space between table cells

Creates table of cells for the canvas

Example

```
            for ( var i = 0; i < side; i++ )
        {
                var row = document.createElement( "tr" );

                for ( var j = 0; j < side; j++ )
          {
                    var cell = document.createElement( "td" );
                    cell.onmousemove = processMouseMove;
            row.appendChild( cell );
          }            // end for

          tbody.appendChild( row );
        }          // end for
      }         // end function createCanvas

      // processes the onmousemove event
          function    processMouseMove( e )
      {
        // get the event object from IE
            if  ( !e )
                var  e = window.event;

        // turn the cell blue if the Ctrl key is pressed
            if  ( e.ctrlKey )
                this .style.backgroundColor = "blue" ;

        // turn the cell red if the Shift key is pressed
            if  ( e.shiftKey )
                this .style.backgroundColor = "red" ;
      }         // end function processMouseMove
```

Assigns `processMouseMove` as the event handler for the cell's `onmousemove` event

Gets the `event` object in Firefox

Gets the `event` object in IE

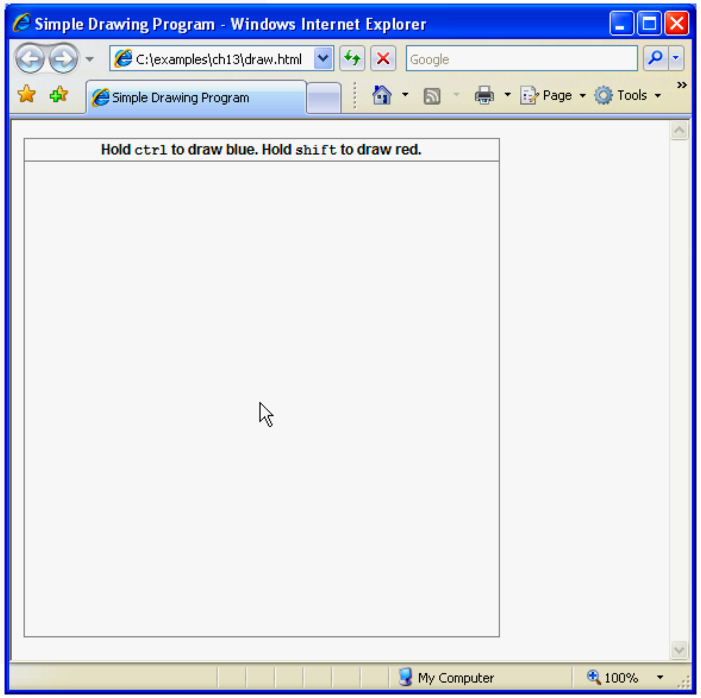
Determines which key is pressed and colors the cell accordingly

`this` refers to the cell that received the event

Example

```
            //  -- >
    </script>
  </head>
    <body onload =  "createCanvas()"   >
        <table id =   "canvas"  class =  "canvas"  ><tbody id =   "tablebody"   >
        <tr><th class =    "key"  colspan =   "100"  >Hold <tt> ctrl  </tt>
      to        draw blue. Hold     <tt> shift  </tt>  to draw red.    </th></tr>
        </tbody></table>
  </body>
</html>
```

a) The page loads and fills with white cells. With no keys held down, moving the mouse does not draw anything.

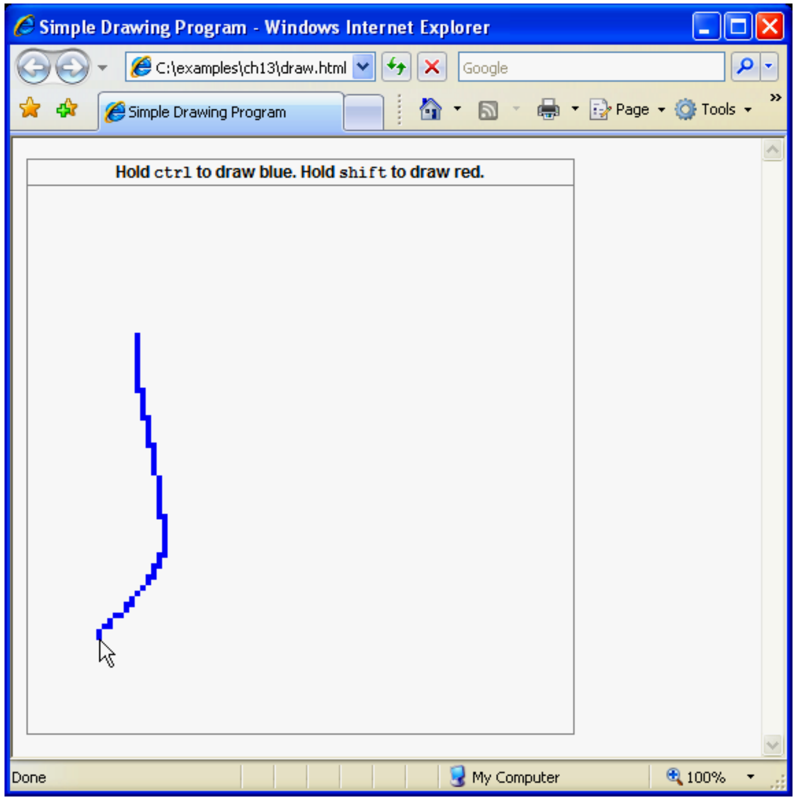

Simple Drawing Program - Windows Internet Explorer

C:\examples\ch13\draw.html

Hold ctrl to draw blue. Hold shift to draw red.

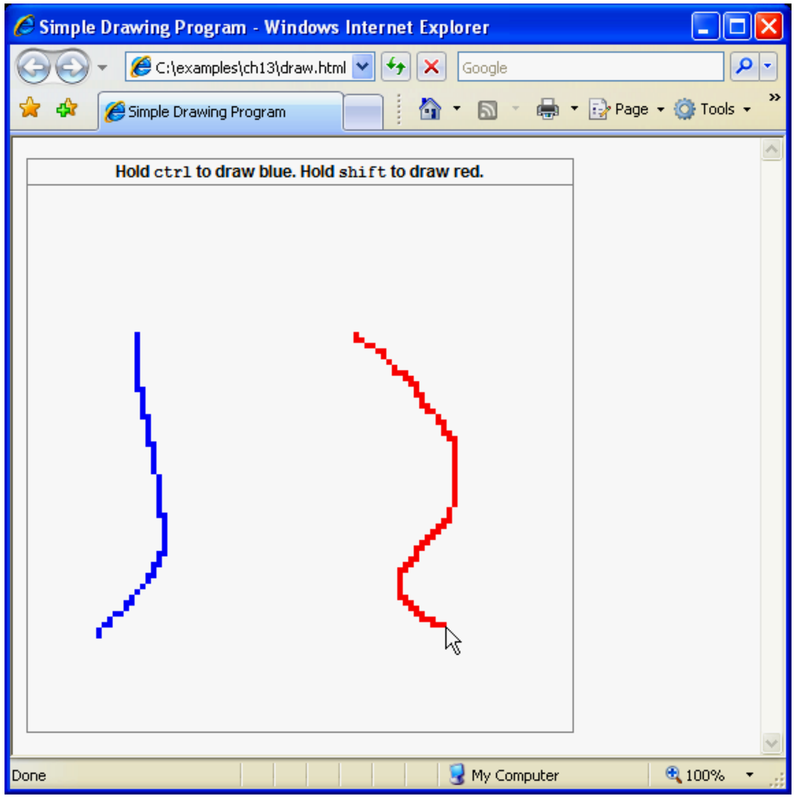
Example

b) The user holds the *Ctrl* key and moves the mouse to draw a blue line.


Simple Drawing Program - Windows Internet Explorer
C:\examples\ch13\draw.html
Google
Simple Drawing Program
Page
Tools
Done
My Computer
100%

Hold `ctrl` to draw blue. Hold `shift` to draw red.

c) The user holds the *Shift* key and moves the mouse to draw a red line.


Simple Drawing Program - Windows Internet Explorer
C:\examples\ch13\draw.html
Google
Simple Drawing Program
Page
Tools
Done
My Computer
100%

Hold `ctrl` to draw blue. Hold `shift` to draw red.

Common Programming Error

Although you can omit the tbody element in an XHTML table, without it you cannot append tr elements as children of a table using JavaScript.

While Firefox treats appended rows as members of the table body, Internet Explorer will not render any table cells that are dynamically added to a table outside a thead, tbody or tfoot element.

Some event object properties

Property	Description
altKey	This value is true if the *Alt* key was pressed when the event fired.
cancelBubble	Set to true to prevent the event from bubbling. Defaults to false. (See Section 14.9, Event Bubbling.)
clientX and clientY	The coordinates of the mouse cursor inside the client area (i.e., the active area where the web page is displayed, excluding scrollbars, navigation buttons, etc.).
ctrlKey	This value is true if the *Ctrl* key was pressed when the event fired.
keyCode	The ASCII code of the key pressed in a keyboard event. See Appendix D for more information on the ASCII character set.
screenX and screenY	The coordinates of the mouse cursor on the screen coordinate system.
shiftKey	This value is true if the *Shift* key was pressed when the event fired.
type	The name of the event that fired, without the prefix "on".

Now, you can see:

02 The Standard Event Model
03 The Legacy IE Event Model

http://youtu.be/OCFCrwYxPT4

http://youtu.be/OW8tF2Kd8tk

Rollovers with onmouseover and onmouseout

When the mouse cursor enters an element, an onmouseover event occurs for that element

When the mouse cursor leaves the element, an onmouseout event occurs for that element

Creating an Image object and setting its src property preloads the image

The event object stores the node on which the action occurred

In Internet Explorer, this node is stored in the event object's srcElement property

In Firefox, it is stored in the event object's target property

Example

```
<?xml version = "1.0" encoding = "utf - 8" ?>
<!DOCTYPE html PUBLIC " - //W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1 - strict.dtd" >

<! -- Fig. 13.5: onmouseoverout.html -- >
<! -- Events onmouseover and onmouseo ut. -- >
<html xmlns = "http://www.w3.org/1999/xhtml" >
  <head>
    <title> Events onmouseover and onmouseout </title>
    <style type = "text/css" >
         body  { background - color: wheat }
         table { border - style: groo ve;
                 text - align: center ;
                 font - family: monospace;
                 font - weight: bold }
         td    { width: 6em}
    </style>
      <script type = "text/javascript" >
        <! --
         image1 = new Image();
         image1.src = "heading1.gif" ;
         image2 = new Image();
         image2.src = "heading2.gif" ;

```

Preloads the heading images

Example

```
      function   mouseOver( e )
   {
         if  ( !e )
            var  e = window.event;

         var  target = getTarget( e );

     // swap the image when the mouse moves over it
         if  ( target.id ==        "heading"    )
     {
            target.src = image2.src;
            return   ;
     }         // end if

     // if an element's id is defined, assign the id to it                           s color
     // to turn hex code's text the corresponding color
         if  ( target.id )
            target.style.color = target.id;
   }        // end function mouseOver

      function   mouseOut( e )
     {
         if  ( !e )
            var  e = window.event;

         var  target = getTarget( e );

```

Stores the return value of `getTarget` to variable `target`—we can't use `this` because we have not defined an event handler for each element in the document

Changes the heading's image to `image2`

If `target` has a defined `id` (true of table cells and the heading), changes its color to that `id`

Example

```
         // put the original image back when the mouse moves away
                  if  ( target.id ==        "heading"    )
         {
                     target.src = image1.src;
                     return  ;
         }          // end if

            // if an element's id is defined, assign id to innerHTML
         // to display the color name
                  if  ( target.id )
                     target.innerHTML = target.id;
       }         // end function mouseOut

              // return either e.srcElement or e.target, whichever exists
              function     getTarget( e )
       {
                  if  ( e.srcElement )
                     return    e.srcElement;
                  else
                     return    e.target;
       }         // end function getTarget

              document.onmouseover = mouseOver;
              document.onmouseout = mouseOut;
       //         -- >
     </script>
   </head>
```

Replaces `image2` with `image1`

If the element's `id` is defined, makes the displayed text equal to the `id`

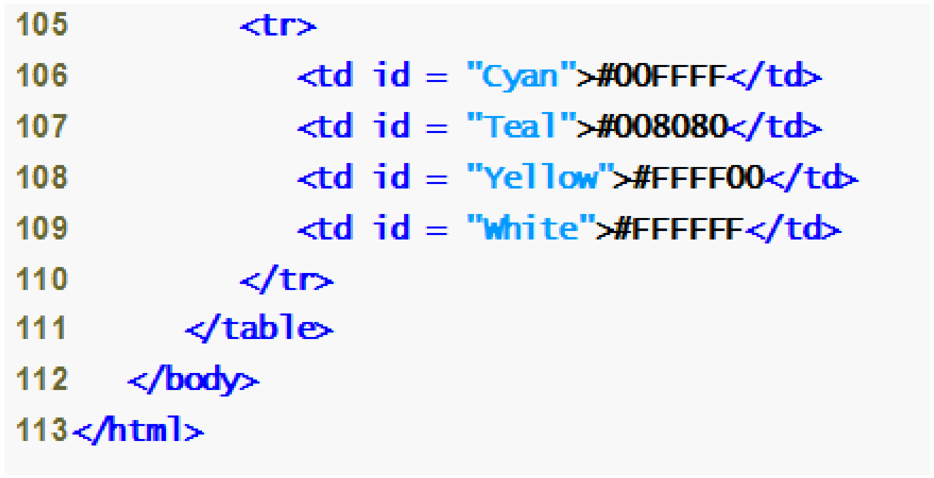
Returns the targeted node in both Internet Explorer and Firefox

Registers the `onmouseover` and `onmouseout` events in the `document` object

Example

```html
   <body>
      <img src = "heading1.gif" id = "heading" alt = "Heading Image" />
      <p>Can you tell a color from its hexadecimal RGB code
      value? Look at the hex code, guess its color. To see
      what color it corresponds to, move the mouse over the
      hex code. Moving the mouse out of the hex code's table
      cell will display the color name.</p>
      <table>
         <tr>
            <td id = "Black">#000000</td>
            <td id = "Blue">#0000FF</td>
            <td id = "Magenta">#FF00FF</td>
            <td id = "Gray">#808080</td>
         </tr>
         <tr>
            <td id = "Green">#008000</td>
            <td id = "Lime">#00FF00</td>
            <td id = "Maroon">#800000</td>
            <td id = "Navy">#000080</td>
         </tr>
         <tr>
            <td id = "Olive">#808000</td>
            <td id = "Purple">#800080</td>
            <td id = "Red">#FF0000</td>
            <td id = "Silver">#C0C0C0</td>
         </tr>
```

Example

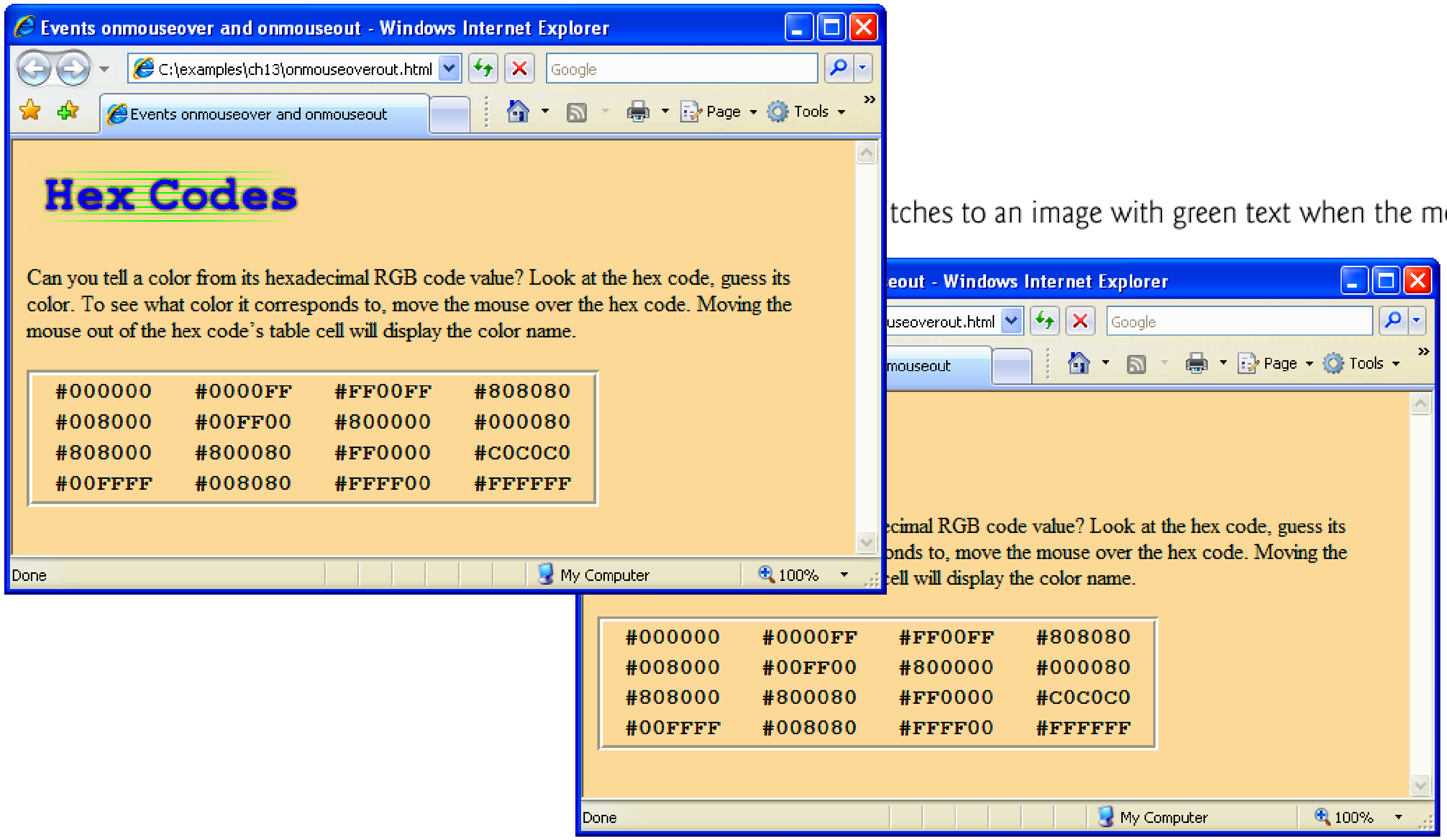


```
            <tr>
               <td id = "Cyan">#00FFFF</td>
               <td id = "Teal">#008080</td>
               <td id = "Yellow">#FFFF00</td>
               <td id = "White">#FFFFFF</td>
            </tr>
         </table>
      </body>
   </html>
```

Example

a) The page loads with the blue heading image and all the hex codes in black.

tches to an image with green text when the mouse rolls over it.



Events onmouseover and onmouseout - Windows Internet Explorer

C:\examples\ch13\onmouseoverout.html

Hex Codes

Can you tell a color from its hexadecimal RGB code value? Look at the hex code, guess its color. To see what color it corresponds to, move the mouse over the hex code. Moving the mouse out of the hex code's table cell will display the color name.

