CIS 4004: Web Based Information Technology
Spring 2013

Introduction To JavaScript – Part 5
The Event Object

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cis4004/spr2013

Department of Electrical Engineering and Computer Science
University of Central Florida

JavaScript – Part 5 – The Event Object

- When a W3C event listener's event occurs and it calls its associated event handler (function), it also passes a single argument to the function – a reference to the event object.
- The event object contains a number of properties that describe the event that occurred.
- The table on the next page lists the names of the most commonly used properties of the event object, which of course usually differ between the W3C and Microsoft models.

JavaScript – Part 5 – The Event Object

W3C Name	Microsoft Name	Description
`e`	`window.event`	The object containing the event properties
`type`	`type`	The event that occurred (click, focus, blur, etc.)
`target`	`srcElement`	The element to which the event occurred
`keyCode`	`keyCode`	The numerical ASCII value of the pressed key

`shiftKey`
`altKey`
`cntlKey`
		Returns 1 if pressed, 0 if not

`currentTarget`	`fromElement`	The element the mouse came from on mouseover
`relatedTarget`	`toElement`	The element the mouse went to on mouseout

Table of the most commonly used event object properties

JavaScript – Part 5 – The Event Object

- By convention, the parameter name `e` is used in event-triggered functions to receive the event object argument.
- If you wanted to determine the type of event that occurred, you'd write the following function:

```
function myEvent(e) {
    var eventType = e.type
    alert("The following event has occurred: ", eventType);
    //the alert will display click, or whatever the event
    //type was
}
```


JavaScript – Part 5 – The Event Object

- Note that this code would not work on Microsoft browsers, because the Microsoft model does not pass an event object reference like the W3C model; instead, it uses a central global object that contains the properties of the most recent event.
- We'll start by looking at the W3C model before we look at the Microsoft model.
- To demonstrate how the W3C event model works, we'll go back and modify the highlighting example from the previous set of notes.
- We'll modify the JavaScript functions `addHighlight()` and `removeHighlight()` so that they no longer have to get the target element before modifying it.

K:\CIS 4004 - Fall 2012\code\JavaScript - Part 5\hilite_field_basic3.html - Notepad++

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ? X

CH08_SaleCo2.SQL | schedule.dat | utlxplan.sql | QRYOPTDATA.SQL | asic3.html

Previous Version

```
15
16  window.onload=setUpFieldEvents;
17
18  function setUpFieldEvents() {
19      var emailField=document.getElementById("email"); // get the field
20      addEvent(emailField, 'focus', addHighlight); // add focus event
21      addEvent(emailField, 'blur', removeHighlight); // add blur event
22      }
23  function addHighlight() {
24      var emailField=document.getElementById("email");
25      emailField.style.backgroundColor="#6F3";
26      }
27  function removeHighlight() {
28      var emailField=document.getElementById("email");
29      emailField.style.backgroundColor=""; // field now goes back to default settings
30      }
31  </script>
32  </head>
33
34  <body>
35      <div id="sign_up">
36          <h3>Sign up for our newsletter</h3>
37          <form id="email_form" action="#" method="get">
38              <label for="email">Email</label>
39              <input id="email" name="email" type="text" size="24" />
```

length : 1261 lines : 45 | Ln : 16 Col : 32 Sel : 0 | UNIX | ANSI | INS

K:\CIS 4004 - Fall 2012\code\JavaScript - Part 5\hilite_field_basic3 - event object version.html - Notep...

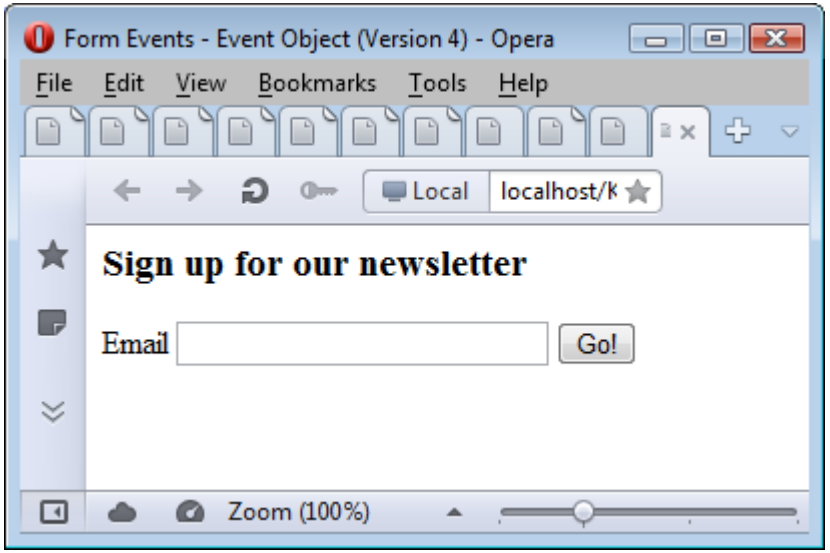
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ? X

QRYOPTDATA.SQL | notes.html | hilite_field_basic3 - event object version.h

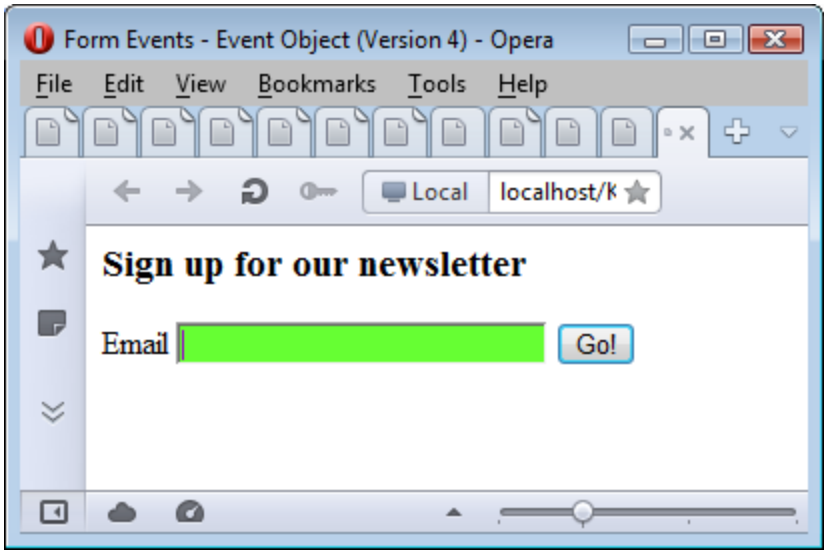
Event Object Version

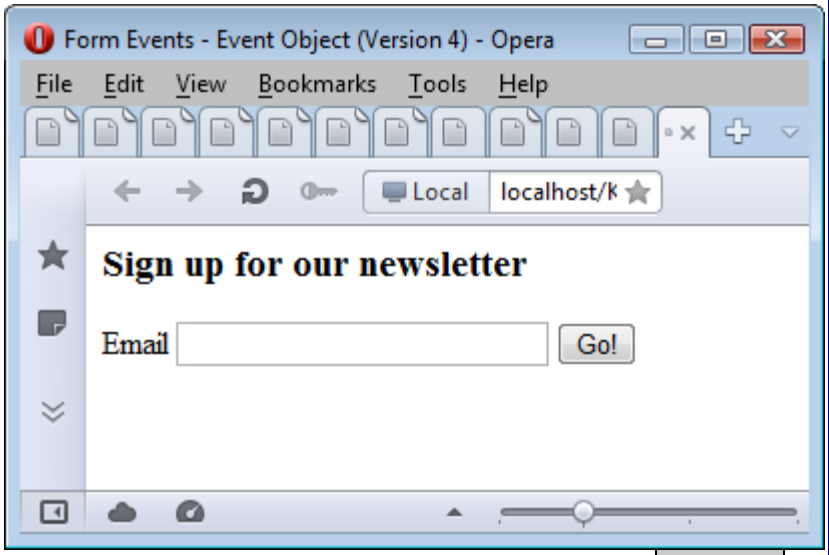
```
15
16  window.onload=setUpFieldEvents;
17
18  function setUpFieldEvents() {
19      var emailField=document.getElementById("email"); // get the field
20      addEvent(emailField, 'focus', addHighlight); // add focus event
21      addEvent(emailField, 'blur', removeHighlight); // add blur event
22      }
23  function addHighlight(e) {
24      var emailField=e.target;
25      emailField.style.backgroundColor="#6F3";
26      }
27  function removeHighlight(e) {
28      var emailField=e.target;
29      emailField.style.backgroundColor=""; // field now goes back to default settings
30      }
31  </script>
32  </head>
33
34  <body>
35      <div id="sign_up">
36          <h3>Sign up for our newsletter</h3>
37          <form id="email_form" action="#" method="get">
38              <label for="email">Email</label>
```

length : 1225 lines : 45 | Ln : 28 Col : 29 Sel : 0 | UNIX | ANSI | INS



Page operates as before





JavaScript – Part 5 – The Event Object

- As you can see, there is no visual change in how the page behaves when using the event object when compared to how the page behaved when accessing the target object directly via the DOM element.
- However, you could now assign this same event listener to multiple input fields, and those fields would all display the same highlight behavior.
- Instead of saying "highlight this field", the code could state "highlight the field to which the event occurred."
- This gives us the ability to streamline the JavaScript to some extent. The next example will illustrate this.

The Event Object's Type Property

- With access to the event object, you can also determine the type of event that occurred, e.g. `focus`, `blur`, `click`, etc., so you can use a single function to detect both the focus and the blur events.
- To do this, we'll again modify the previous example.
- We'll modify the event listeners to call the same function, we'll call this function `checkHighlight`, which will make more sense because this function will both add and remove the highlighting.

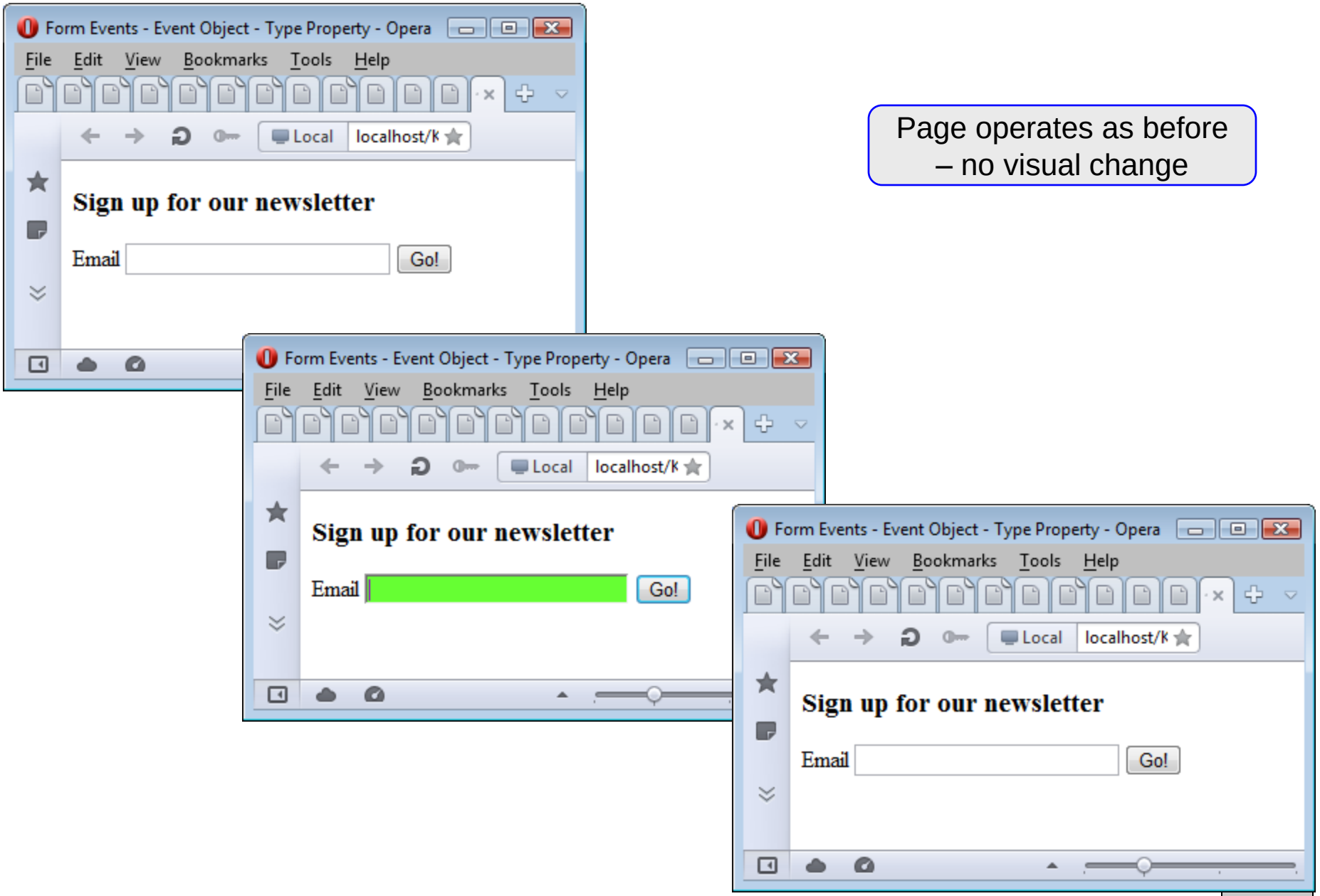
K:\CIS 4004 - Fall 2012\code\JavaScript - Part 5\hilite_field_basic5 - event type.html - Notepad++