#000000	#0000FF	#FF00FF	#808080
#008000	#00FF00	#800000	#000080
#808000	#800080	#FF0000	#C0C0C0
#00FFFF	#008080	#FFFF00	#FFFFFF

eout - Windows Internet Explorer

useoverout.html

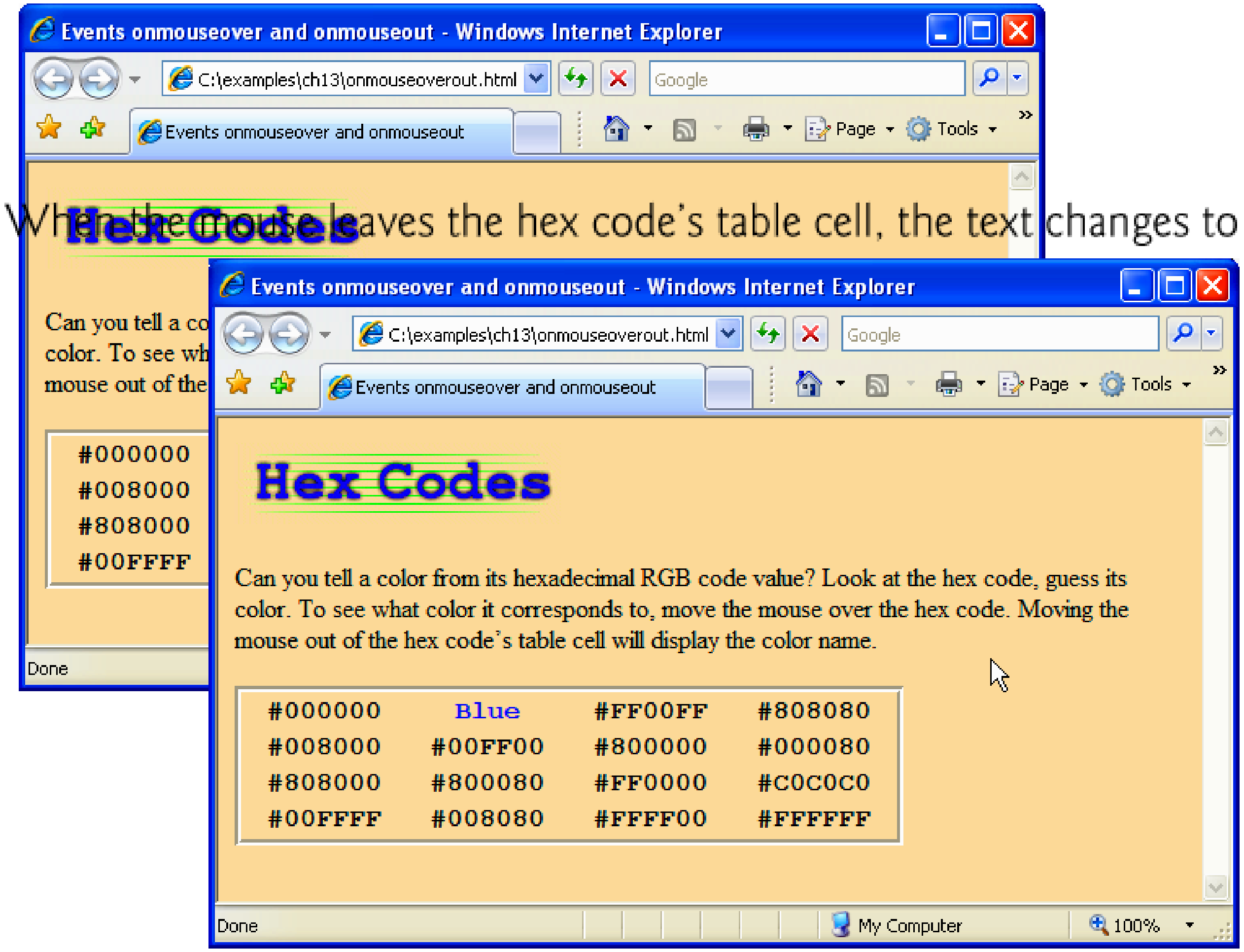
cimal RGB code value? Look at the hex code, guess its onds to, move the mouse over the hex code. Moving the ell will display the color name.

#000000	#0000FF	#FF00FF	#808080
#008000	#00FF00	#800000	#000080
#808000	#800080	#FF0000	#C0C0C0
#00FFFF	#008080	#FFFF00	#FFFFFF

Example

c) When mouse rolls over a hex code, the text color changes to the color represented by the hex code. Notice that the heading image has become blue again because the mouse is no longer over it.

d) When the mouse leaves the hex code's table cell, the text changes to the name of the color.


Events onmouseover and onmouseout - Windows Internet Explorer

C:\examples\ch13\onmouseoverout.html

Hex Codes

Can you tell a color from its hexadecimal RGB code value? Look at the hex code, guess its color. To see what color it corresponds to, move the mouse over the hex code. Moving the mouse out of the hex code's table cell will display the color name.

#000000	Blue	#FF00FF	#808080
#008000	#00FF00	#800000	#000080
#808000	#800080	#FF0000	#C0C0C0
#00FFFF	#008080	#FFFF00	#FFFFFF

Tips

Preloading images used in rollover effects prevents a delay the first time an image is displayed.

Form Processing with onfocus and onblur

onfocus event fires when an element gains focus
i.e., when the user clicks a form field or uses the *Tab* key to move between form elements

onblur fires when an element loses focus
i.e., when another control gains the focus

Example

```
<?xml version = "1.0" encoding = "utf - 8" ?>
<!DOCTYPE html PUBLIC " - //W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1 - strict.dtd" >

<! -- Fig. 13.6: onfocusblur.html -- >
<! -- Demonstrating the onfocus and onbl ur events. -- >
<html xmlns = "http://www.w3.org/1999/xhtml" >
  <head>
    <title> A Form Using onfocus and onblur </title>
    <style type = "text/css" >
          .tip { font - family: sans - serif ;
                 color: blue ;
                 font - size: 12px }
    </style>
    <script type = "text/javascript" >
          <! --
      var helpArray =
            [ "Enter your name in this input box." , // element 0
        "Enter your e - mail address in this input box, " +
        "in the format user@domain." , // element 1
        "Check this box if you liked our site." , // element 2
        "In this box, enter any comments you would " +
        "l ike us to read." , // element 3
        "This button submits the form to the " +
        "server - side script." , // element 4
        "This button clears the form." , // element 5
              "" ]; // element 6

```

Array of help messages

Example

```
      function              helpText( messageNum )
            {
               document.getElementById(      "tip"     ).innerHTML =
                  helpArray[ messageNum ];
            }  // end function helpText
            //    -- >
      </script>
   </head>
   <body>
     <form id =            "myForm"   action =      ""   >
       <div>
            Name:  <input type =        "text"     name =    "name"
               onfocus =     "helpText(0)"       onblur =      "helpText(6)"       /><br />
            E- mail:     <input type =        "text"     name =    "e - mail"
               onfocus =     "helpText(1)"       onblur =      "helpText(6)"       /><br />
            Click here if you like this site
       <input type =                  "checkbox"     name =    "like"     onfocus =
               "helpText(2)"        onblur =      "helpText(6)"       /><br /><hr />

            Any comments?  <br />
       <textarea name =                  "comments"    rows =     "5"   cols =      "45"
               onfocus =     "helpText(3)"       onblur =      "helpText(6)"       ></textarea>
       <b          r />
       <input type =                  "submit"     value =     "Submit"     onfocus =
               "helpText(4)"        onblur =      "helpText(6)"       />
       <input type =                  "reset"     value =     "Reset"     onfocus =
               "helpText(5)"        onblur =      "helpText(6)"       />
            </div>
        </form>
```

Displays the corresponding help message in the `div` element at the bottom of the document

When a user clicks into a field, the `onfocus` event is fired, which feeds the appropriate message number to function `helpText` in order to display the help message

When an element loses focus, the `onblur` event is fired, and `helpText(6)` is called, clearing the old message from the screen

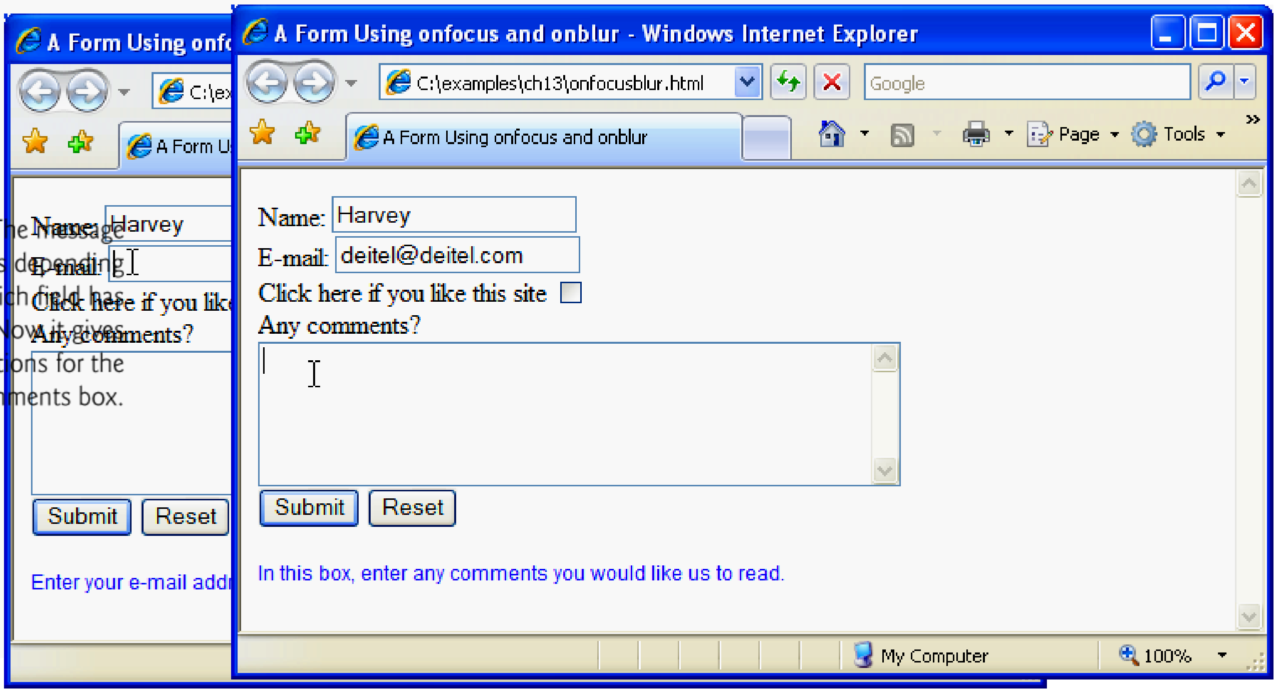
Example

```
            <div id =     "tip"    class =     "tip"    ></div>
        </body>
    </html>
```

`div` element where the help message is displayed

a) The blue message at the bottom of the page instructs the user to enter an e-mail when the e-mail field has focus.

b) The message changes depending on which field has focus. Now it gives instructions for the comments box.



A Form Using onfocus and onblur - Windows Internet Explorer

C:\examples\ch13\onfocusblur.html

Name: Harvey

E-mail: deitel@deitel.com

Click here if you like this site ☐

Any comments?

Submit Reset

In this box, enter any comments you would like us to read.

Enter your e-mail addr

More Form Processing with onsubmit and onreset

onsubmit and onreset events fire when a form is submitted or reset, respectively

Anonymous function

A function that is defined with no name

Created in nearly the same way as any other function, but with no identifier after the keyword function

Useful when creating a function for the sole purpose of assigning it to an event handler

confirm method asks the users a question, presenting them with an OK button and a Cancel button

If the user clicks OK, confirm returns true; otherwise, confirm returns false

More Form Processing with onsubmit and onreset

By returning either true or false, event handlers dictate whether the default action for the event is taken

If an event handler returns true or does not return a value, the default action is taken once the event handler finishes executing

Example

```
<?xml version = "1.0" encoding = "utf - 8" ?>
<!DOCTYPE html PUBLIC " - //W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1 - strict.dtd" >

<! -- Fig. 13.7: onsubmitreset.html -- >
<! -- Demonstrating the onsubmit and onreset events. -- >
<html xmlns = "http://www.w3.org/1999/xhtml" >
  <head>
    <title> A Form Using onsubmit and onreset </title>
    <style type = "text/css" >
          .tip { font - family: sans - serif ;
                 color: blu e;
                 font - size: 12px }
    </style>
    <script type = "text/javascript" >
          <! --
      var helpArray =
            [ "Enter your name in this input box." ,
         "Enter your e - mail addr ess in this input box, " +
         "in the format user@domain." ,
         "Check this box if you liked our site." ,
         "In this box, enter any comments you would " +
         "like us to read." ,
              "This button submits the form to the " +
         "server - side script." ,
         "This button clears the form." ,
              "" ];

```

Example

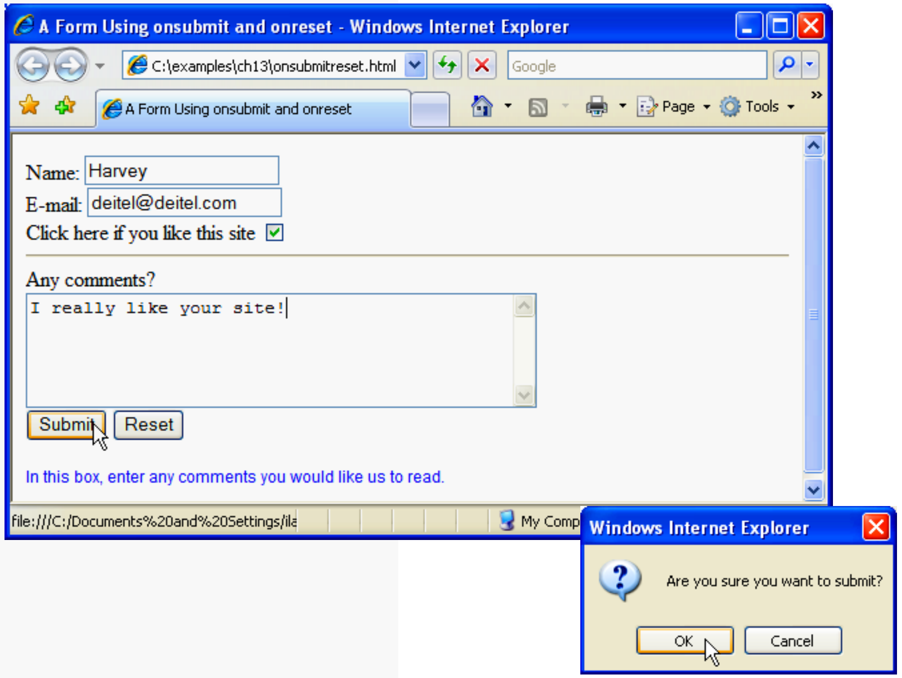
```
function helpText( messageNum )
    {
      document.getElementById( "tip" ).innerHTML =
    helpArray[ messageNum ];
} // end function helpText

    function registerEvents()
{
        document.getElementById( "myForm" ).onsubmit = function ()
        {
          return confirm( "Are you sure you want to submit?" );
        } // end anonymous function

        document.getElementById( "myForm" ).onreset = function ()
        {
          return confirm( "Are you sure you want to reset?" );
        } // end anonymous function
} // end function registerEvents
    // -- >
  </script>
</head>
<body onload = "registerEvents()" >
  <form id = "myForm" action = "" >
    <div>
        Name: <input type = "text" name = "name"
      onfocus = "helpText(0)" onblur = "helpText(6)" /><br />
        E- mail: <input type = "text" name = "e - mail"
          onfocus = "helpText(1)" onblur = "helpText(6)" /><br />
        Click here if you like this site
```

Creates an anonymous function to register as an event handler for the `onsubmit` event

Uses `confirm` to return a boolean stating whether or not the form should be submitted or reset

Example

```
        <input type = "checkbox" name = "like" onfocus =
            "helpText(2)" onblur = "helpText(6)" /><br /><hr />

            Any comments? <br />
        <textarea name = "comments" rows = "5" cols = "45"
            onfocus = "helpText(3)" onblur = "helpText(6)" ></textarea>
        <br />
        <input type = "submit" value = "Submit" onfocus =
            "helpText(4)" onblur = "helpText(6)" />
        <input type = "reset" value = "Reset" onfocus =
            "helpText(5)" onblur = "helpText(6)" />
            </div>
        </form>
        <div id = "tip" class = "tip" ></div>
    </body>
</html>
```


A Form Using onsubmit and onreset - Windows Internet Explorer

C:\examples\ch13\onsubmitreset.html

Name: Harvey
E-mail: deitel@deitel.com
Click here if you like this site ☑

Any comments?
I really like your site!

Submit Reset

In this box, enter any comments you would like us to read.

file:///C:/Documents%20and%20Settings/ila

Windows Internet Explorer
Are you sure you want to submit?
OK Cancel

Event Bubbling

Event bubbling

The process whereby events fired in child elements “bubble” up to their parent elements

When an event is fired on an element, it is first delivered to the element’s event handler (if any), then to the parent element’s event handler (if any)

If you intend to handle an event in a child element alone, you should cancel the bubbling of the event in the child element’s event-handling code by using the cancelBubble property of the event object

Example

```
<?xml version = "1.0" encoding = "utf - 8" ?>
<!DOCTYPE html PUBLIC " - //W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1 - strict.dtd" >

<! -- Fig. 13.8: bubbling.html -- >
<! -- Canceling event bubbling. -- >
<ht ml xmlns = "http://www.w3.org/1999/xhtml" >
  <head>
        <title> Event Bubbling </title>
        <script type = "text/javascript" >
     <! --
            function documentClick()
     {
       alert( "You clicked in the document." );
     } // end function documentClick

            function bubble( e )
     {
                if ( !e )
         var e = window.event;

       alert( "This will bubble." ) ;
                e.cancelBubble = false ;
     } // end function bubble

            function noBubble( e )
     {
                if ( !e )
                    var e = window.event;

```

Does not cancel bubbling, which is the default

Example

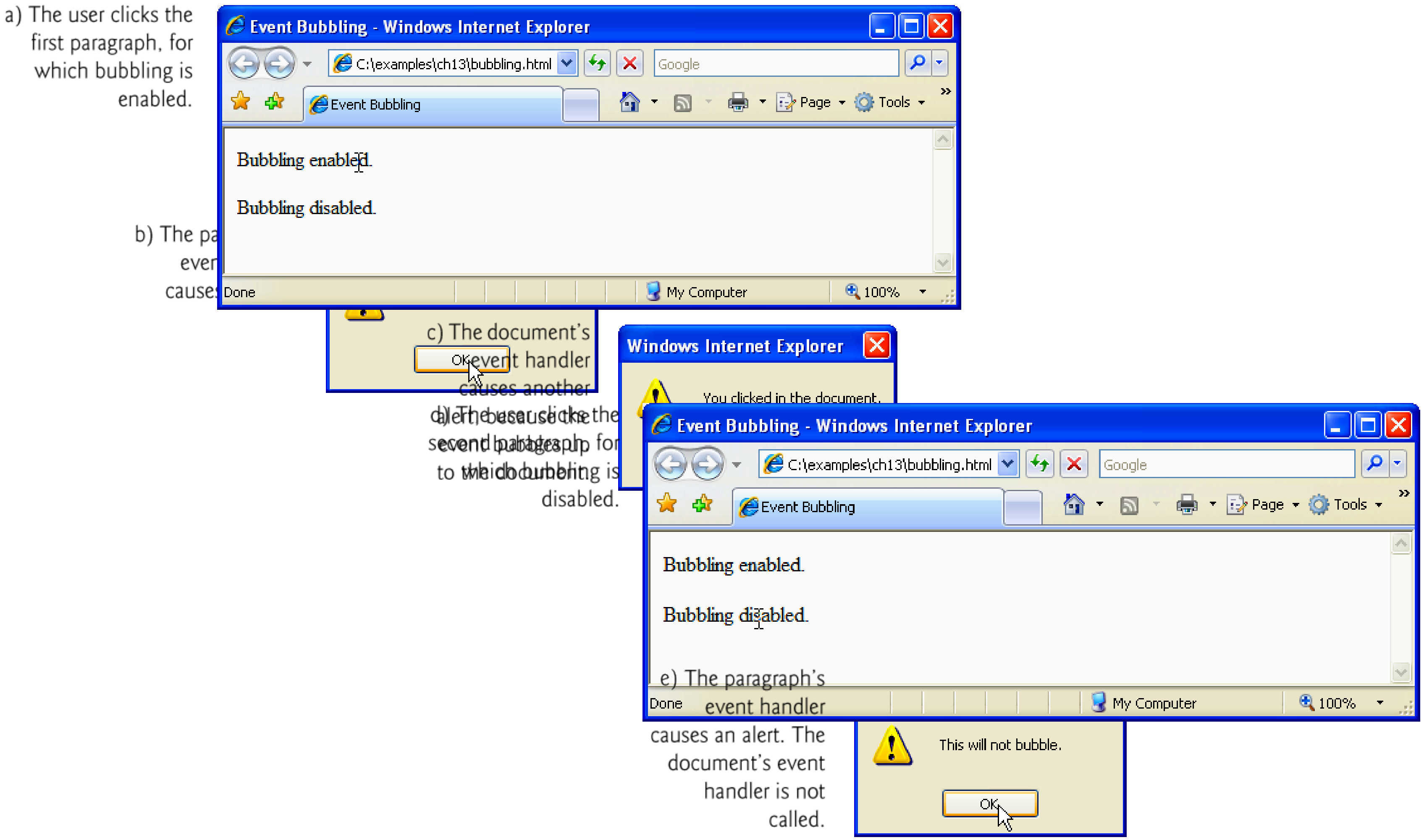
```
           alert(         "This will not bubble."          );
                  e.cancelBubble =      true   ;
        }         // end function noBubble

                function     registerEvents()
        {
                  document.onclick = documentClick;
                  document.getElementById(       "bubble"     ).onclick = bubble;
                  document.getElementById(       "noBubble"     ).onclick = noBubble;
        }         // end function registerEvents
        //        -- >
      </script>
    </head>
     <body onload =      "registerEvents()"          >
         <p id =     "bubble"     >Bubbling enabled.       </p>
         <p id =      "noBubble"    >Bubbling disabled.        </p>
    </body>
  </html>
```

Cancels event bubbling

Registers an event for the `document` object

Registers events for clicking in the two `p` elements, which are children of the `document` object

Example



a) The user clicks the first paragraph, for which bubbling is enabled.

b) The pa
even
cause

c) The document's event handler causes another alert because the event bubbles up to the document.

d) The user clicks the second paragraph, for which bubbling is disabled.

e) The paragraph's event handler causes an alert. The document's event handler is not called.