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ? X

hilite_field_basic3 - event object version.html | hilite_field_basic4.html | hilite_field_basic5 - event type.html

```
14  }
15
16  window.onload=setUpFieldEvents;
17
18  function setUpFieldEvents() {
19      var emailField=document.getElementById("email"); // get the field
20      addEvent(emailField, 'focus', checkHighlight); // add focus event
21      addEvent(emailField, 'blur', checkHighlight); // add blur event
22      }
23
24  function checkHighlight(e) {
25      switch (e.type) {
26          case "focus":
27              e.target.style.backgroundColor="#6F3";
28              break;
29          case "blur":
30              e.target.style.backgroundColor="";
31              break;
32      }
33  }
34  </script>
35  </head>
36
37  <body>
```

length : 1176 lines : 48 | Ln : 6 Col : 22 Sel : 0 | UNIX | ANSI as UTF-8 | INS



Page operates as before – no visual change

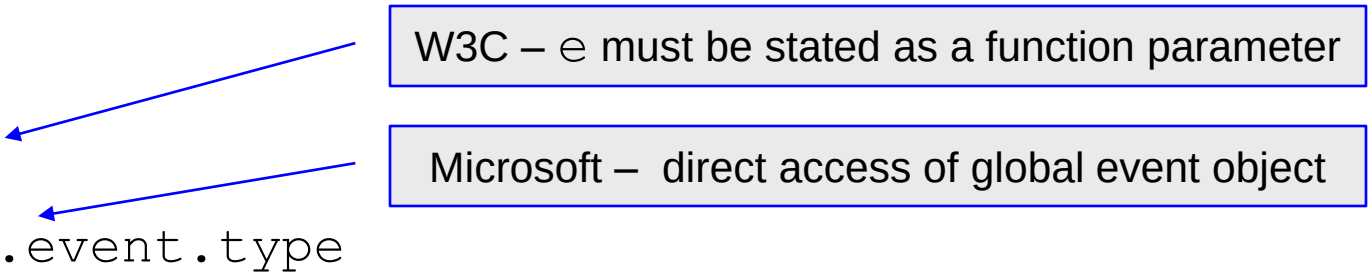
The Event Object In Microsoft Browsers

- When an event triggers a function in W3C-compliant browsers, a reference to an object containing properties that describe the triggering event is passed to the function. This event object can be accessed through the `e` parameter.
- In Microsoft browsers, the model is slightly different. There is one global object, `window.event`, that holds the last event that occurred.
- Because its global, it doesn't have to be passed to the function like the W3C event object; it's always available to your code.

The Event Object In Microsoft Browsers

- The following two lines are equivalent in their respective browsers:

```
e.type

window.event.type
```



- The simplest way to write cross-browser event object code is like this:

```
function eventType(e) {
        if (e) { alert(e.type) }//W3C
        else { alert(window.event.type) } //Microsoft
        //displays the triggering event
}
```


The Event Object In Microsoft Browsers

- While the technique on the previous page works fine, branching your code for every event object property you want to use gets old fast.
- A better solution is to get the object, whichever kind it is , and give it a new name and use the `OR` operator to distinguish:

```
var evt = e || window.event;
```

- If `e` evaluates to true (a W3C object exists, the `evt` variable is set to `e` – the W3C event object with all its properties. If not, `evt` is set to the Microsoft object instead. Now you would have:

```
var evt = e || window.event;
alert(evt.type);
```


The Event Object In Microsoft Browsers

- The preceding code works because, unlike many event object properties, the property name for the type of event that occurred is the same – `type` – in both W3C and Microsoft browsers.
- If you want to get the event target, which is `target` for W3C-compliant browsers and `srcElement` for Microsoft browsers, you can expand on the previous example and again use an `OR` statement to create a common cross-browser name for the event target as well:

```
var evt = e || window.event;

var evtTarget = evt.target || evt.srcElement;

alert(evtTarget);
```

The Event Object In Microsoft Browsers

- The next example applies this concept to create a cross-browser compatible version of the highlighted form example that we've been dealing with in the few previous examples.
- Notice again, that there are no visual changes to the way the page behaves with the exception that it now will also work in a Microsoft browser.
- The JavaScript is shown on the next page.

*K:\CIS 4004 - Fall 2012\code\JavaScript - Part 5\hilite_field_basic6.html - Notepad++