Event Bubbling - Windows Internet Explorer

C:\examples\ch13\bubbling.html

Bubbling enabled.

Bubbling disabled.

Windows Internet Explorer

You clicked in the document.

This will not bubble.

Common Programming Error

Forgetting to cancel event bubbling when necessary may cause unexpected results in your scripts.

Cross Browser Events

Event	Description
onabort	Fires when image transfer has been interrupted by user.
onchange	Fires when a new choice is made in a select element, or when a text input is changed and the element loses focus.
onclick	Fires when the user clicks using the mouse.
ondblclick	Fires when the mouse is double clicked.
onfocus	Fires when a form element gains focus.
onkeydown	Fires when the user pushes down a key.
onkeypress	Fires when the user presses then releases a key.
onkeyup	Fires when the user releases a key.
onload	Fires when an element and all its children have loaded.
onsubmit	Fires when a form is submitted.
onunload	Fires when a page is about to unload.

Cross Browser Events

Event	Description
onmousedown	Fires when a mouse button is pressed down.
onmousemove	Fires when the mouse moves.
onmouseout	Fires when the mouse leaves an element.
onmouseover	Fires when the mouse enters an element.
onmouseup	Fires when a mouse button is released.
onreset	Fires when a form resets (i.e., the user clicks a reset button).
onresize	Fires when the size of an object changes (i.e., the user resizes a window or frame).
onse lect	Fires when a text selection begins (applies to input or textarea).
onsubmit	Fires when a form is submitted.
onunload	Fires when a page is about to unload.

Now, you can see:

04 Cross Browser Event Handling
05 Event Delegation

http://youtu.be/nzv4PWkWBRw
http://youtu.be/sF47i1v_EYQ

AJAX

1. What’s AJAX?
2. Why AJAX?
3. Look at some AJAX examples
4. AJAX for Libraries
5. Walkthrough sample AJAX application

Now, you can see:

06 What is AJAX

http://youtu.be/hBi5CNa-F-o

What is AJAX?

- Asynchronous Javascript and XML
  - Not all AJAX apps involve XML
- Combination of technologies
  - XHTML, CSS, DOM
  - XML, XSLT, XMLHttp, JavaScript
  - Some server scripting language
- A method for building more responsive and interactive applications

History

Internet Explorer introduces the concept of IFrame element in 1996.(a technique that helps in loading the contents of a web page.)

In the year 1998, Microsoft introduces another technique, called ‘Microsoft’s Remote Scripting’ as a replacement to the older techniques.

History

A year later, in 1999, Microsoft introduces the XMLHttpRequest object, an ActiveX control, in IE 5.

The term AJAX is coined on February 18, 2005, by **Jesse James Garret** in a short essay published a few days after Google released its Maps application.

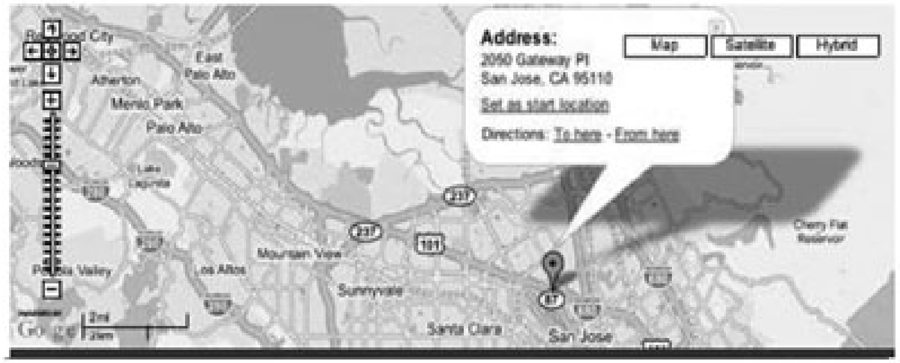


Figure 1.16 Google Maps

History

Finally, in the year 2006, the W3C (World Wide Web Consortium) announces the release of the first draft which includes the specification for the object (XHR) and makes it an official web standard.

AJAX Components

XHTML and CSS

Ajax applies these familiar Web standards for styling the look and feel of a page and to markup those areas on a page that will be targeted for data updates.

DOM (document object model)

Ajax uses the DOM to manipulate dynamic page views for data and to walkthrough documents to “cherrypick” data. The DOM enables certain pieces of an Ajax page to be transformed and updated with data.

XML, JSON (Javascript Object Notation), HTML, or plain text

Ajax can use any of these standards to provide structure to the data it passes to and from a page.

AJAX Components

XMLHttpRequest (XHR) object
The heavy lifter for Ajax: It's a javascript object embedded in most modern browsers that sets up data request/response pipelines between client and server.

Javascript
Lightweight programming language that Ajax uses for instructions to bind all of the components together.

Why AJAX?

Want to make your applications more interactive

Want to incorporate data from external Web Services

Don’t want your users to have to download a plugin

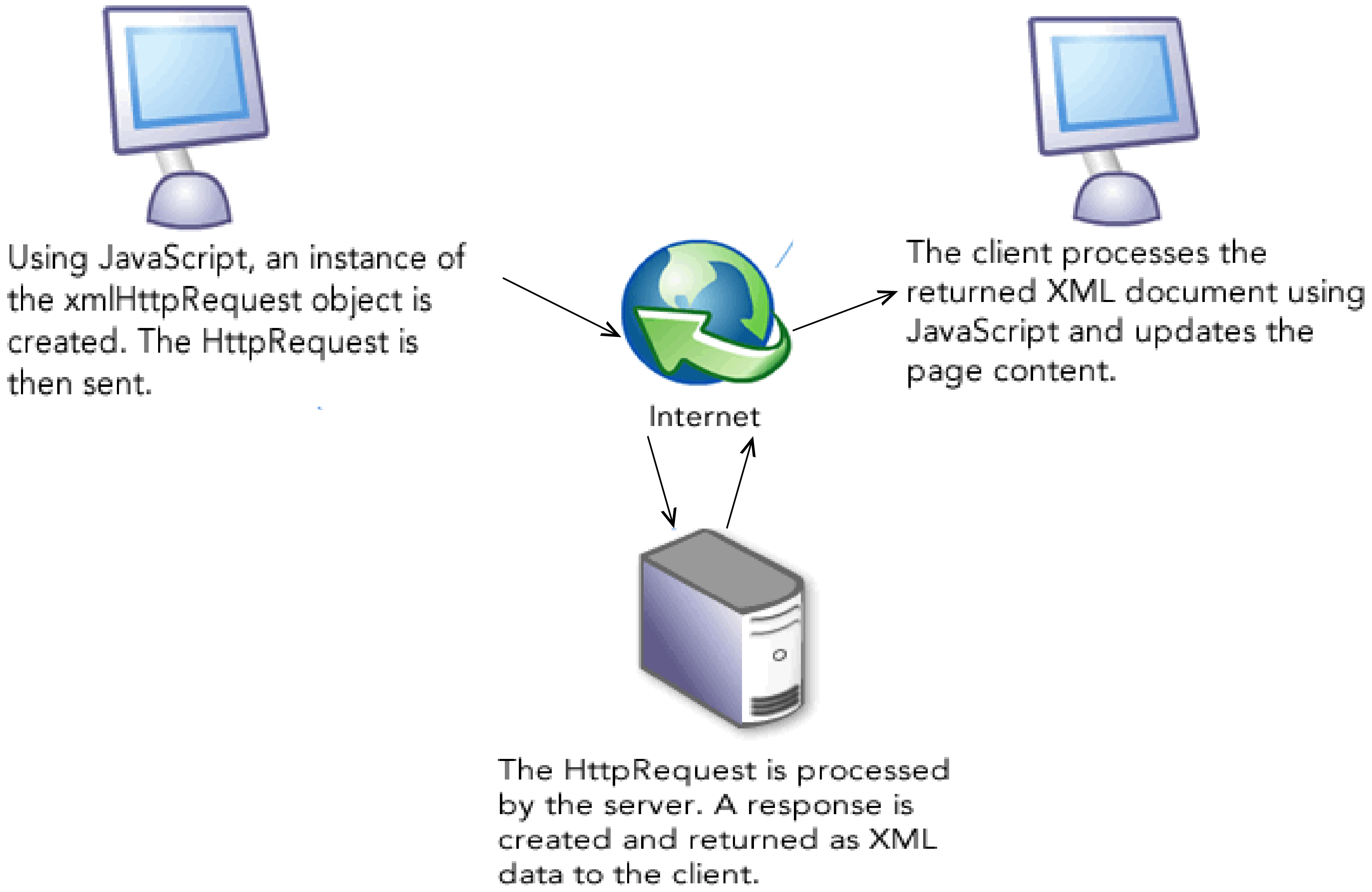
Client vs. Server Scripting

Client scripting
Web browser does all the work

Server Scripting
Web server does all the work

AJAX leverages both client and server side scripting

How AJAX Works

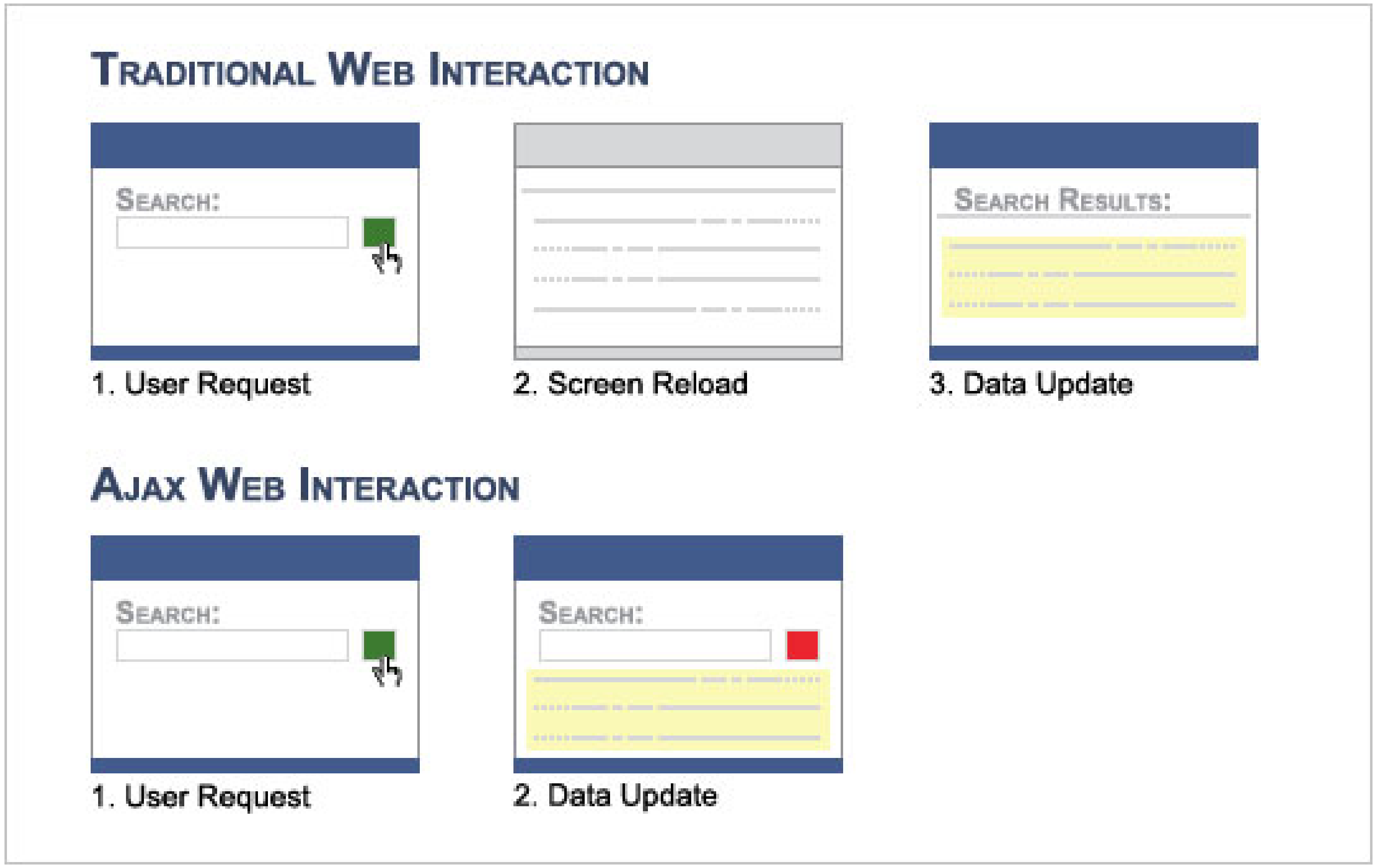


Using JavaScript, an instance of the xmlHttpRequest object is created. The HttpRequest is then sent.

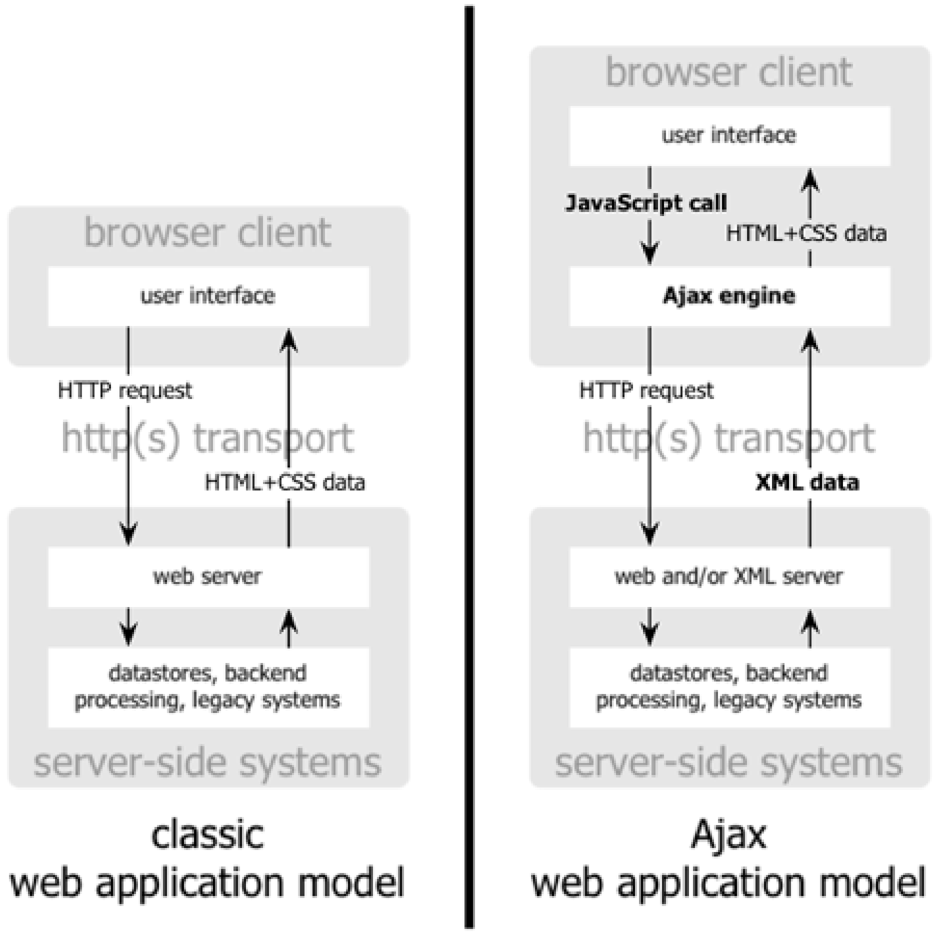
The client processes the returned XML document using JavaScript and updates the page content.

The HttpRequest is processed by the server. A response is created and returned as XML data to the client.

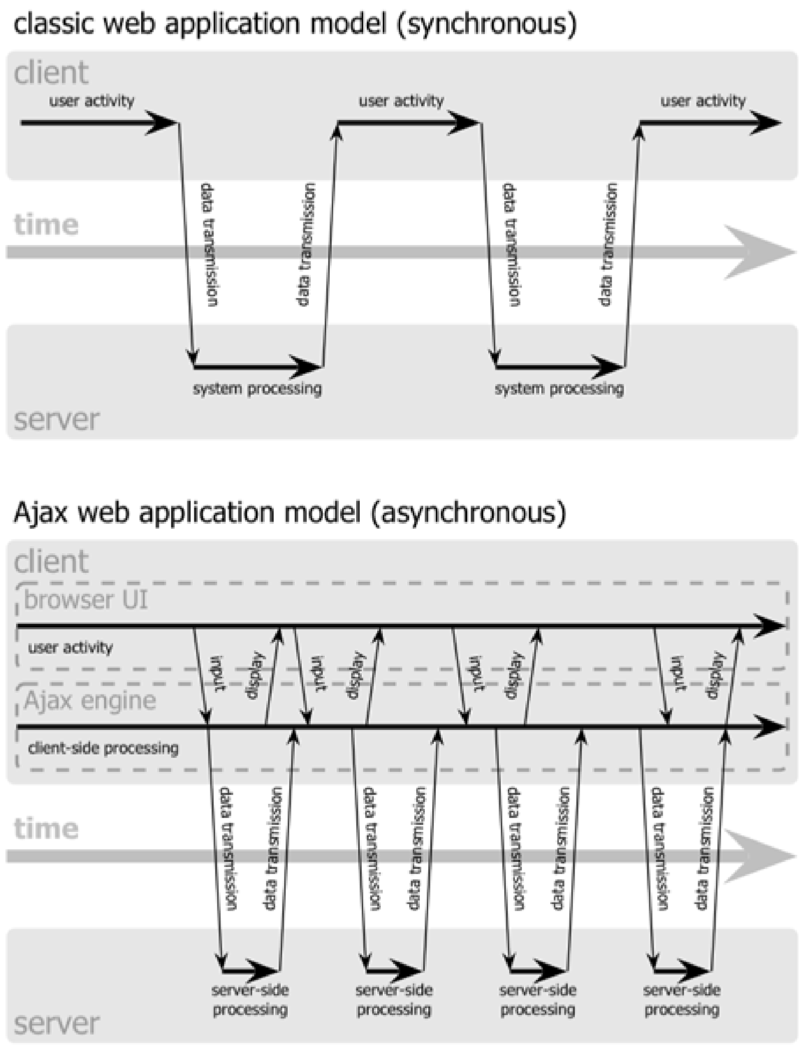
How AJAX Works



How AJAX Works



Synchronous vs. Asynchronous



AJAX Web Interaction

What you don’t see

Data reload happens in the background

JavaScript queries the server to get the proper data without you knowing it

Page updates without a screen “reload”

Potential Problems

Javascript MUST be enabled
Back button doesn’t always work
Pages can be difficult to bookmark
Search engines may not be able to index all portions of an AJAX site
Cross browser differences in how XML is dealt with

Some AJAX examples

Google Calendar
Flickr
Backpack

Basic AJAX Components

Server-side Component

Communicates with the database, or web service
Can be written in any server-side language (PHP, ASP, Coldfusion, etc)

Client-side Component

Written in Javascript, often uses XMLHttp
Accesses the server side page in the background

Hidden Frame Method

Communication with server takes place in a frame that user can't see

Back and Forward buttons still work

If something goes wrong user receives no notification

XMLHttp Method

Code is cleaner and easier to read

Able to determine if there is a failure

No browser history, Back and Forward buttons break

XMLHttpRequest

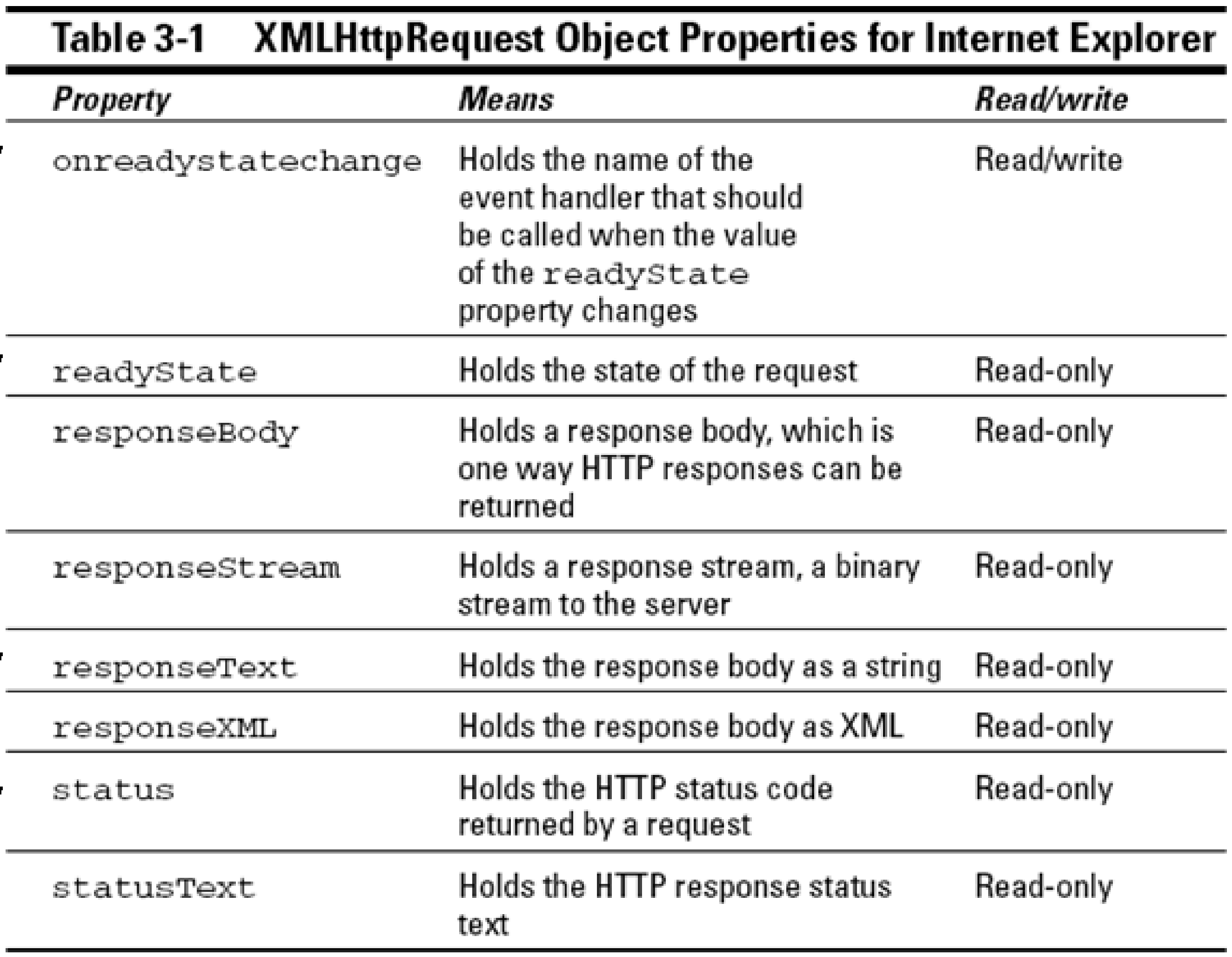


Table 3-1 XMLHttpRequest Object Properties for Internet Explorer

Property	*Means*	*Read/write*
`onreadystatechange`	Holds the name of the event handler that should be called when the value of the `readyState` property changes	Read/write
`readyState`	Holds the state of the request	Read-only
`responseBody`	Holds a response body, which is one way HTTP responses can be returned	Read-only
`responseStream`	Holds a response stream, a binary stream to the server	Read-only
`responseText`	Holds the response body as a string	Read-only
`responseXML`	Holds the response body as XML	Read-only
`status`	Holds the HTTP status code returned by a request	Read-only
`statusText`	Holds the HTTP response status text	Read-only

XMLHttpRequest

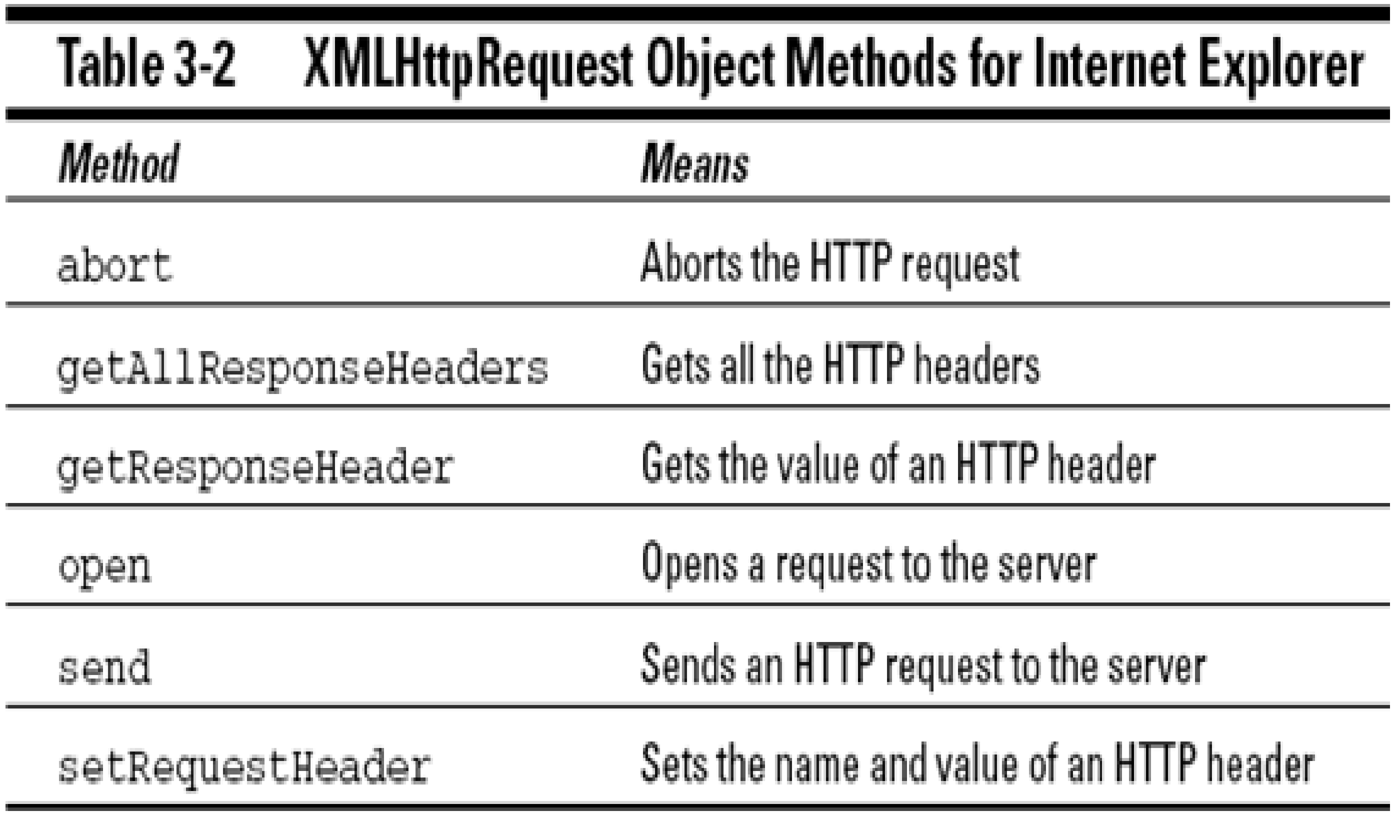


Table 3-2 XMLHttpRequest Object Methods for Internet Explorer

Method	*Means*
`abort`	Aborts the HTTP request
`getAllResponseHeaders`	Gets all the HTTP headers
`getResponseHeader`	Gets the value of an HTTP header
`open`	Opens a request to the server
`send`	Sends an HTTP request to the server
`setRequestHeader`	Sets the name and value of an HTTP header

The readyState values

State	Description
0	uninitialized
1	loading
2	loaded
3	interactive
4	complete

A few status values

Status	Description
200	OK
400	Bad Request
404	File Not Found
500	Internal Server Error
505	HTTP version not supported

Now, you can see:

07 Using a synchronous XHR request

http://youtu.be/t9MIWwVzRfg

Potential Uses for AJAX

Error checking in Forms
AutoSuggest
Drag and Drop objects functionality
Dynamically move view around on image or map
Preload content you want to show later
Apply limits to search results and get new results quickly

Now, you can see:

08 Making requests asynchronous

http://youtu.be/8Yo0X6dQ_jk

AJAX for Libraries

Browsing subject headings
“Pre-displaying” indexes and databases categories
Complex ILL or contact forms
Federated Search
OPAC and digital library interfaces

Now, you can see:

09 Scripting for backwards compatibility

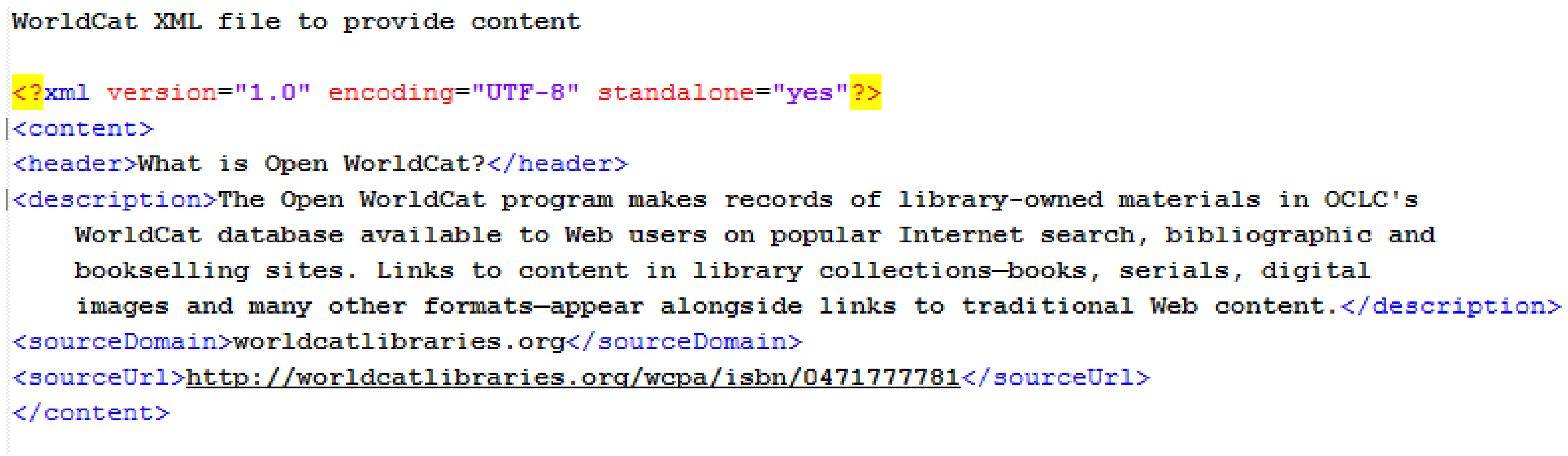
http://youtu.be/2lbU3GMQYJg

AJAX – Sample Applications

PageInsert - WorldCat Form

BrowseSearch - LOC Subject Headings

Code Sample #1: WorldCat Form



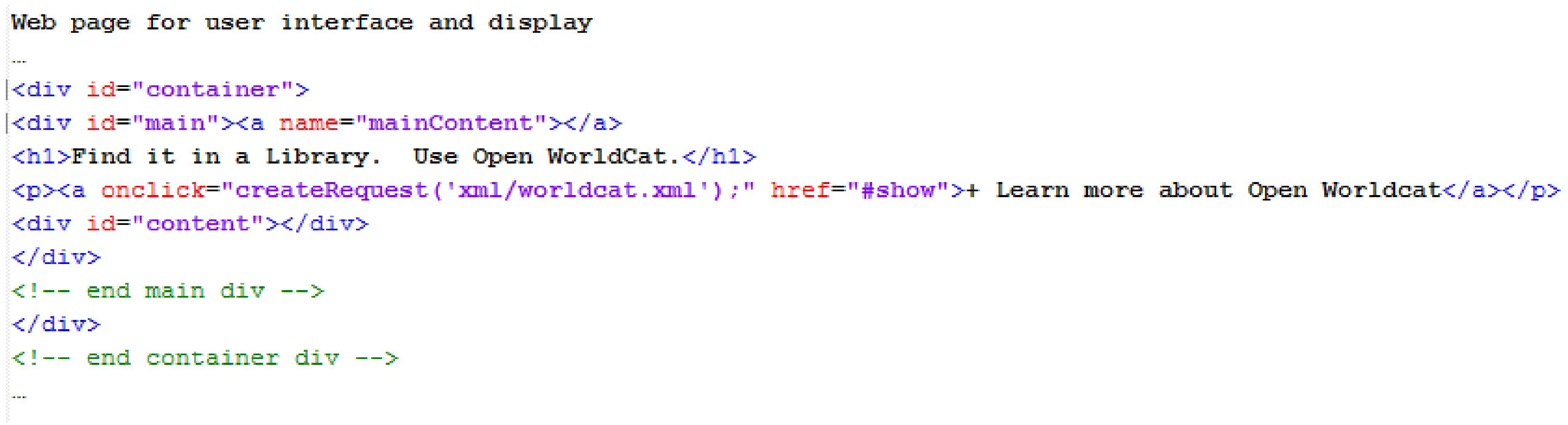
WorldCat XML file to provide content

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<content>
<header>What is Open WorldCat?</header>
<description>The Open WorldCat program makes records of library-owned materials in OCLC's
    WorldCat database available to Web users on popular Internet search, bibliographic and
    bookselling sites. Links to content in library collections—books, serials, digital
    images and many other formats—appear alongside links to traditional Web content.</description>
<sourceDomain>worldcatlibraries.org</sourceDomain>
<sourceUrl>http://worldcatlibraries.org/wcpa/isbn/0471777781</sourceUrl>
</content>
```

Code Sample #1: Explanation

Our source file
Various and sundry factoids about WorldCat, some associated urls
header and description element to populate the heading and description of the content
sourceDomain will give an action value to our WorldCat search form
sourceUrl element will provide a link to an Open Worldcat record

Code Sample #2: WorldCat Form



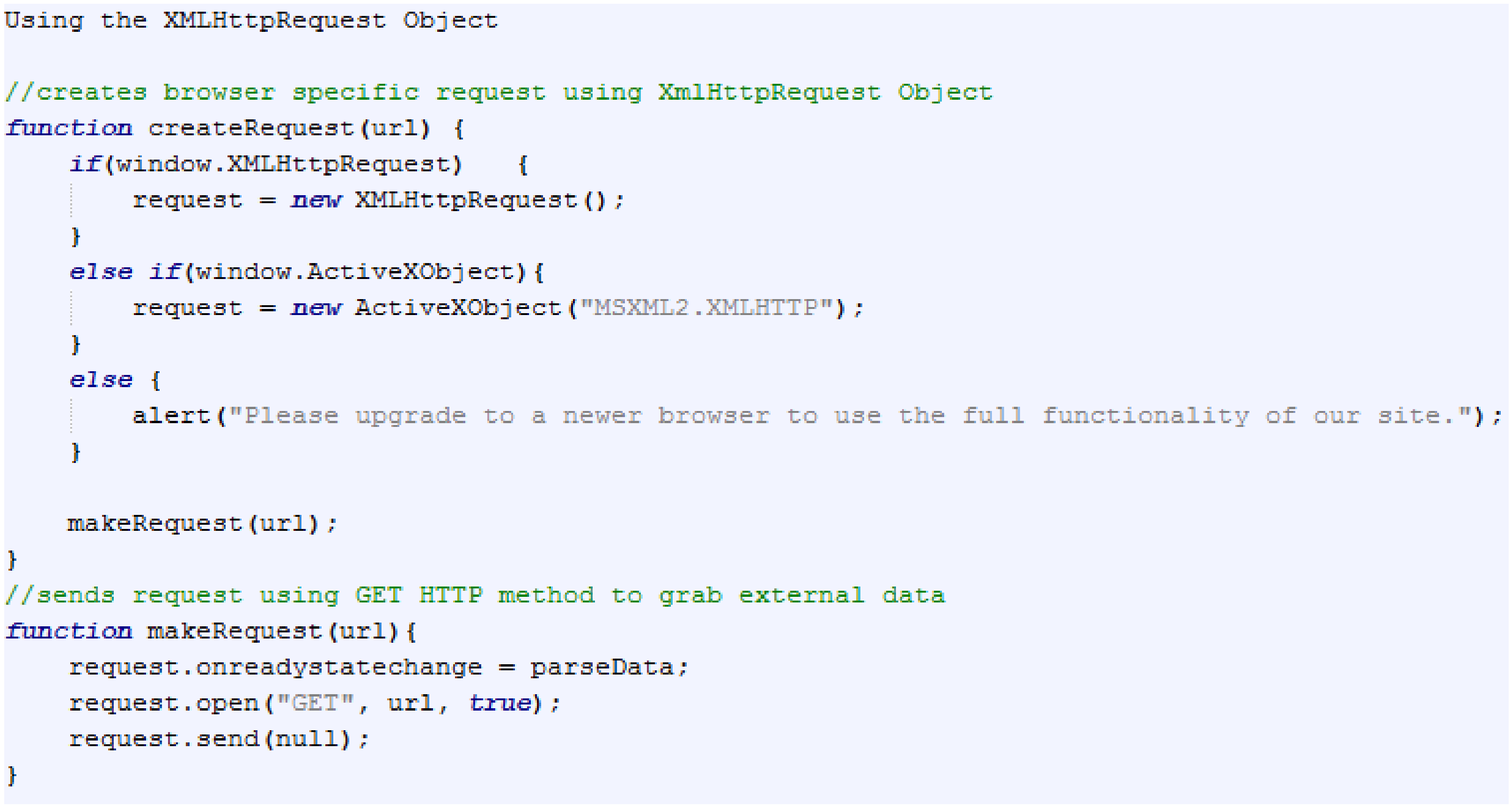
Web page for user interface and display

```
…
<div id="container">
<div id="main"><a name="mainContent"></a>
<h1>Find it in a Library.  Use Open WorldCat.</h1>
<p><a onclick="createRequest('xml/worldcat.xml');" href="#show">+ Learn more about Open Worldcat</a></p>
<div id="content"></div>
</div>
<!-- end main div -->
</div>
<!-- end container div -->
…
```

Code Sample #2: Explanation

XHTML form that gives action to our script
Notice the javascript “onclick” event handler on <p> tag
<div id=“content”> will be populated with script messages OR new html tags received via our Ajax requests

Code Sample #3: WorldCat Form