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ? X

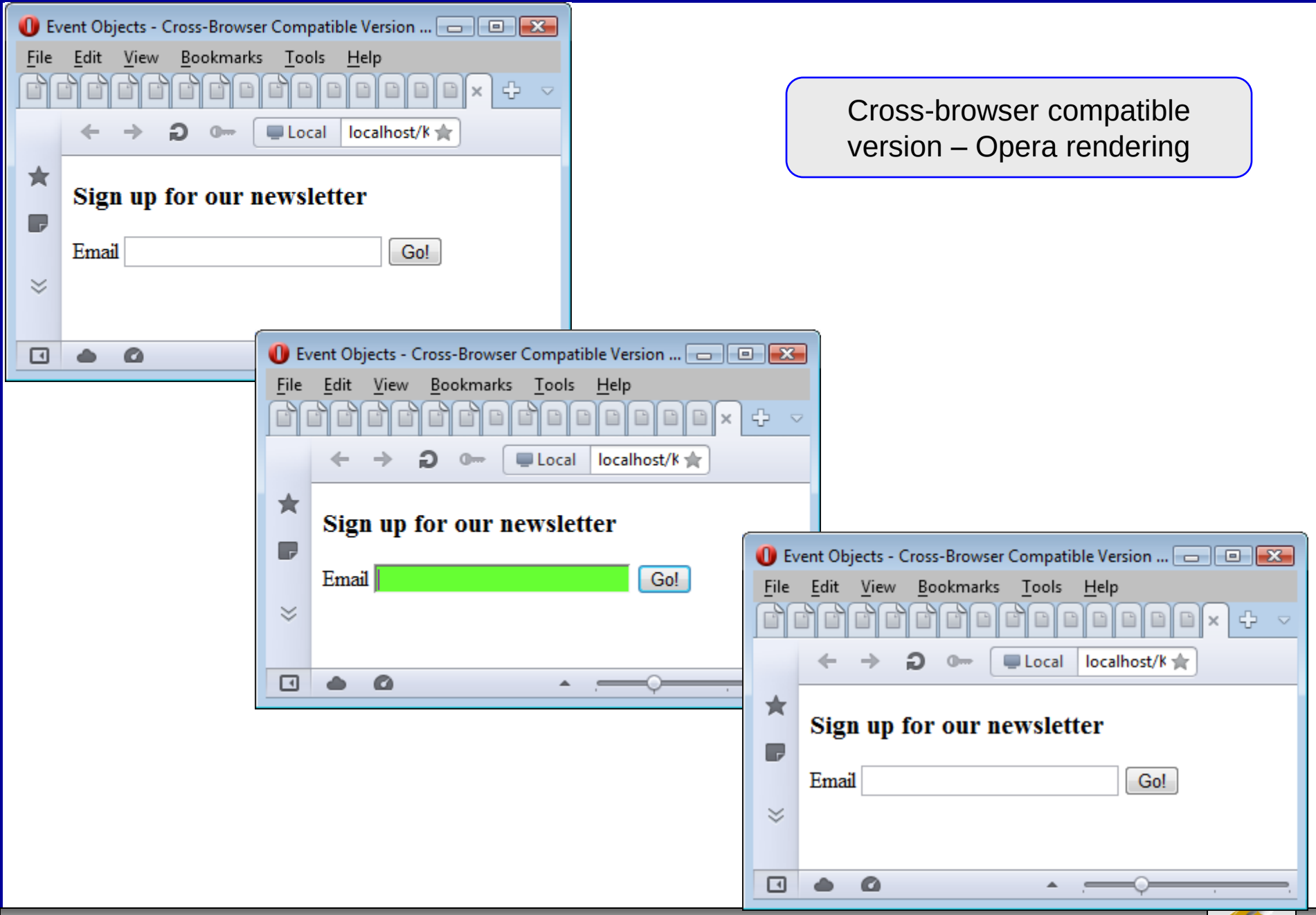
CH08_SaleCo2.SQL | schedule.dat | utlxplan.sql | QRYOPTDATA.SQL | notes.html | hilite_field_basic6.html

```
16  window.onload=setUpFieldEvents;
17
18  function setUpFieldEvents() {
19      var emailField=document.getElementById("email"); // get the field
20      addEvent(emailField, 'focus', checkHighlight); // add focus event
21      addEvent(emailField, 'blur', checkHighlight); // add blur event
22      }
23
24  function checkHighlight(e) {
25      var evt = e || window.event; // sets evt variable to either W3C or Microsoft event obj
26      var evtTarget = evt.target || evt.srcElement; // gets the target of the event
27      switch (evt.type) { // type is the same name in both objs
28          case "focus":
29              evtTarget.style.backgroundColor="#6F3";
30              break;
31          case "blur":
32              evtTarget.style.backgroundColor="";
33              break;
34      }
35  }
36  </script>
37  </head>
38
39  <body>
```

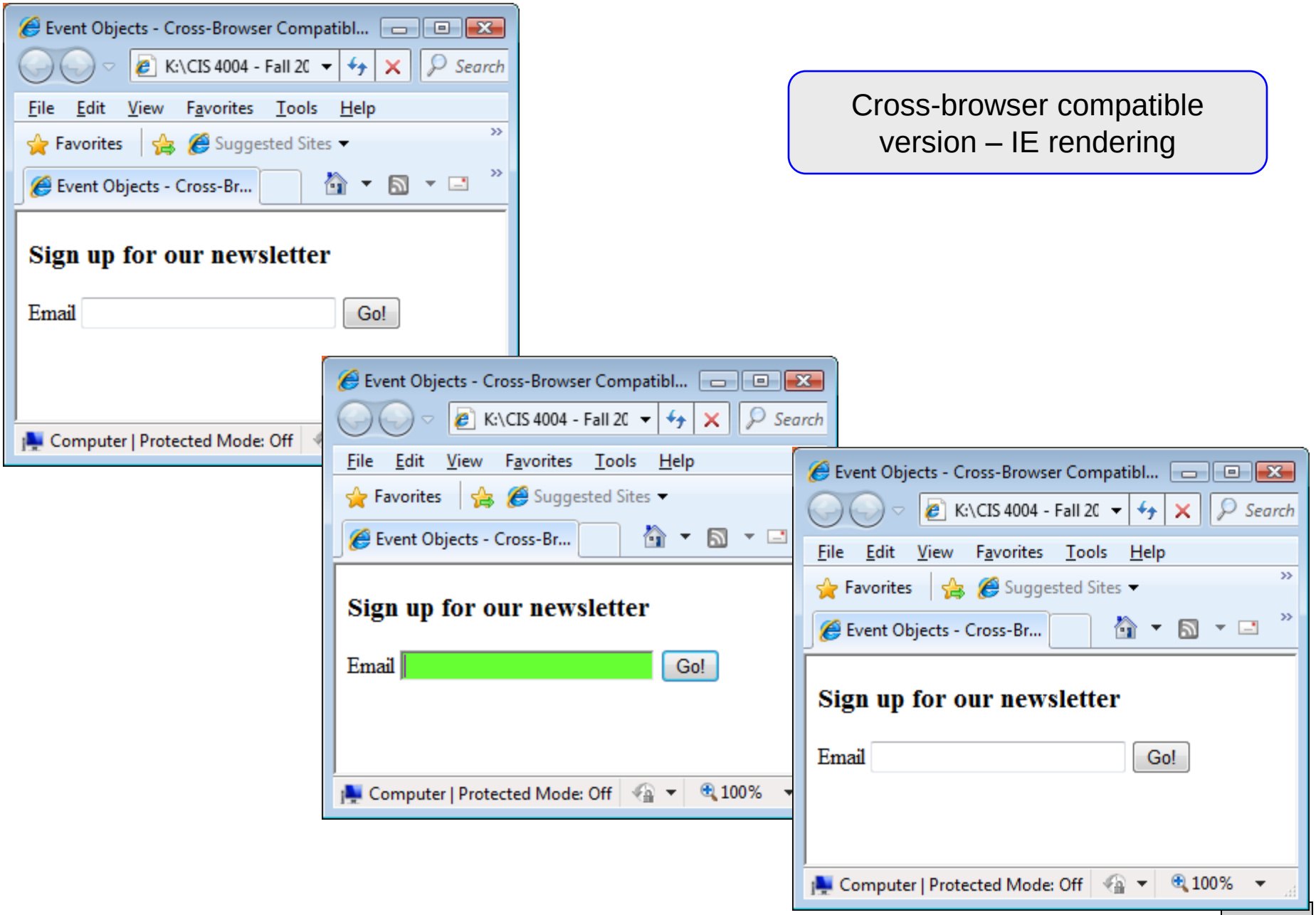
Cross-browser compatible version of the single form field highlighting example