```
Using the XMLHttpRequest Object

//creates browser specific request using XmlHttpRequest Object
function createRequest(url) {
    if(window.XMLHttpRequest)   {
        request = new XMLHttpRequest();
    }
    else if(window.ActiveXObject){
        request = new ActiveXObject("MSXML2.XMLHTTP");
    }
    else {
        alert("Please upgrade to a newer browser to use the full functionality of our site.");
    }

    makeRequest(url);
}
//sends request using GET HTTP method to grab external data
function makeRequest(url){
    request.onreadystatechange = parseData;
    request.open("GET", url, true);
    request.send(null);
}
```

Globant
we are ready

Code Sample #3: Explanation

First part of our javascript
Creates the XMLHttpRequest
Using the if and else statements to check for Web browsers’ different implementations of XMLHttpRequest
Ends with makeRequest function

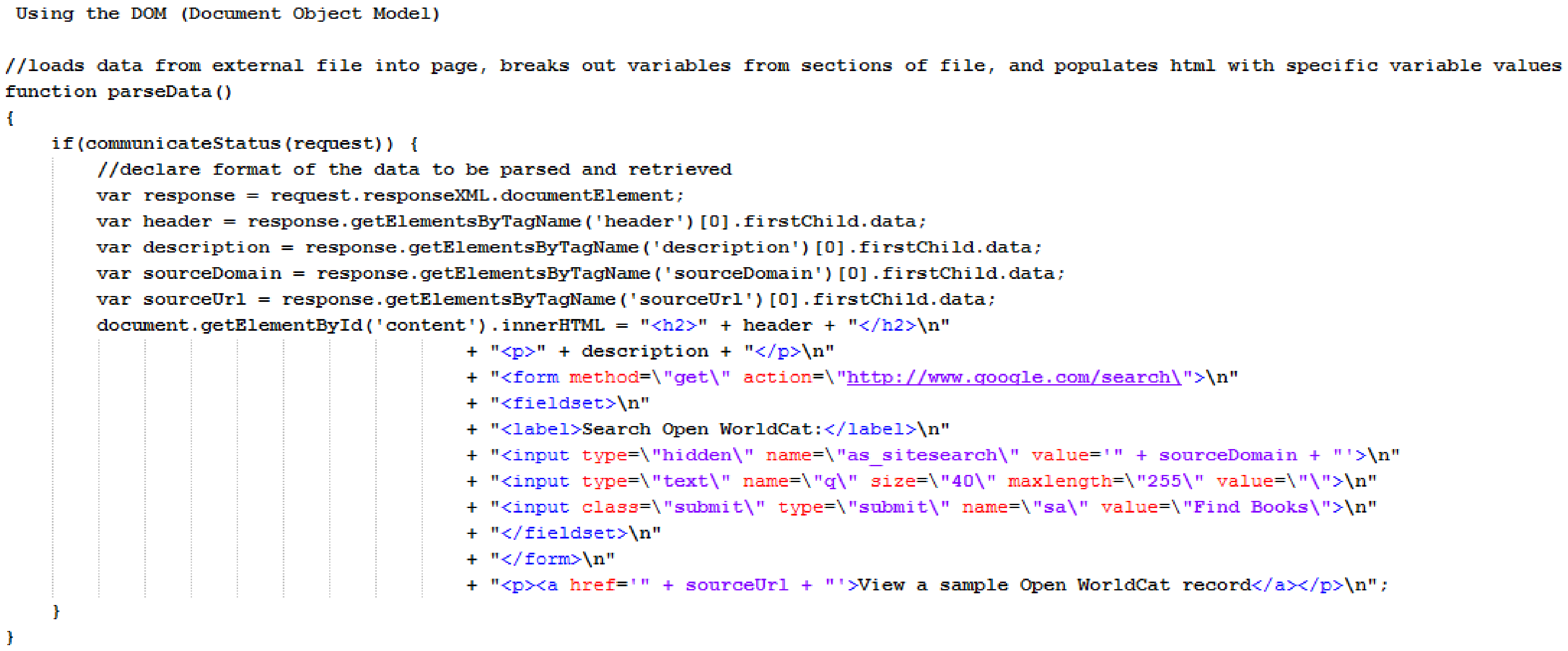
Code Sample #4: WorldCat Form

```
Communicating the status of our request

//checks state of HTTP request and gives brief status note to user
function communicateStatus(obj)
{
    if(obj.readyState == 0) { document.getElementById('content').innerHTML = "Sending Request..."; }
    if(obj.readyState == 1) { document.getElementById('content').innerHTML = "Loading Response..."; }
    if(obj.readyState == 2) { document.getElementById('content').innerHTML = "Response Loaded..."; }
    if(obj.readyState == 3) { document.getElementById('content').innerHTML = "Response Ready..."; }
    if(obj.readyState == 4) {
        if(obj.status == 200){
           return true;
        }
        else if(obj.status == 404){
           // Add a custom message or redirect the user to another page
           document.getElementById('content').innerHTML = "File not found";
        }
        else {
           document.getElementById('content').innerHTML = "There was a problem retrieving the XML.";
        }
    }
}
```

Code Sample #4: Explanation

Next part of our javascript
Displays different messages to the user based on the status of the request on the server
uses the "obj" variable which was created earlier when we called the XMLHttpRequest
First peek at Document Object Model (DOM) in action

Code Sample #5: WorldCat Form



Using the DOM (Document Object Model)

```
//loads data from external file into page, breaks out variables from sections of file, and populates html with specific variable values
function parseData()
{
    if(communicateStatus(request)) {
        //declare format of the data to be parsed and retrieved
        var response = request.responseXML.documentElement;
        var header = response.getElementsByTagName('header')[0].firstChild.data;
        var description = response.getElementsByTagName('description')[0].firstChild.data;
        var sourceDomain = response.getElementsByTagName('sourceDomain')[0].firstChild.data;
        var sourceUrl = response.getElementsByTagName('sourceUrl')[0].firstChild.data;
        document.getElementById('content').innerHTML = "<h2>" + header + "</h2>\n"
                                + "<p>" + description + "</p>\n"
                                + "<form method=\"get\" action=\"http://www.google.com/search\">\n"
                                + "<fieldset>\n"
                                + "<label>Search Open WorldCat:</label>\n"
                                + "<input type=\"hidden\" name=\"as_sitesearch\" value='" + sourceDomain + "'>\n"
                                + "<input type=\"text\" name=\"q\" size=\"40\" maxlength=\"255\" value=\"\">\n"
                                + "<input class=\"submit\" type=\"submit\" name=\"sa\" value=\"Find Books\">\n"
                                + "</fieldset>\n"
                                + "</form>\n"
                                + "<p><a href='" + sourceUrl + "'>View a sample Open WorldCat record</a></p>\n";
    }
}
```

Code Sample #5: Explanation

Last part of our javascript
Applies DOM to give us a standard means of modeling the structure of XHTML or XML documents
DOM functions like “getElementsByTagName”
Grab data and push it into prescribed sections of our XHTML page

Code Sample #6: WorldCat Form

```
CSS (Cascading Style Sheets)

...
/* =container
--------------------------------------------- */
div#container {width:65em;margin:0 auto;background:#fff;}

/* =main
--------------------------------------------- */
div#main {width:63em;margin:0 auto;padding:1em .5em 2em .5em;}

/* =content
--------------------------------------------- */
div#content {width:95%;margin:0 auto;}
#content p.warn {color:red;}

/* =forms
--------------------------------------------- */
form {padding:10px;border-top:1px solid #ccc;border-right:2px solid #ccc;border-bottom:2px solid #ccc;
        border-left:1px solid #ccc;background-color:#F2F2F2;}
fieldset {border:none;}
label {font-size:1.2em;color:#2b4268;vertical-align:middle;cursor:pointer;}
input, select, textarea {width:25em;font:1.0em verdana,arial,sans-serif;padding:3px;margin:3px;
        border:1px solid gray;border-color:#AAA #DDD #DDD #AAA;vertical-align:middle;}
input:focus {border:1px #000 solid;}
input.submit {width:10em;font-size:.90em;color:#2b4268;}
...
```

Globant
we are ready

Code Sample #6: Explanation

Part of our CSS file
Means of passing style rules for different pieces of the Web page
<div> tags are given specific, relative widths, <form> tags are styled with attractive borders

Final thoughts – What’s Next?

That’s AJAX and an AJAX application in a nutshell.
Consider AJAX advantages and disadvantages
Fundamentals of method are there
Keep practicing and learning

Code Sample #1: LOC Subject Headings

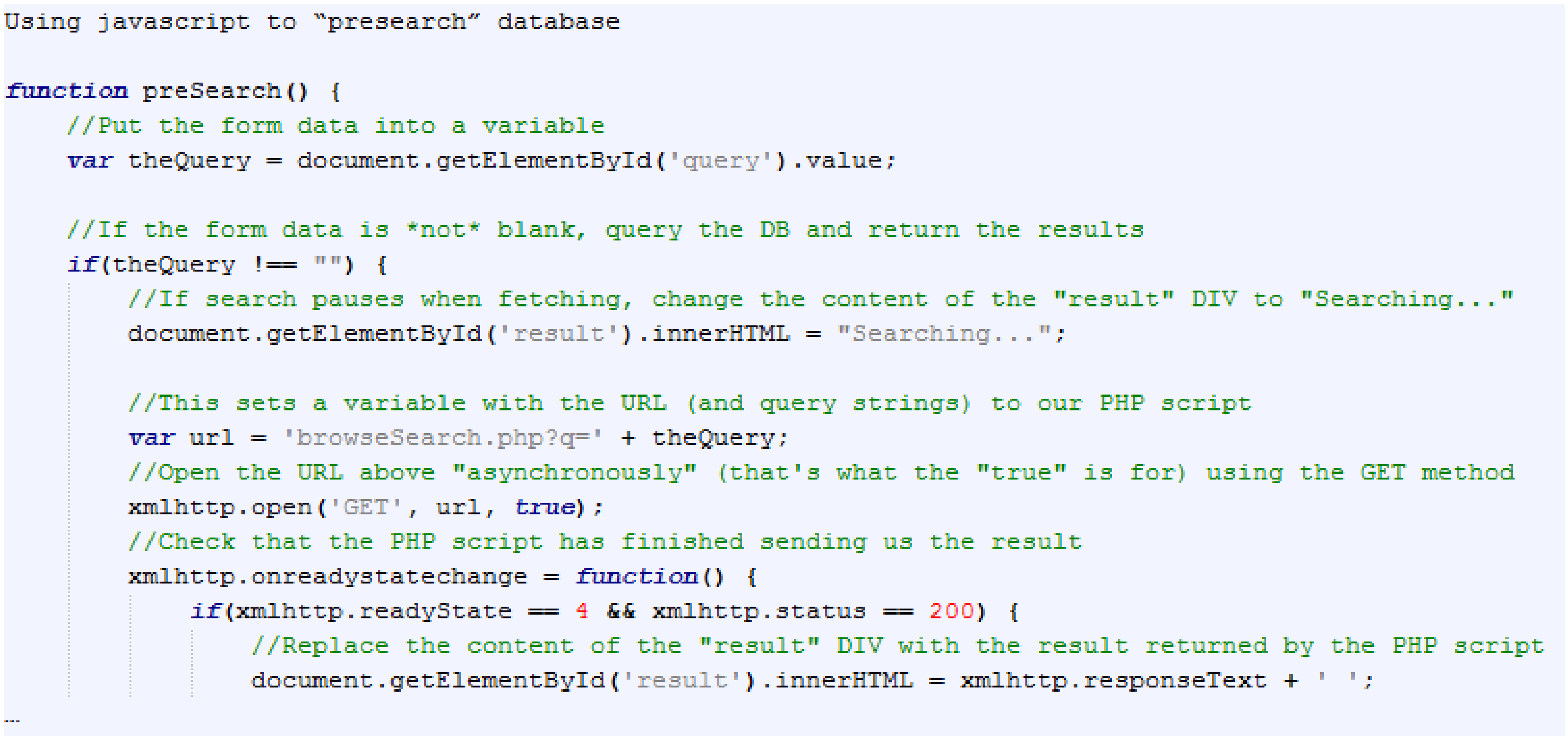
Web page for user interface and display

```
<div id="main"><a name="mainContent"></a>
<h2 class="mainHeading">CIL 2006 :: Example: Library of Congress BrowseSearch</h2>
<form id="searchbox" action="browseSearch.php" method="post">
  <p>
    <label for="query"><strong>BrowseSearch:</strong></label>&nbsp;
    <input type="text" name="query" autocomplete="off" id="query" onKeyUp="preSearch()" />
    &nbsp;<img id="searchStatus" alt="searching..." src="./meta/img/spinner.gif" />
  </p>
</form>
<div id="result">&nbsp;</div>
</div>
```

Code Sample #1: Explanation

XHTML form that gives action to our script
Note the javascript “onKeyUp” event handler on <input> tag
<input> also given “name” and “id”
<div id=“result”> will be populated with script messages OR new html tags received via our Ajax requests

Code Sample #2: LOC Subject Headings



Using javascript to “presearch” database

```
function preSearch() {
    //Put the form data into a variable
    var theQuery = document.getElementById('query').value;

    //If the form data is *not* blank, query the DB and return the results
    if(theQuery !== "") {
        //If search pauses when fetching, change the content of the "result" DIV to "Searching..."
        document.getElementById('result').innerHTML = "Searching...";

        //This sets a variable with the URL (and query strings) to our PHP script
        var url = 'browseSearch.php?q=' + theQuery;
        //Open the URL above "asynchronously" (that's what the "true" is for) using the GET method
        xmlhttp.open('GET', url, true);
        //Check that the PHP script has finished sending us the result
        xmlhttp.onreadystatechange = function() {
            if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {
                //Replace the content of the "result" DIV with the result returned by the PHP script
                document.getElementById('result').innerHTML = xmlhttp.responseText + ' ';
...
```

Code Sample #2: Explanation

Piece of javascript that creates instant search

Talks to server-side PHP script - browseSearch.php

Uses DOM to populate <div id="result"> with search results

Code Sample #3: LOC Subject Headings

PHP search loc database script

```
<?php
//declare variables to be used in query and display
$keywords = $_GET['query'];
$link = "<p><a href=\"browseSearch.php\">Library of Congress LiveSearch</a></p>";
...
// bring database parameters and functions onto page
...
//form sql statement
$query = "SELECT subject_id, label, callno FROM subject WHERE label LIKE '%$keywords%' ORDER BY callno ASC";
//store sql result as an array
$result = mysql_query($query) or die('<p class=\"warn\">Error retrieving subjects from loc database!<br />'.
                'Error: ' . mysql_error() . '</p>');
//create message if no rows match search terms
…
//format sql result for display
    while($record = mysql_fetch_object($result))
        {
        echo '<dl><dt><strong>'.stripslashes($record->label).'</strong></dt>';
        echo '<dd>Call Number Range: '.stripslashes($record->callno).'</dd>';
        echo '<dd><a href="http://www.lib.montana.edu/help/locationguide.html">
                    Find Call Number on Library Floor Map</a></dd></dl>';
        echo '<hr size="1" />';
        }
        echo $link;
?>
```

Code Sample #3: Explanation

Piece of PHP script that searches loc database

Basic SQL SELECT statement

Uses <dl> to format search results

Now, you can see:

10 Modifying elements with getElementsByTagName
11 Updating the DOM with getElementById

http://youtu.be/IFkv2fhZlto

http://youtu.be/Rqlo78JW6PE

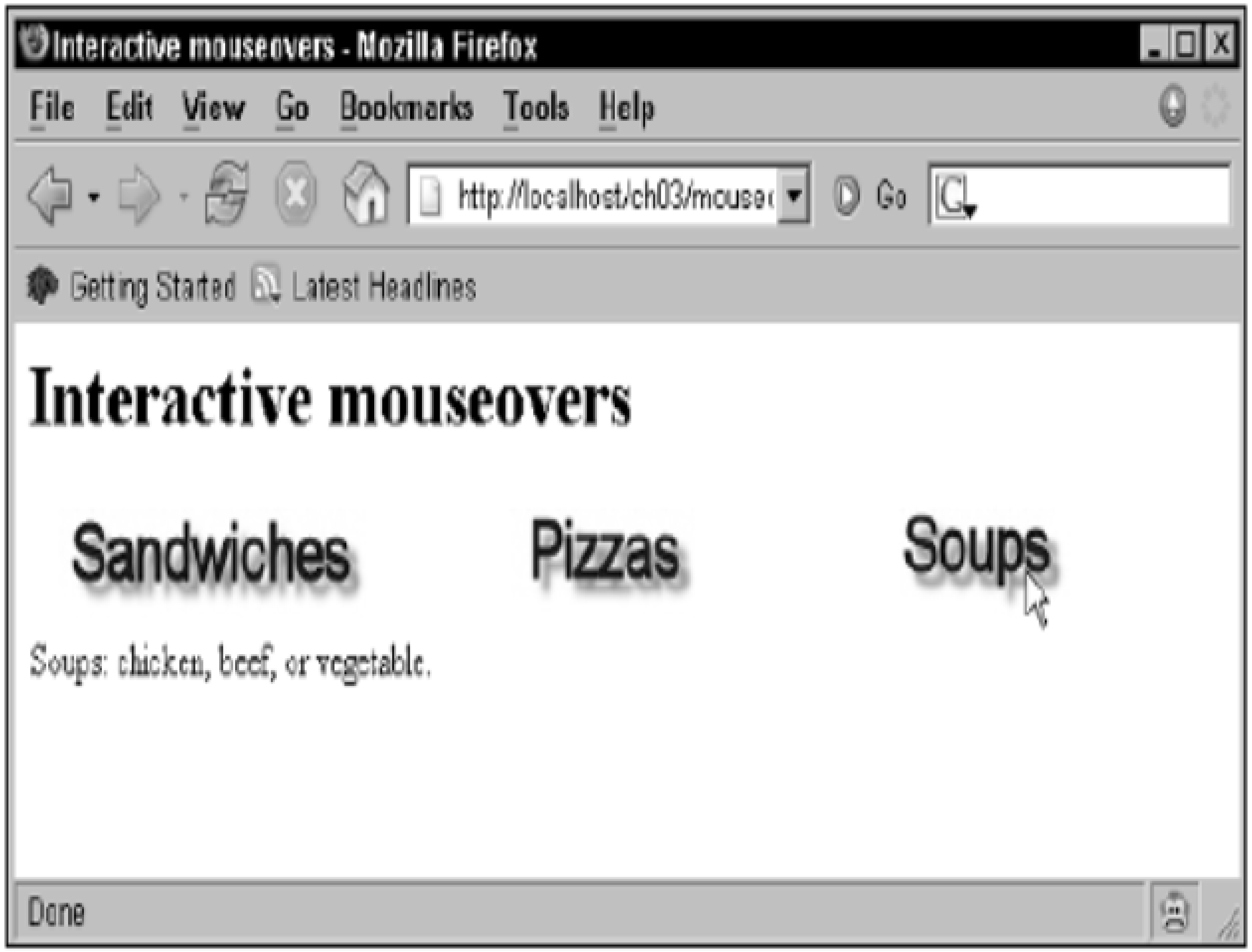
Example Interactive mouse-overs

Here comes another Ajax example — one that's a little more impressive visually.

When you move the mouse over one of the images on this page, the application fetches text for that mouseover by using Ajax.

All you really have to do is to connect the getData function (which fetches text data and displays it in the <div> element whose name you pass) to the 'onmouseover' event of each of the images.

Example Interactive mouse-overs

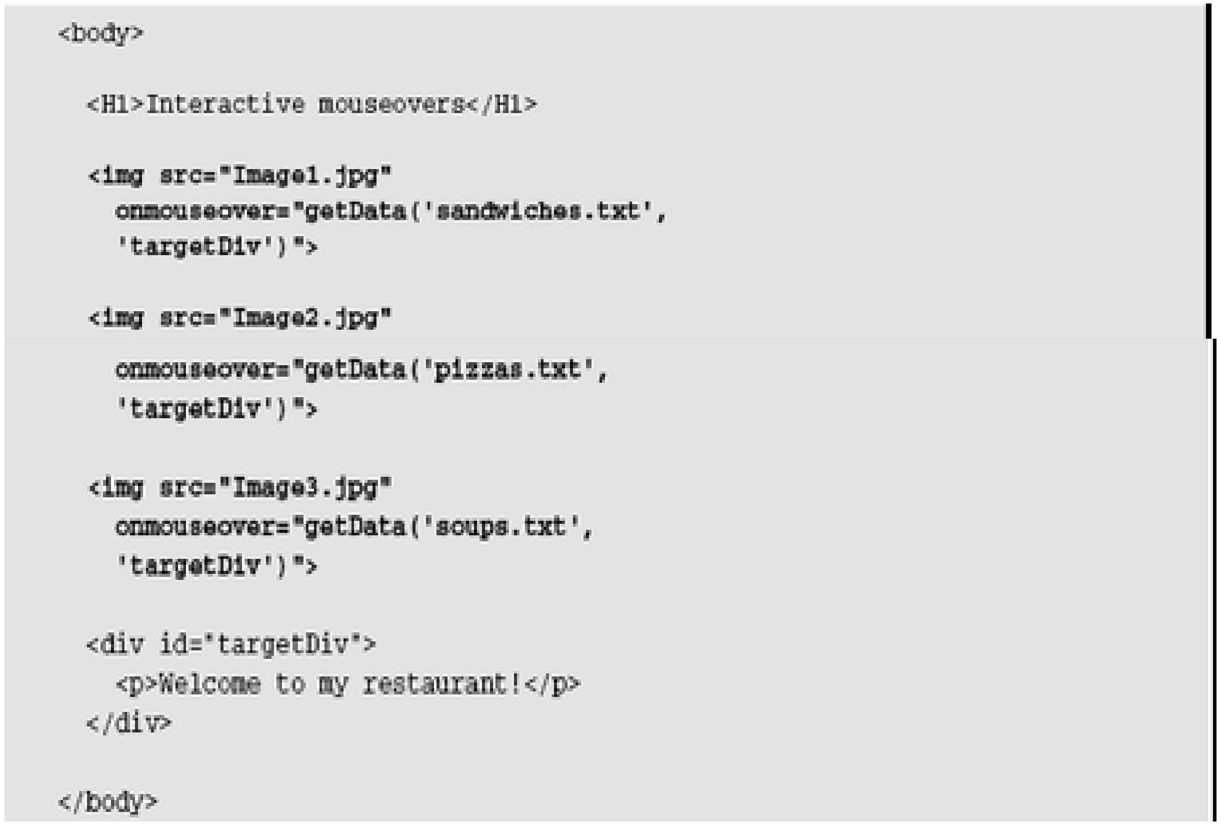

Interactive mouseovers - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/ch03/mouse
Go
Getting Started
Latest Headlines

Interactive mouseovers

Sandwiches Pizzas Soups

Soups: chicken, beef, or vegetable.

Example Interactive mouse-overs

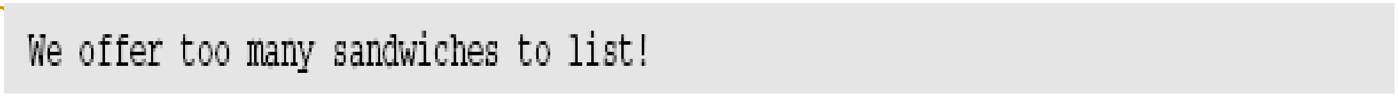