Hyper Text M | length : 1362 | lines : 50 | Ln : 5 | Col : 22 | Sel : 0 | UNIX | ISO 8859-1 | INS

Cross-browser compatible version – Opera rendering

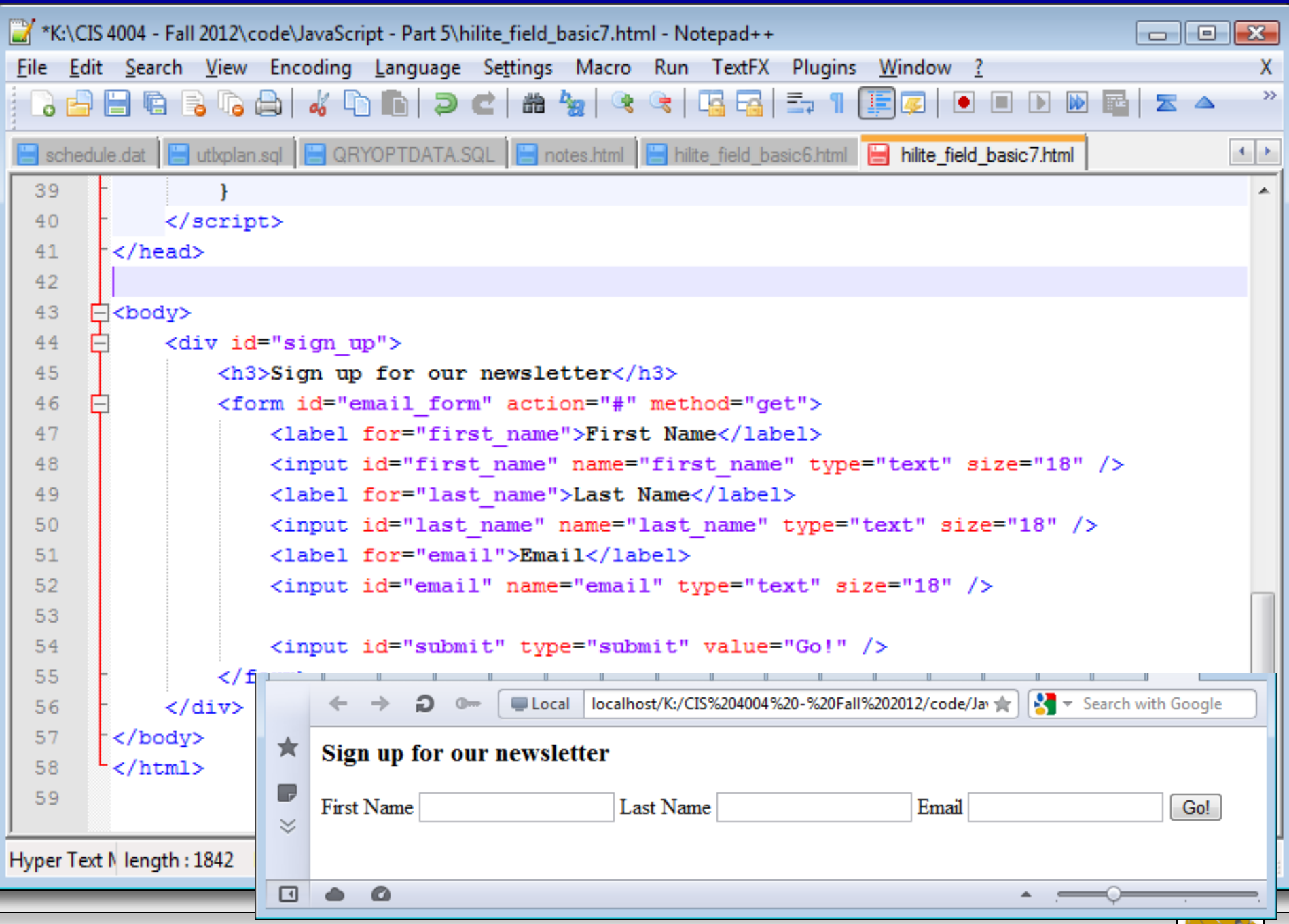


Cross-browser compatible version – IE rendering



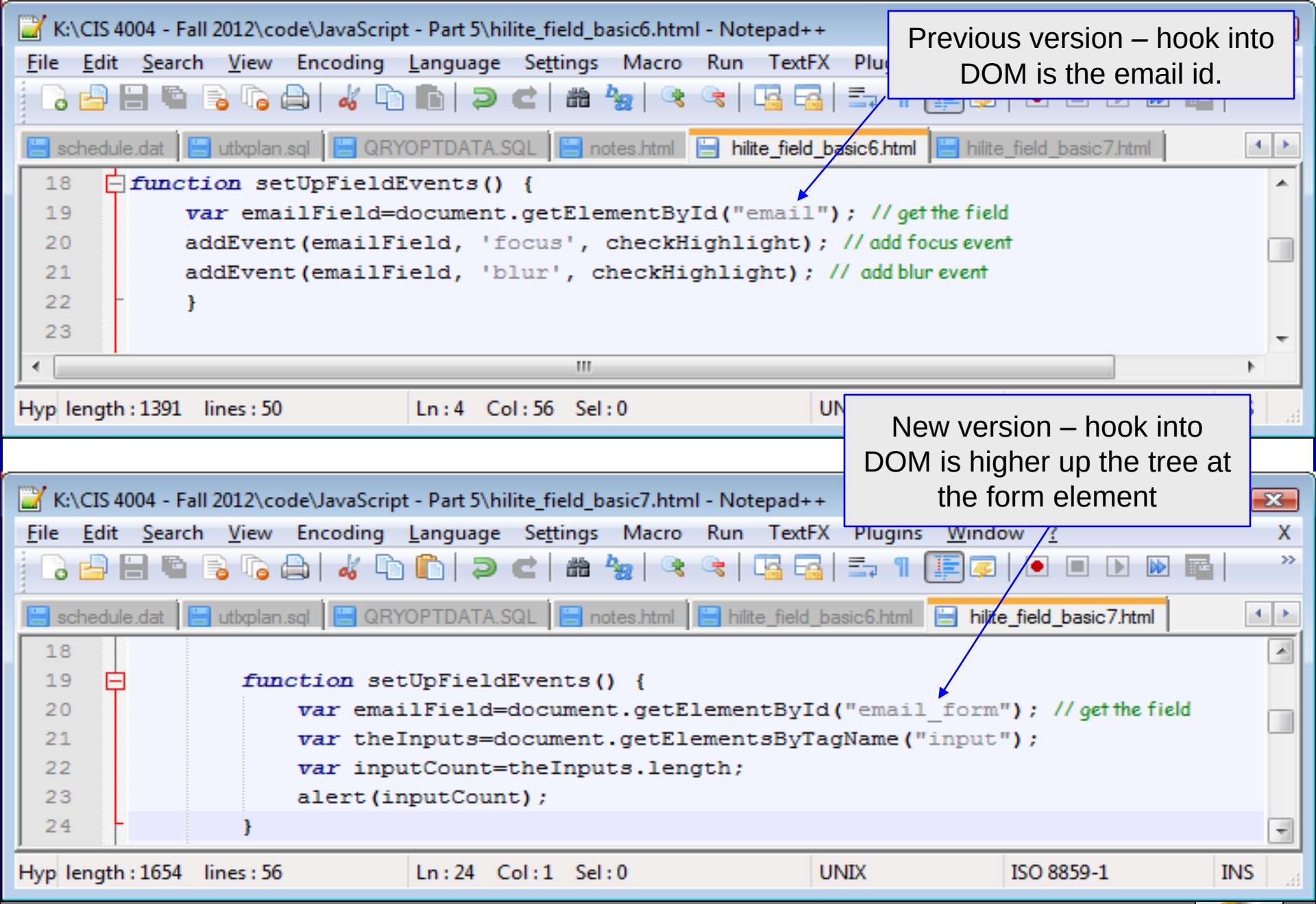
Attaching Event Handlers To Multiple Fields

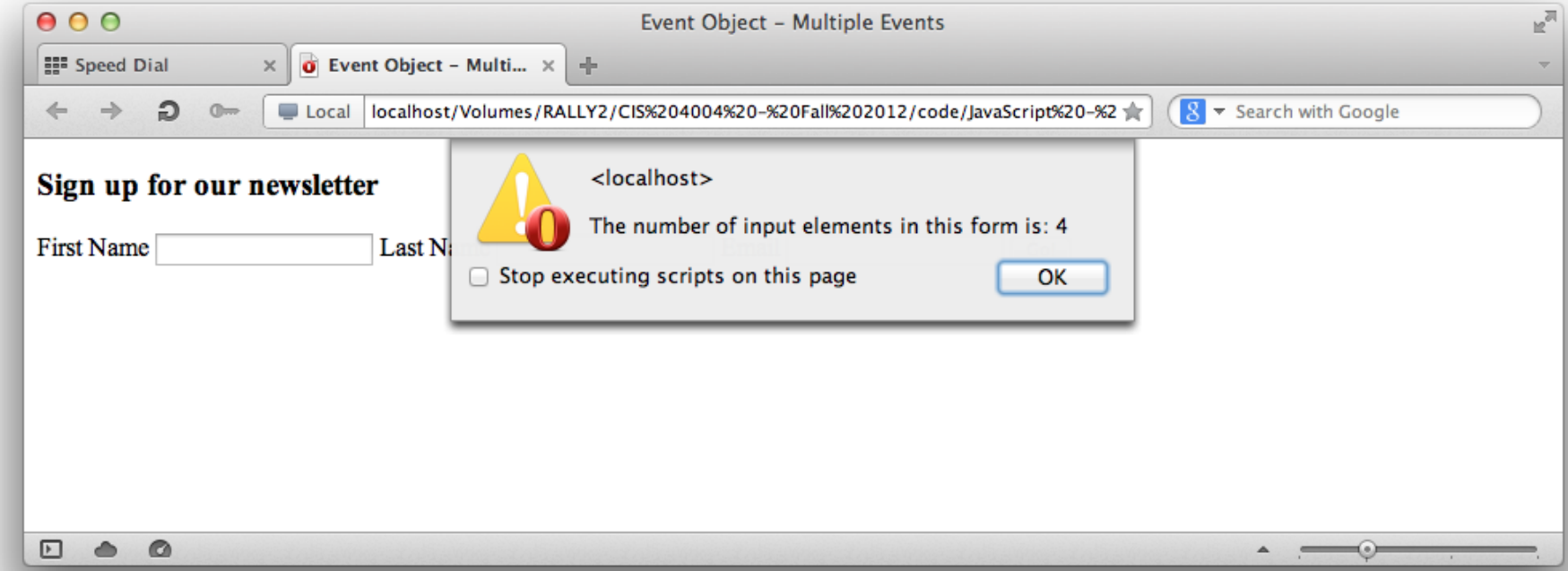
- The running example that we've been using has included a single field in the form to which we've been attaching event handlers and responding to events that happened in that particular field.
- The versatility of the event object is in allowing you to attach event handlers to as many fields as you would like.
- To illustrate, we'll add a couple more fields to the basic form we've been using. In addition to the email field, I've added fields for the user to enter their first and last names.
- The markup is shown on the next page.