```
<body>

  <H1>Interactive mouseovers</H1>

  <img src="Image1.jpg"
    onmouseover="getData('sandwiches.txt',
    'targetDiv')">

  <img src="Image2.jpg"
    onmouseover="getData('pizzas.txt',
    'targetDiv')">

  <img src="Image3.jpg"
    onmouseover="getData('soups.txt',
    'targetDiv')">

  <div id='targetDiv'>
    <p>Welcome to my restaurant!</p>
  </div>

</body>
```

Example Interactive mouse-overs

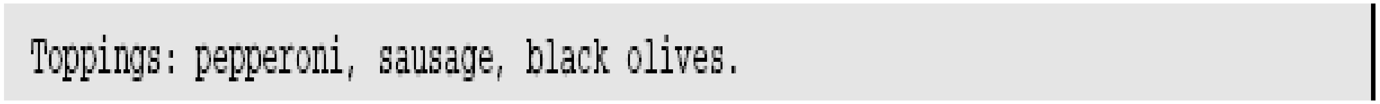
Here’s the content of sandwiches.txt :

ar

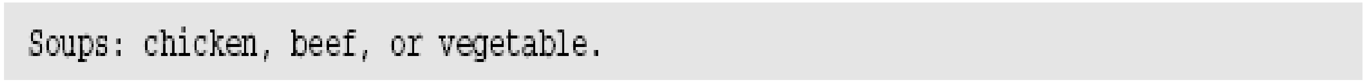


```
We offer too many sandwiches to list!
```

and soups.txt :



```
Toppings: pepperoni, sausage, black olives.
```



```
Soups: chicken, beef, or vegetable.
```

Now, you can see:

12 Using event driven AJAX

http://youtu.be/_Co933RVMjs

Hm…done with coding ☺

AJAX

Callback

Promise

Event Emitter

Publish/Suscribe

CALLBACKS

CALLBACKS

Callbacks In Real Life
Karlsruher
Versicherungen
Ralf Siebert

CALLBACKS

What we've seen so far has been doing asynchronicity through *callbacks*.

Callbacks are OK for simple operations, but force us into *continuation passing style*.

CALLBACKS EXAMPLE

```
var customer = {
    placeOrder: function() {
        restaurant.takeOrder('burger', this.onFoodReady);
    },
    onFoodReady: function(food) { … }
};

var restaurant = {
    takeOrder: function(order, foodReadyCallback) {
        // call foodReadyCallback(food) when food is ready
    }
};
```

Recurring StackOverflow question:

```
function getY() {
  var y;
  $.get("/gety", function (jsonData) {
    y = jsonData.y;
  });
  return y;
}

var x = 5;
var y = getY();

console.log(x + y);
```

Why doesn't it work???

CALLBACKS

After getting our data, we

have to do everything else in a

continuation:

CALLBACKS

```
function getY(continueWith) {
  $.get("/gety", function (jsonData) {
    continueWith(jsonData.y);
  });
}

var x = 5;
getY(function (y) {
  console.log(x + y);
});
```

CALLBACKS

Used to notify of completion of an **asynchronous task**

Simple

Efficient

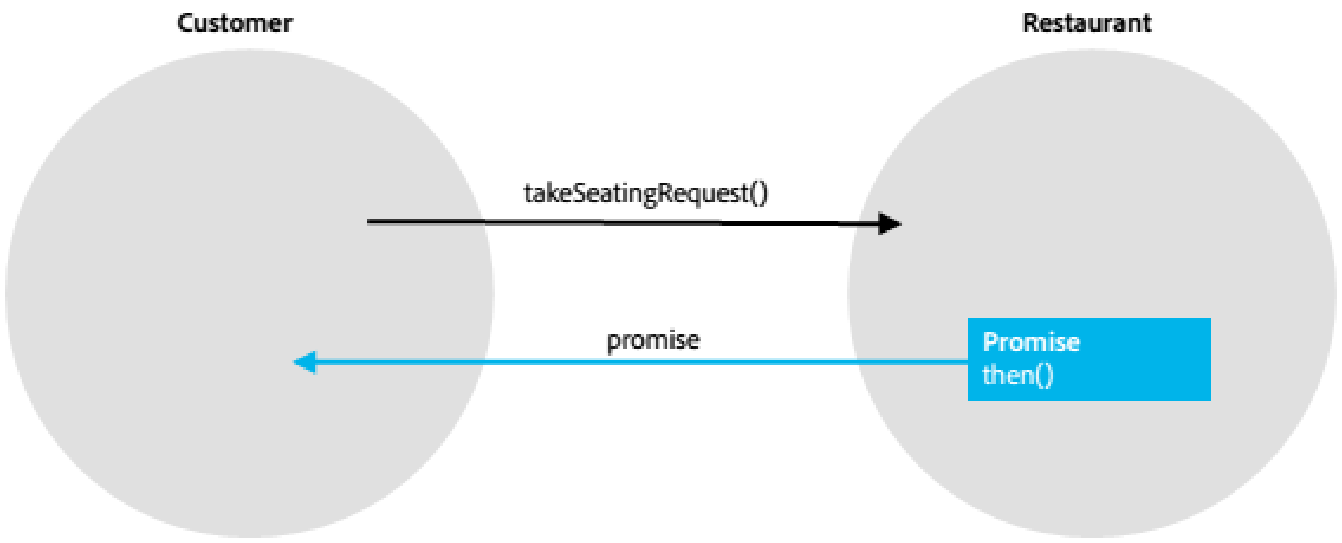
No libraries required

PROMISE

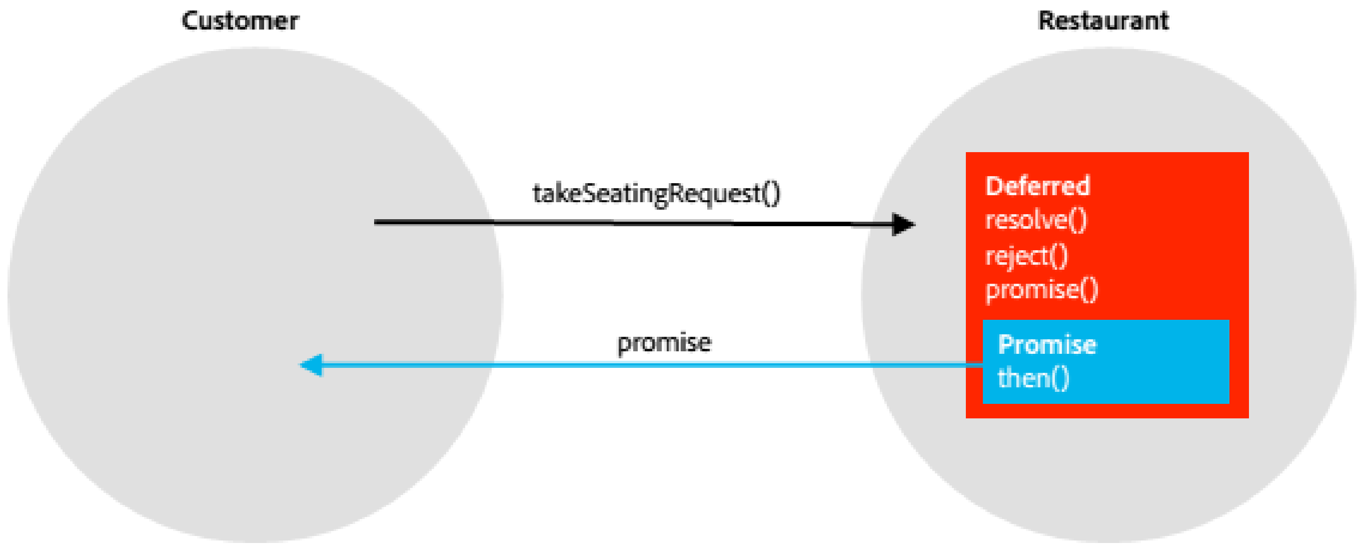
Promises In Real Life

personalized gift coasters available at www.koolcoaster.com
LRS
The Original Coaster Call
www.pager.net
800.437.4996
5

JQUERY PROMISE ANATOMY



JQUERY PROMISE ANATOMY

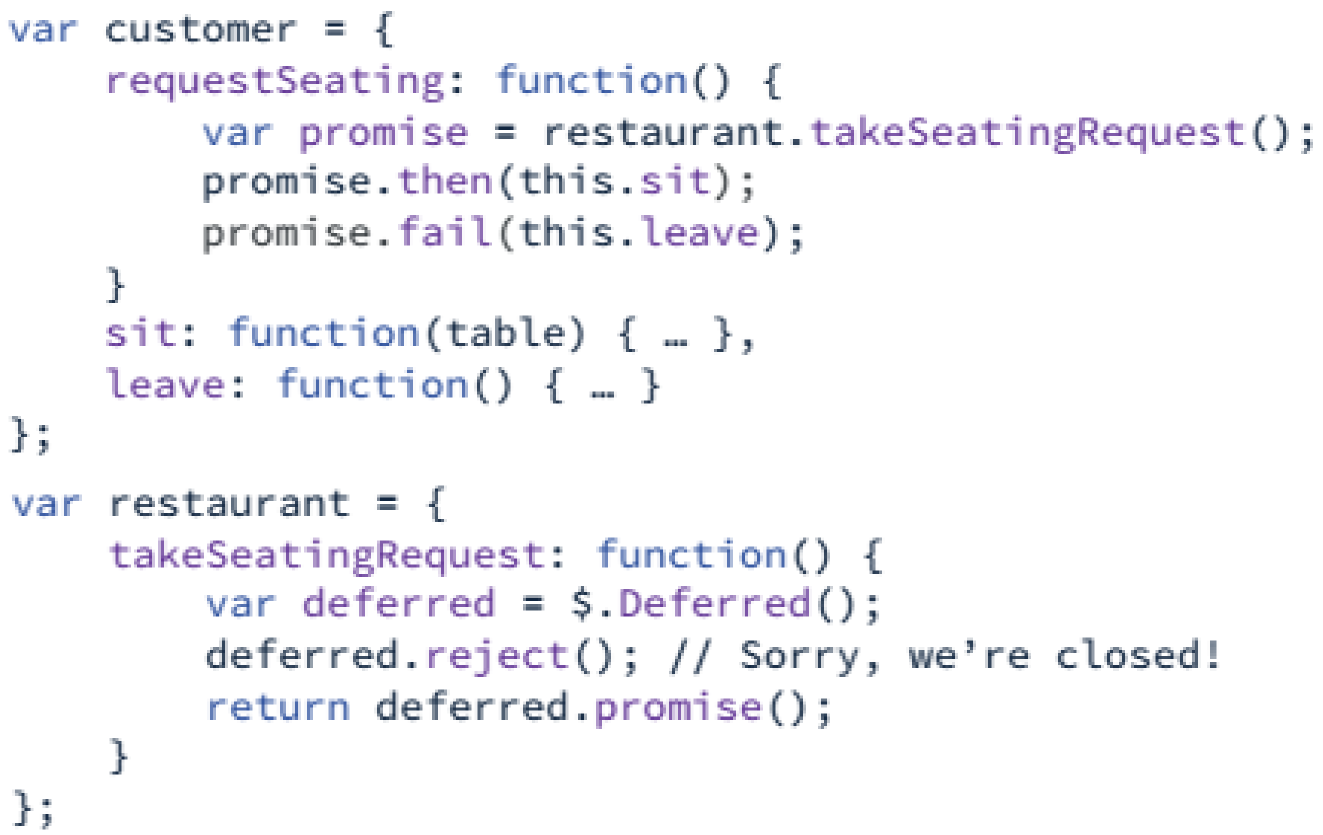


PROMISE EXAMPLE

```
var customer = {
    requestSeating: function() {
        var promise = restaurant.takeSeatingRequest();
        promise.then(this.sit);
    }
    sit: function(table) { … }
};

var restaurant = {
    takeSeatingRequest: function() {
        var deferred = $.Deferred();
        setTimeout(function() {
            deferred.resolve({seats: 4});
        }, 5000);
        return deferred.promise();
    }
};
```

Globant
we are ready

PROMISE EXAMPLE

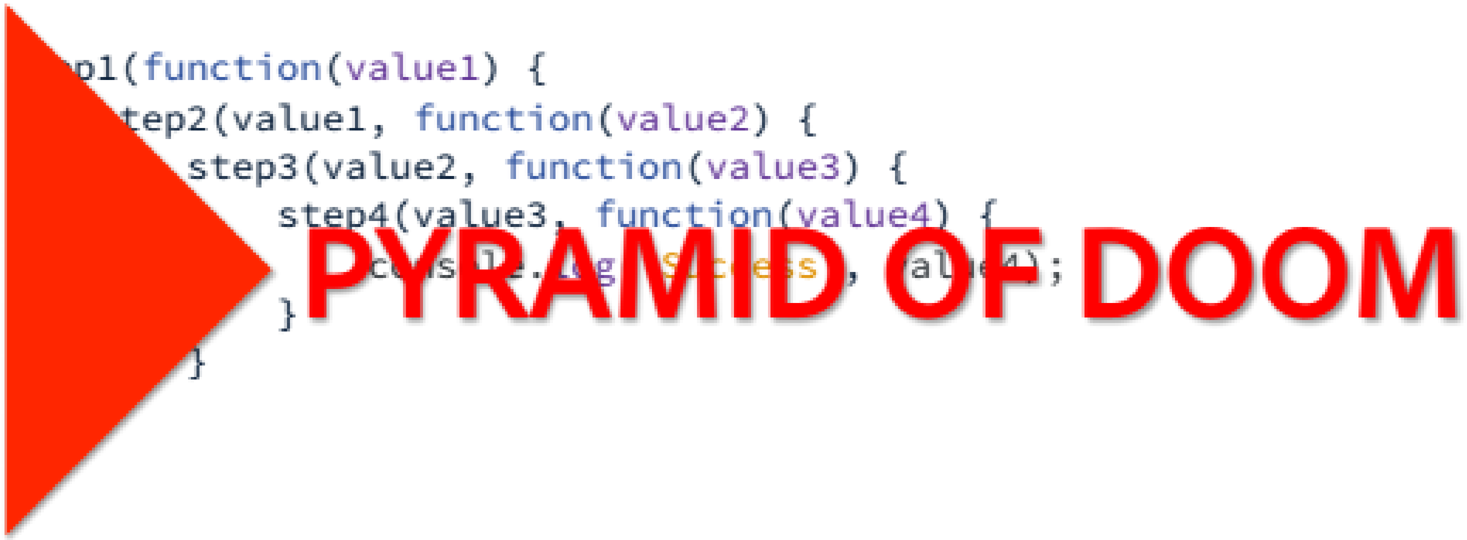


```
var customer = {
    requestSeating: function() {
        var promise = restaurant.takeSeatingRequest();
        promise.then(this.sit);
        promise.fail(this.leave);
    }
    sit: function(table) { … },
    leave: function() { … }
};

var restaurant = {
    takeSeatingRequest: function() {
        var deferred = $.Deferred();
        deferred.reject(); // Sorry, we're closed!
        return deferred.promise();
    }
};
```

ASYNCHRONOUS SEQUENCE USING CALLBACKS

```
step1(function(value1) {
    step2(value1, function(value2) {
        step3(value2, function(value3) {
            step4(value3, function(value4) {
                console.log('Success', value4);
            }
        }
    }
}
```

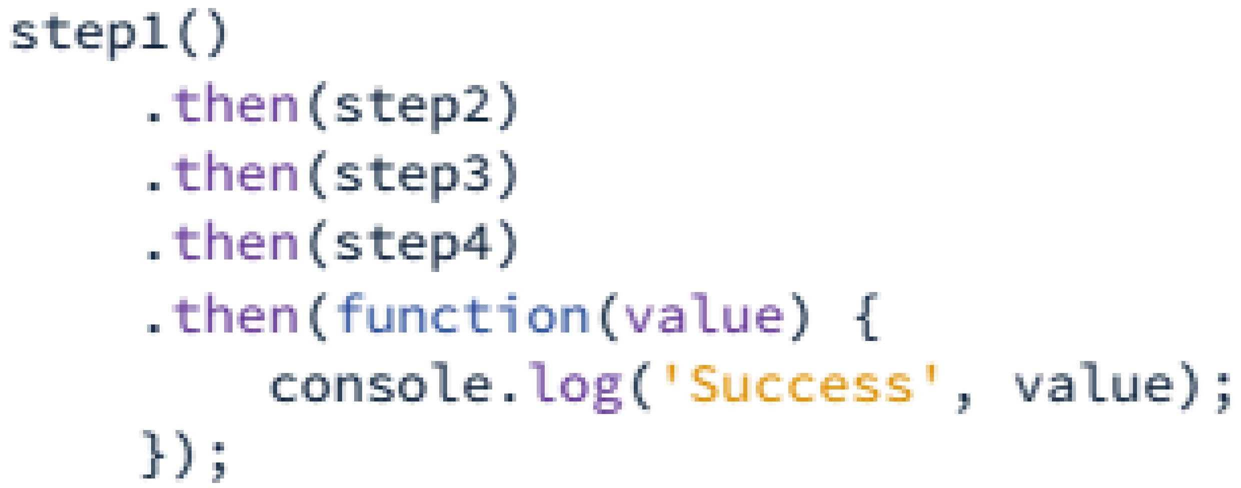
ASYNCHRONOUS SEQUENCE USING CALLBACKS



```
p1(function(value1) {
   tep2(value1, function(value2) {
        step3(value2, function(value3) {
            step4(value3, function(value4) {
                console.log("Success", value4);
            }
        }
```

PYRAMID OF DOOM

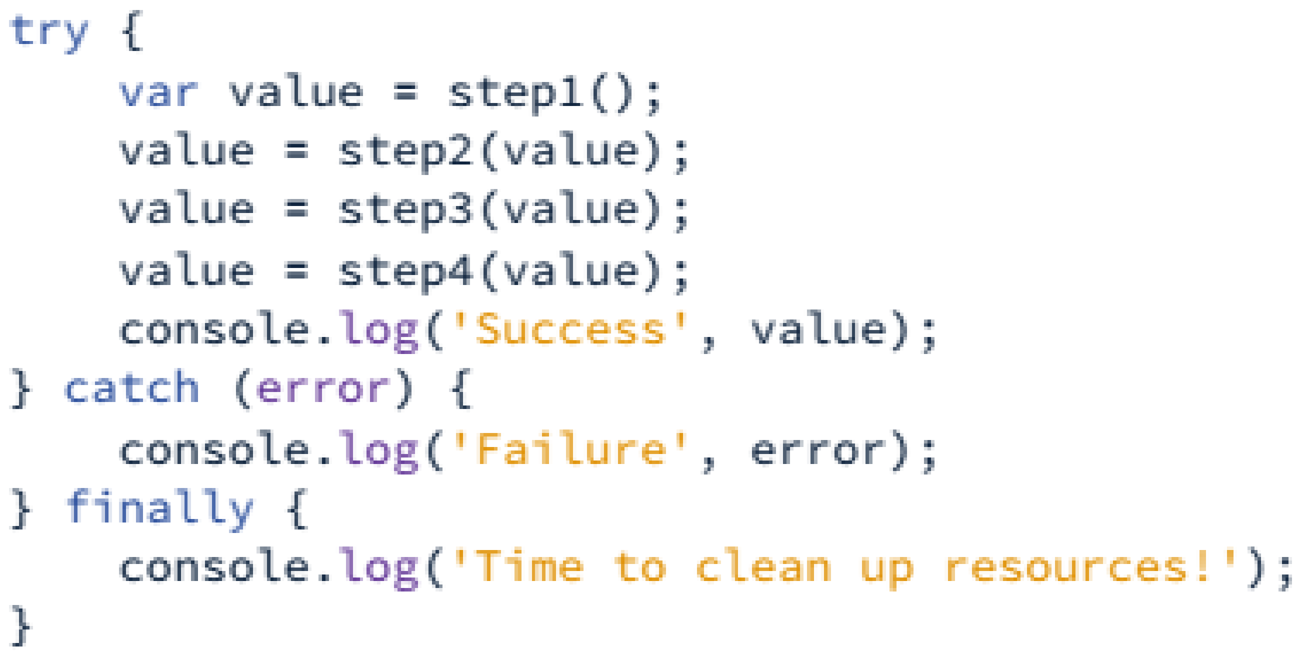
ASYNCHRONOUS SEQUENCE USING PROMISES



```
step1()
    .then(step2)
    .then(step3)
    .then(step4)
    .then(function(value) {
        console.log('Success', value);
    });
```

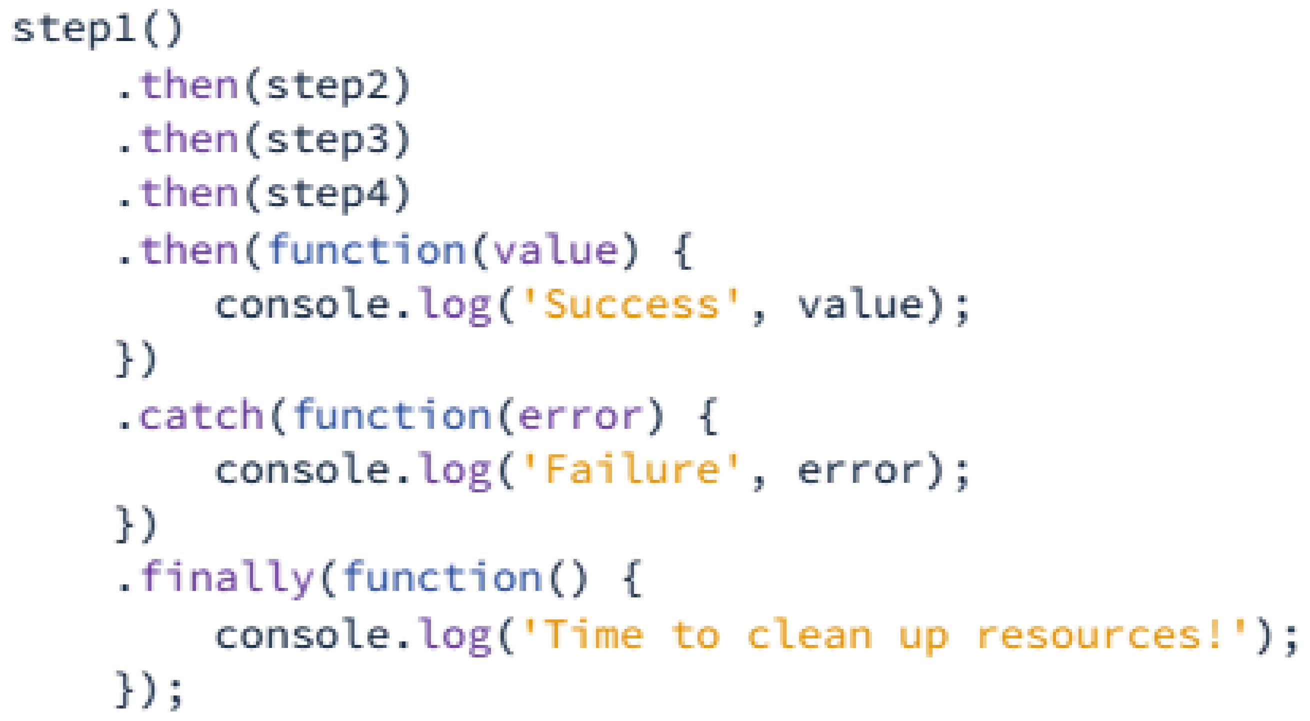
Globant
we are ready

TRY-CATCH IN A SYNCHRONOUS WORLD



```
try {
    var value = step1();
    value = step2(value);
    value = step3(value);
    value = step4(value);
    console.log('Success', value);
} catch (error) {
    console.log('Failure', error);
} finally {
    console.log('Time to clean up resources!');
}
```

ASYNCHRONOUS TRY-CATCH USING PROMISES

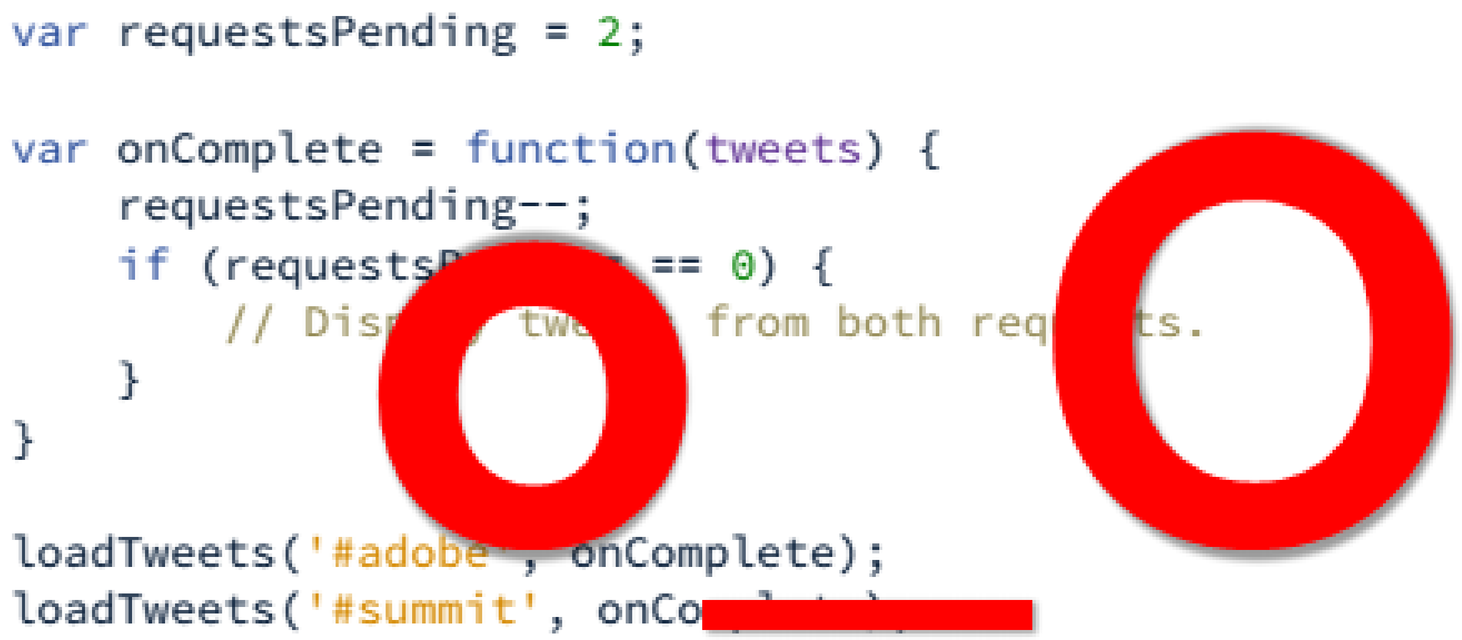


```
step1()
    .then(step2)
    .then(step3)
    .then(step4)
    .then(function(value) {
        console.log('Success', value);
    })
    .catch(function(error) {
        console.log('Failure', error);
    })
    .finally(function() {
        console.log('Time to clean up resources!');
    });
```

ASYNCHRONOUS PARALLEL USING CALLBACKS

```
var requestsPending = 2;

var onComplete = function(tweets) {
    requestsPending--;
    if (requestsPending == 0) {
        // Display tweets from both requests.
    }
}

loadTweets('#adobe', onComplete);
loadTweets('#summit', onComplete);
```

ASYNCHRONOUS PARALLEL USING CALLBACKS



```
var requestsPending = 2;

var onComplete = function(tweets) {
    requestsPending--;
    if (requestsP[illegible] == 0) {
        // Disp[illegible] twe[illegible] from both req[illegible]ts.
    }
}

loadTweets('#adobe', onComplete);
loadTweets('#summit', onCo[illegible]
```

ASYNCHRONOUS PARALLEL USING PROMISES

```
var adobePromise = loadTweets('#adobe');
var summitPromise = loadTweets('#summit');
$.when(adobePromise, summitPromise).then(displayTweets);
```

PROMISE KEY POINTS

Used to notify of completion of an **asynchronous task**

Object **passable now** representing something to be determined in the **future**

Great for **sequential/parallel** management

Generally makes use of a **third party library**

EVENT EMITTER

Used to notify of completion of an **asynchronous task**

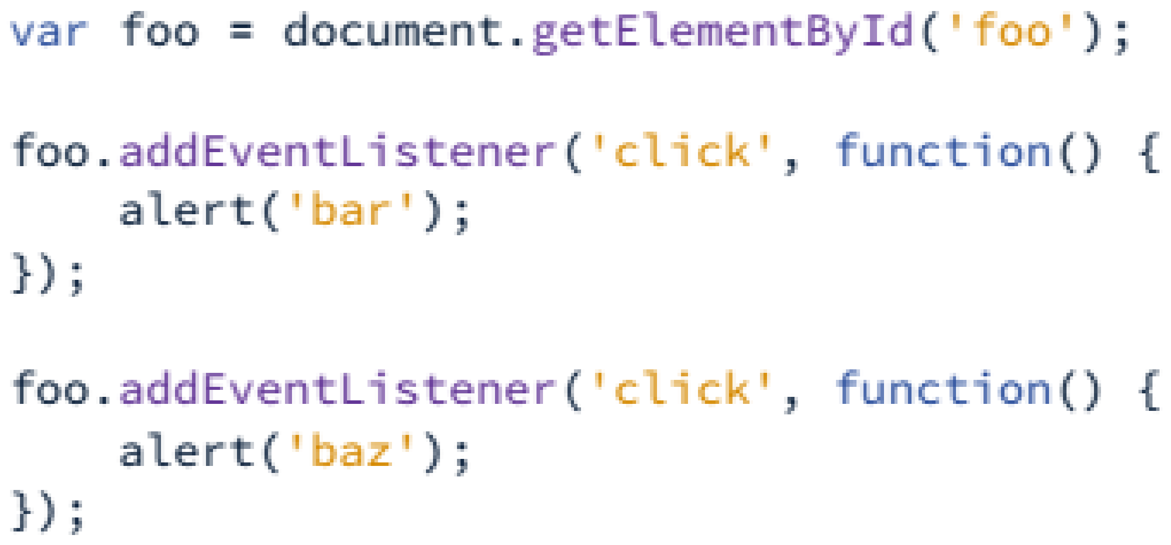
Object **passable now** representing something to be determined in the **future**

Great for **sequential/parallel** management

Generally makes use of a **third party library**

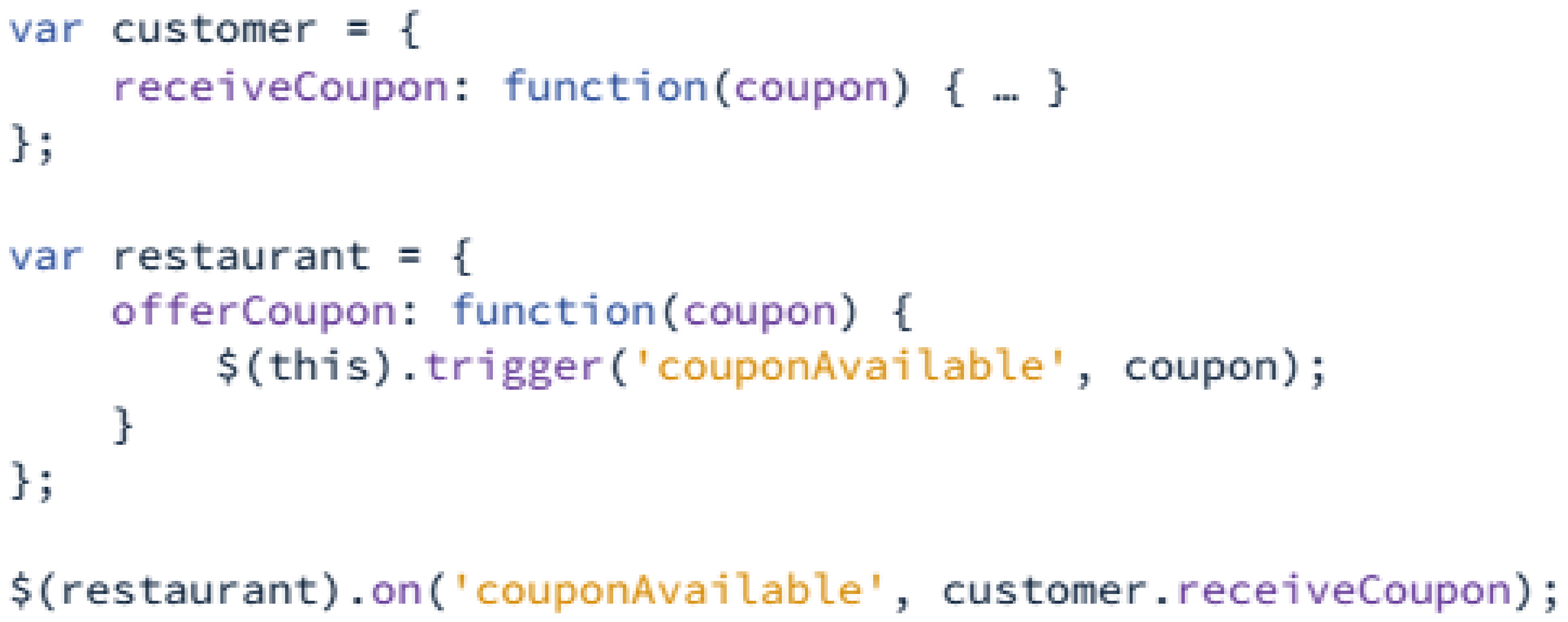
EVENT EMITTER

DOM EVENT EMITTER EXAMPLE



```
var foo = document.getElementById('foo');

foo.addEventListener('click', function() {
    alert('bar');
});

foo.addEventListener('click', function() {
    alert('baz');
});
```

JQUERY EVENT EMITTER EXAMPLE



```
var customer = {
    receiveCoupon: function(coupon) { … }
};

var restaurant = {
    offerCoupon: function(coupon) {
        $(this).trigger('couponAvailable', coupon);
    }
};