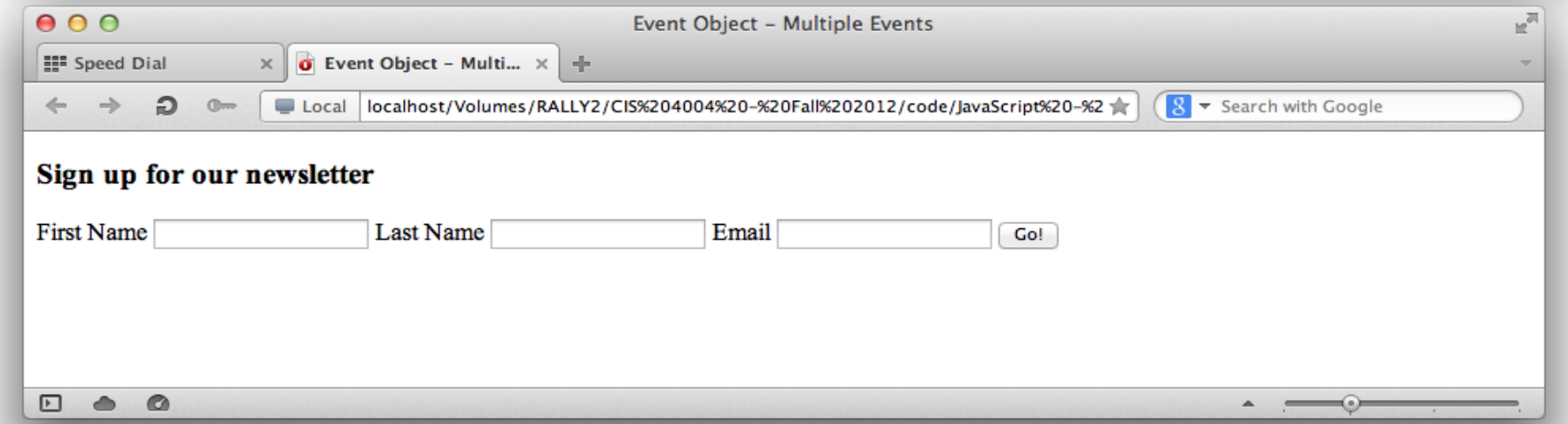

*K:\CIS 4004 - Fall 2012\code\JavaScript - Part 5\hilite_field_basic7.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
schedule.dat utlxplan.sql QRYOPTDATA.SQL notes.html hilite_field_basic6.html hilite_field_basic7.html
39 }
40 </script>
41 </head>
42
43 <body>
44
45 <h3>Sign up for our newsletter</h3>
46 <form id="email_form" action="#" method="get">
47 <label for="first_name">First Name</label>
48 <input id="first_name" name="first_name" type="text" size="18" />
49 <label for="last_name">Last Name</label>
50 <input id="last_name" name="last_name" type="text" size="18" />
51 <label for="email">Email</label>
52 <input id="email" name="email" type="text" size="18" />
53
54 <input id="submit" type="submit" value="Go!" />
55 </f
56
57 </body>
58 </html>
59
Hyper Text M length : 1842
Local
localhost/K:/CIS%204004%20-%20Fall%202012/code/Ja
Search with Google
Sign up for our newsletter
First Name
Last Name
Email
Go!

Attaching Event Handlers To Multiple Fields

- Since there are now several different form elements, the "hook" into the DOM will need to be higher up at the form element's `id`, which is `email_form`.
- Once the parent element is established, the you can get at all the form's child elements within.
- To do this we'll need to modify the `setUpFieldEvents` function.
- In keeping with our technique of testing that what we're doing is correct, we'll first modify this function to simply tell us how many `input` tags (children) are present in the form.
- This part of the modified JavaScript is shown on the next page.


K:\CIS 4004 - Fall 2012\code\JavaScript - Part 5\hilite_field_basic6.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plu
schedule.dat utbxplan.sql QRYOPTDATA.SQL notes.html hilite_field_basic6.html hilite_field_basic7.html
18 function setUpFieldEvents() {
19 var emailField=document.getElementById("email"); // get the field
20 addEvent(emailField, 'focus', checkHighlight); // add focus event
21 addEvent(emailField, 'blur', checkHighlight); // add blur event
22 }
23
Hyp length : 1391 lines : 50 Ln : 4 Col : 56 Sel : 0
Previous version – hook into DOM is the email id.
K:\CIS 4004 - Fall 2012\code\JavaScript - Part 5\hilite_field_basic7.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ? X
schedule.dat utbxplan.sql QRYOPTDATA.SQL notes.html hilite_field_basic6.html hilite_field_basic7.html
18
19 function setUpFieldEvents() {
20 var emailField=document.getElementById("email_form"); // get the field
21 var theInputs=document.getElementsByTagName("input");
22 var inputCount=theInputs.length;
23 alert(inputCount);
24 }
Hyp length : 1654 lines : 56 Ln : 24 Col : 1 Sel : 0 UNIX ISO 8859-1 INS
New version – hook into DOM is higher up the tree at the form element


Event Object – Multiple Events
Speed Dial
Event Object – Multi...
Local
localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%2
Search with Google
Sign up for our newsletter
First Name
Last Na
<localhost>
The number of input elements in this form is: 4
Stop executing scripts on this page
OK


Event Object – Multiple Events
Speed Dial
Event Object – Multi...
Local
localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%2
Search with Google
Sign up for our newsletter
First Name
Last Name
Email
Go!

Attaching Event Handlers To Multiple Fields

- Notice that the alert window informs us that there are 4 input elements in this form. Does this surprise you?
- There are four input elements because the submit button is also an input element, it simply has a different type ("submit") than the other input elements ("text").
- I don't want to apply the highlighting styles to the button element. Without step-by-step testing such as this it would be easy to overlook this sort of case and this might have induced a weird "bug" into the code that would have changed the background color of the button every time it was clicked.
- Now, I'm aware of this and will worry about filtering out the button a bit later. For now, we'll loop through all of the input fields and apply the event listeners to each of them.

hilite_field_basic8.html — Edited

```
        window.onload=setUpFieldEvents;

        function setUpFieldEvents() {
                var emailField=document.getElementById("email_form"); // get the field
                var theInputs=document.getElementsByTagName("input");
                var inputCount=theInputs.length;
                for (i=0; i < inputCount; i++) {
                   addEvent(theInputs[i], 'focus', checkHighlight); // add focus event
                   addEvent(theInputs[i], 'blur', checkHighlight); //  add blur event
                }
        }
        function checkHighlight(e) {
                var evt = e || window.event; // sets evt variable to either W3C or
Microsoft event obj
                var evtTarget = evt.target || evt.srcElement; // gets the target of the
event
                switch (evt.type) { // type is the same name in both objs
                        case "focus":
                                evtTarget.style.backgroundColor="#6F3";
                                break;
                        case "blur":
                                evtTarget.style.backgroundColor="";
                                break;
                }
        }
        </script>
</head>
```