$(restaurant).on('couponAvailable', customer.receiveCoupon);
```

EVENT EMITTER KEY POINTS

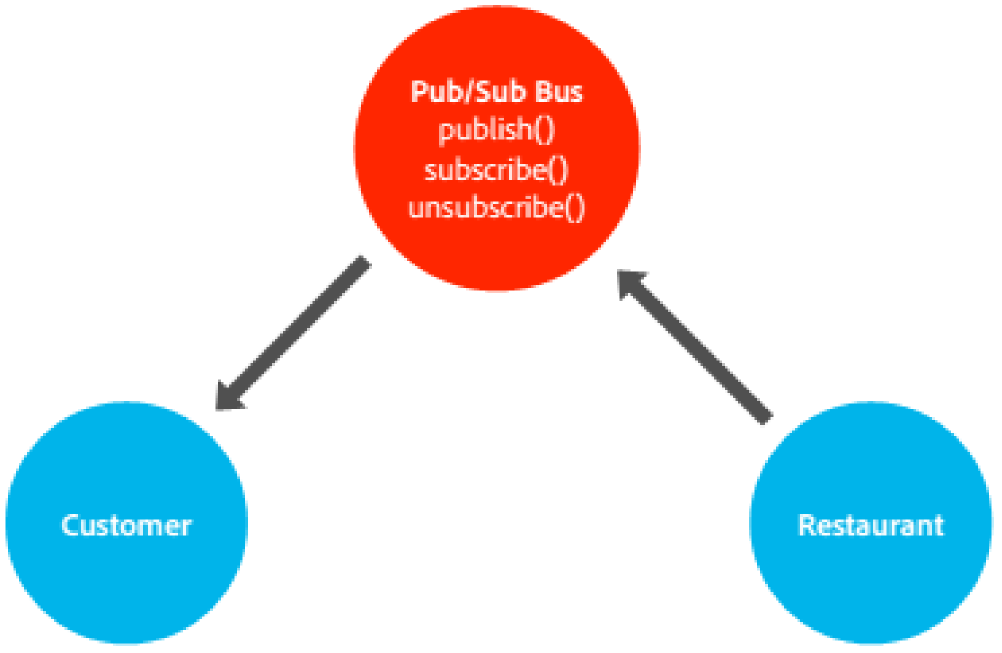
Notifies of **state change, user interaction**, etc.

Fires an event **any number of times** (possibly never)

Native for DOM

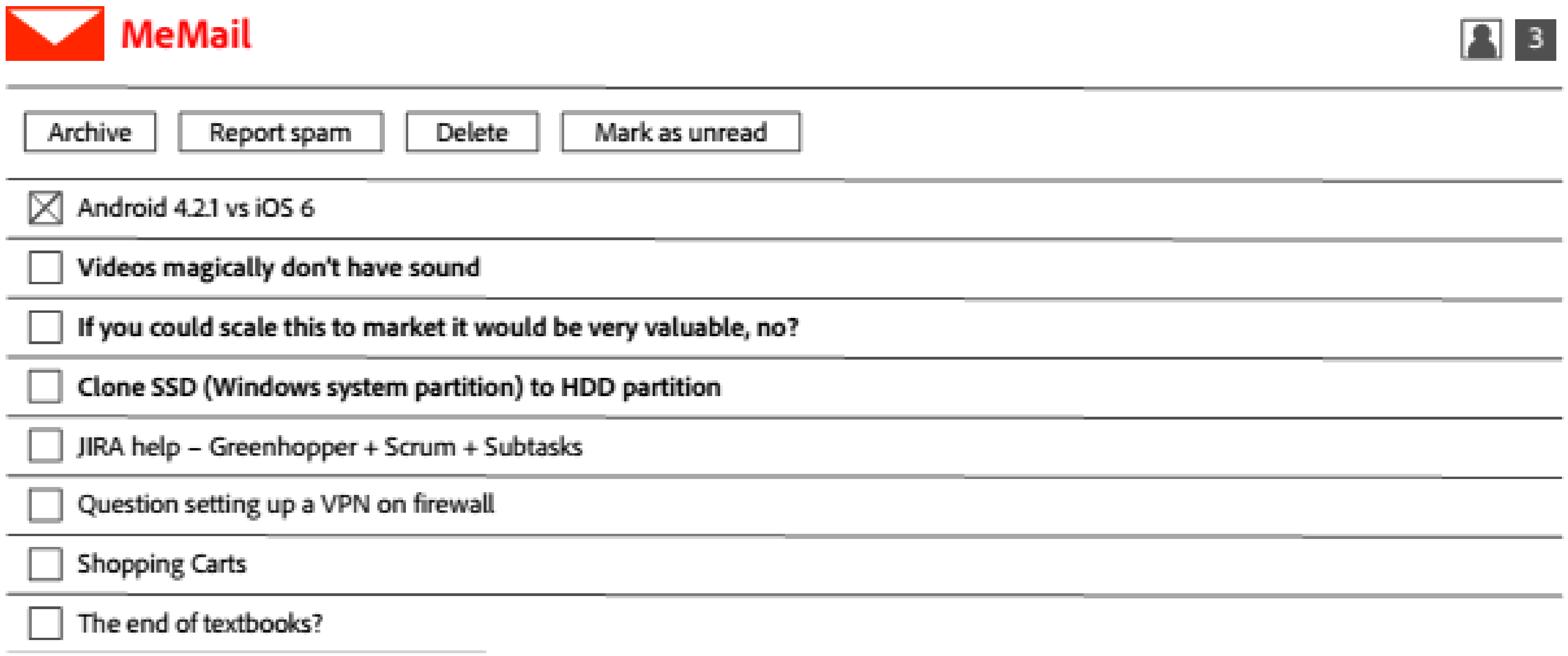
Arbitrary objects make use of a **third-party library**

PUBLISH/SUSCRIBE



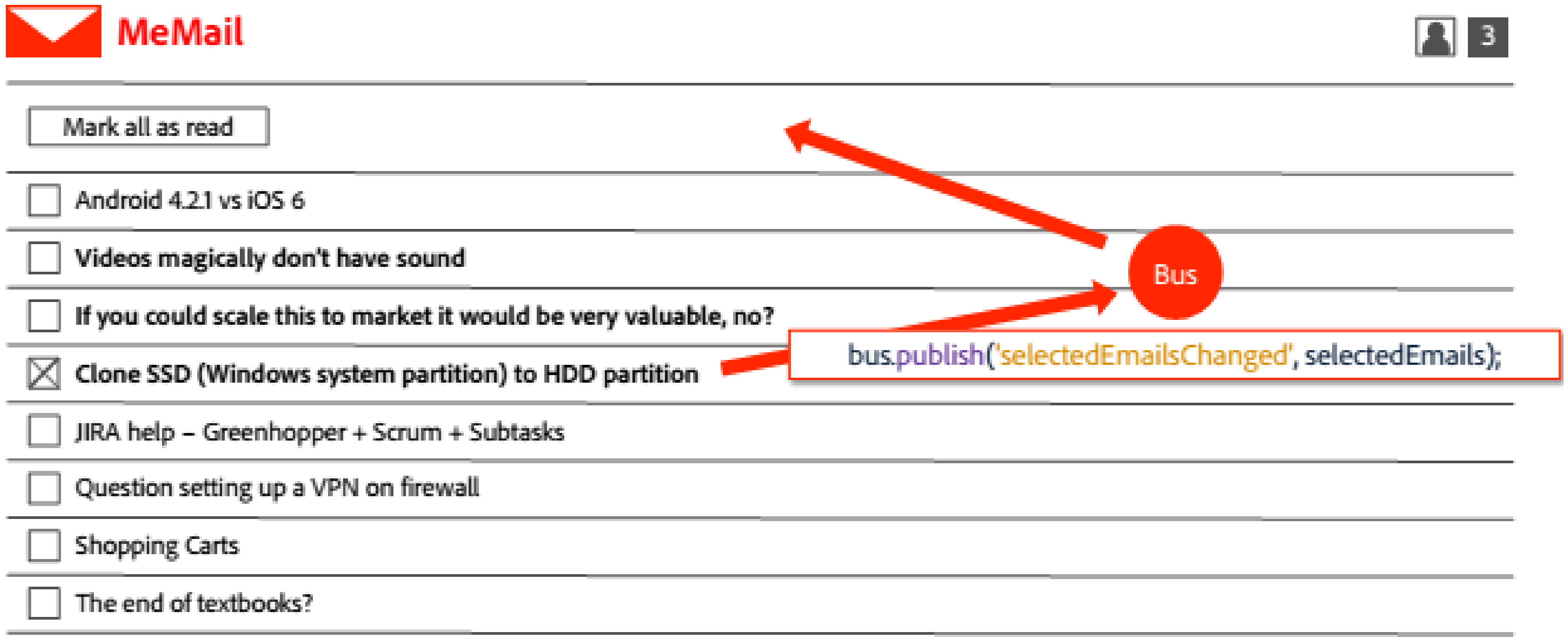
```
bus.publish('couponAvailable', 'Buy 1 get 1 free');
```

PUB/SUB EXAMPLE



- [✓] Android 4.2.1 vs iOS 6
- [ ] **Videos magically don't have sound**
- [ ] **If you could scale this to market it would be very valuable, no?**
- [ ] **Clone SSD (Windows system partition) to HDD partition**
- [ ] JIRA help – Greenhopper + Scrum + Subtasks
- [ ] Question setting up a VPN on firewall
- [ ] Shopping Carts
- [ ] The end of textbooks?

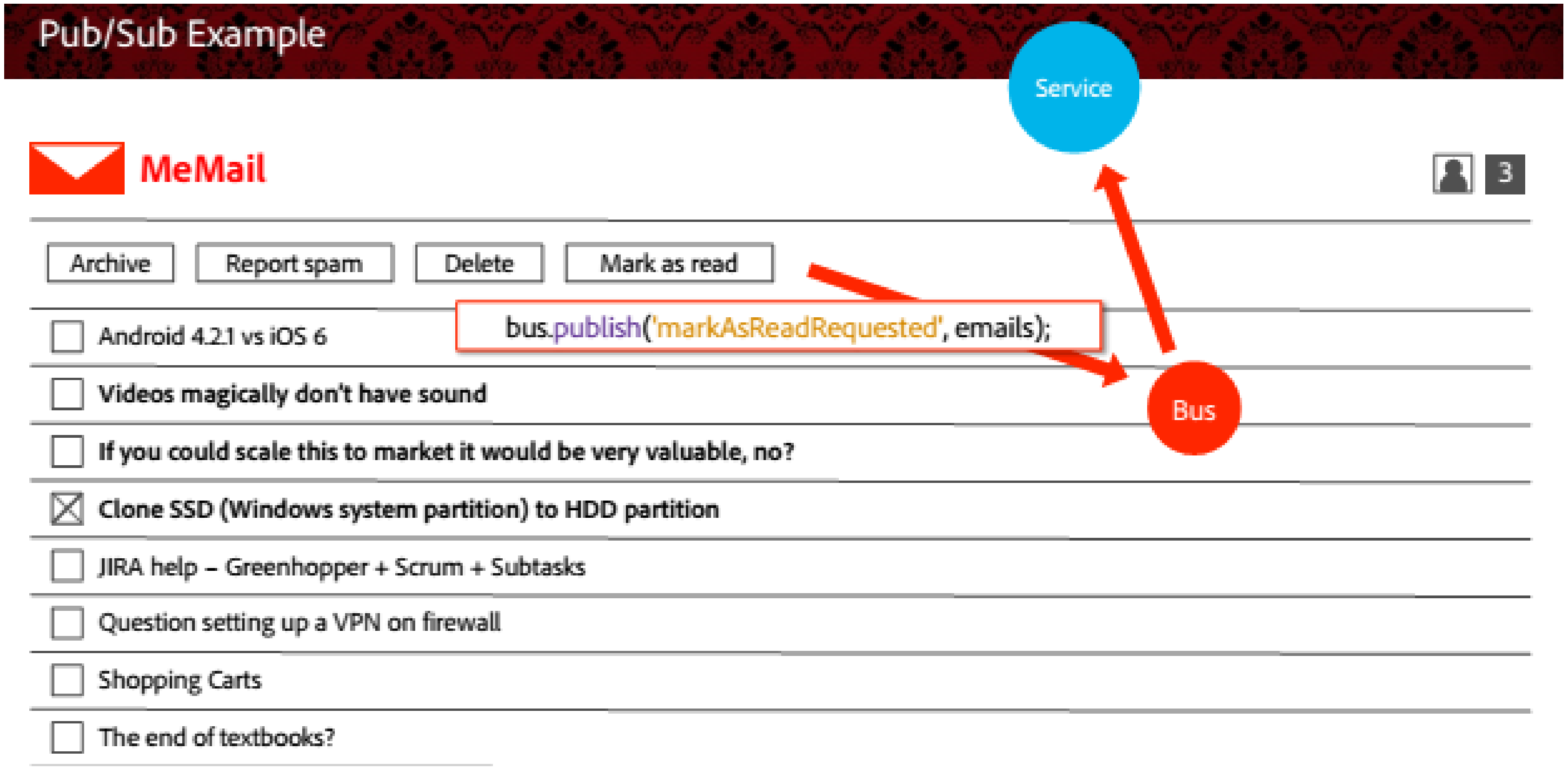
PUB/SUB EXAMPLE



- [ ] Android 4.2.1 vs iOS 6
- [ ] Videos magically don't have sound
- [ ] If you could scale this to market it would be very valuable, no?
- [x] Clone SSD (Windows system partition) to HDD partition
- [ ] JIRA help – Greenhopper + Scrum + Subtasks
- [ ] Question setting up a VPN on firewall
- [ ] Shopping Carts
- [ ] The end of textbooks?

```
bus.publish('selectedEmailsChanged', selectedEmails);
```

PUB/SUB EXAMPLE



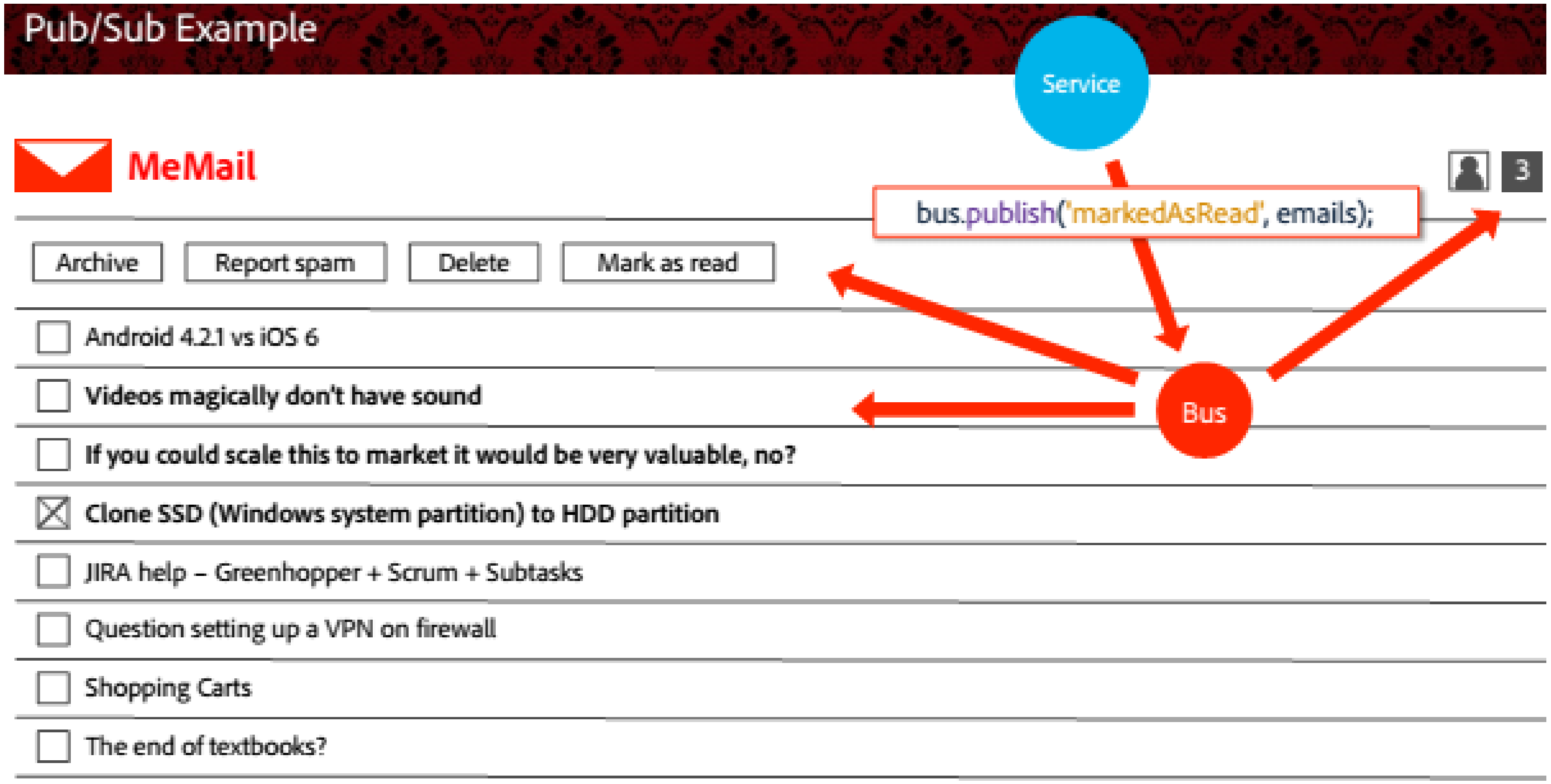
Pub/Sub Example

MeMail

```
bus.publish('markAsReadRequested', emails);
```

- [ ] Android 4.2.1 vs iOS 6
- [ ] Videos magically don't have sound
- [ ] If you could scale this to market it would be very valuable, no?
- [x] Clone SSD (Windows system partition) to HDD partition
- [ ] JIRA help – Greenhopper + Scrum + Subtasks
- [ ] Question setting up a VPN on firewall
- [ ] Shopping Carts
- [ ] The end of textbooks?

PUB/SUB EXAMPLE


Pub/Sub Example

MeMail

```
bus.publish('markedAsRead', emails);
```

- [ ] Android 4.2.1 vs iOS 6
- [ ] **Videos magically don't have sound**
- [ ] **If you could scale this to market it would be very valuable, no?**
- [x] **Clone SSD (Windows system partition) to HDD partition**
- [ ] JIRA help – Greenhopper + Scrum + Subtasks
- [ ] Question setting up a VPN on firewall
- [ ] Shopping Carts
- [ ] The end of textbooks?

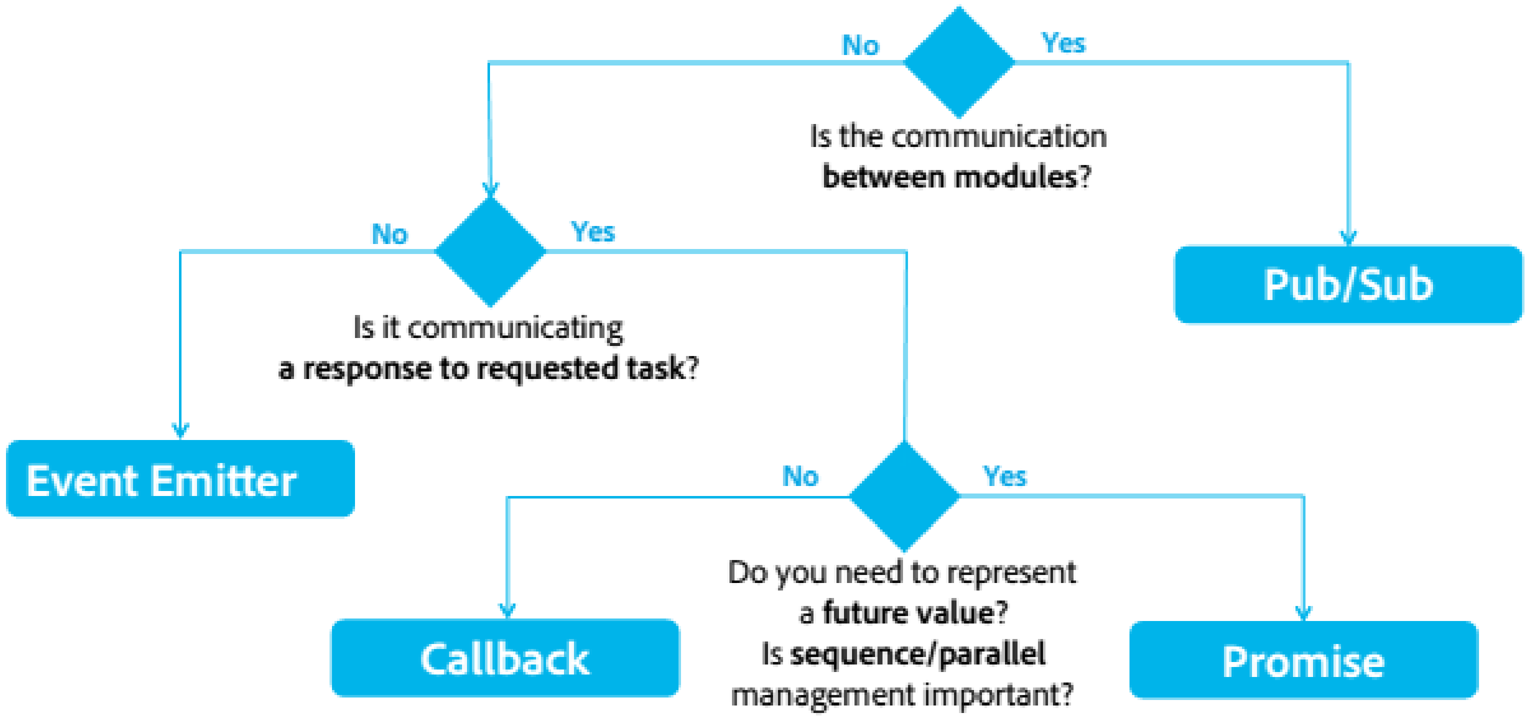
PUB/SUB KEYS POINTS

Communication **between modules**

Publishers and subscribers **don't address one another**

Provides excellent **decoupling**

WHICH PATTERN SHOULD I USE


No
Yes
Is the communication between modules?
Pub/Sub
No
Yes
Is it communicating a response to requested task?
Event Emitter
No
Yes
Do you need to represent a future value? Is sequence/parallel management important?
Callback
Promise

Thank you!

Globant
we are ready