An Aside – JavaScript Processing

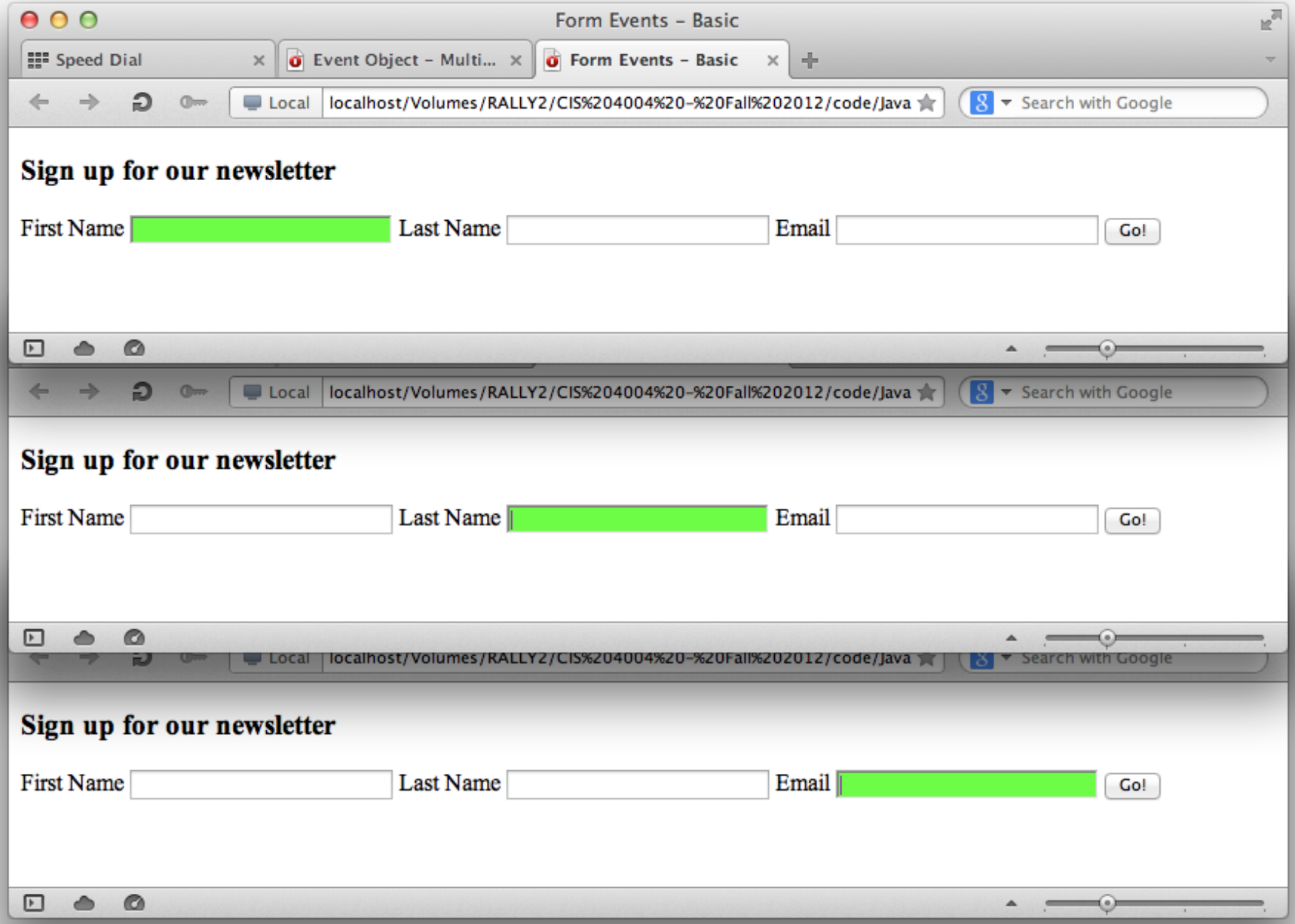
- I noticed several of you in your last program did something like the following (a modified version of the script on the previous page).

Untitled — Edited

```
function setUpFieldEvents() {
                var emailField=document.getElementById("email_form"); // get the field
                var theInputs=document.getElementsByTagName("input");
                for (i=0; i < theInputs.length; i++) {
                   addEvent(theInputs[i], 'focus', checkHighlight); // add focus event
                   addEvent(theInputs[i], 'blur', checkHighlight); //  add blur event
                }
        }
```

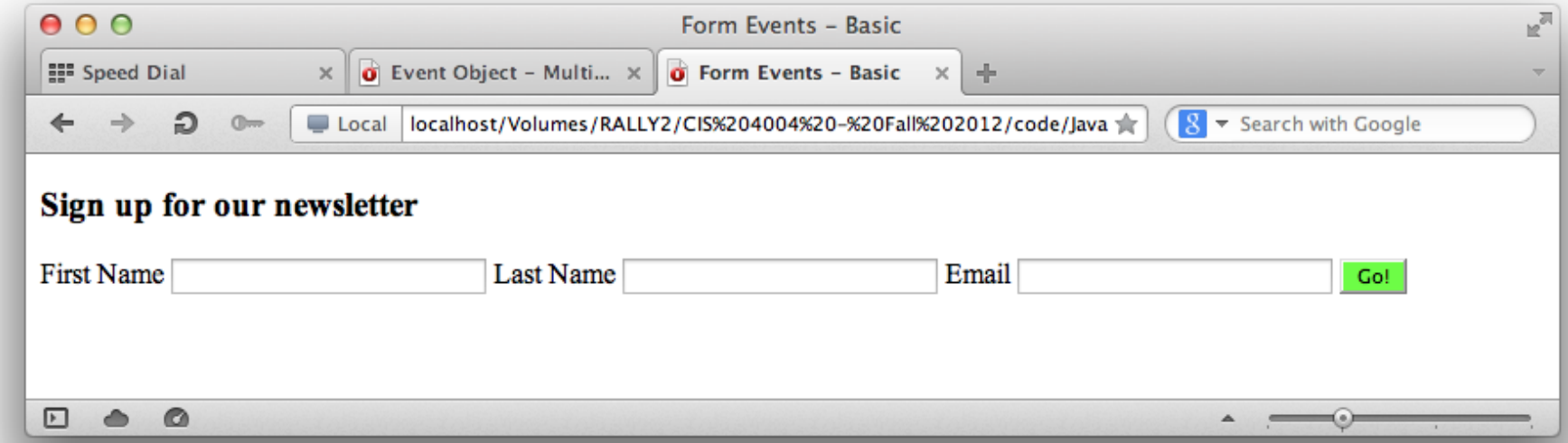
- This version is less efficient than the one on the previous page because JavaScript must recalculate the length of `theInputs` node list every time the loop executes. Counting items in arrays and node lists is a relatively slow process in JavaScript. It isn't such a big deal with a few items, but if you're looping over a big data set or hundreds of table rows, the wasted time can add up.

- Its always good practice to get the number of items once and store it in a variable that you can use as the loop count.


Form Events - Basic
Speed Dial
Event Object - Multi...
Form Events - Basic
Local
localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/Java
Search with Google
Sign up for our newsletter
First Name
Last Name
Email
Go!
Sign up for our newsletter
First Name
Last Name
Email
Go!
Sign up for our newsletter
First Name
Last Name
Email
Go!

Attaching Event Handlers To Multiple Fields

- Returning to our original problem. Notice in the screen shot below that the problem we identified with the submit button is indeed a problem. We don't want the submit button to be highlighted when the cursor moves over the button. We want that behavior to apply only to the form elements that the user is entering data.

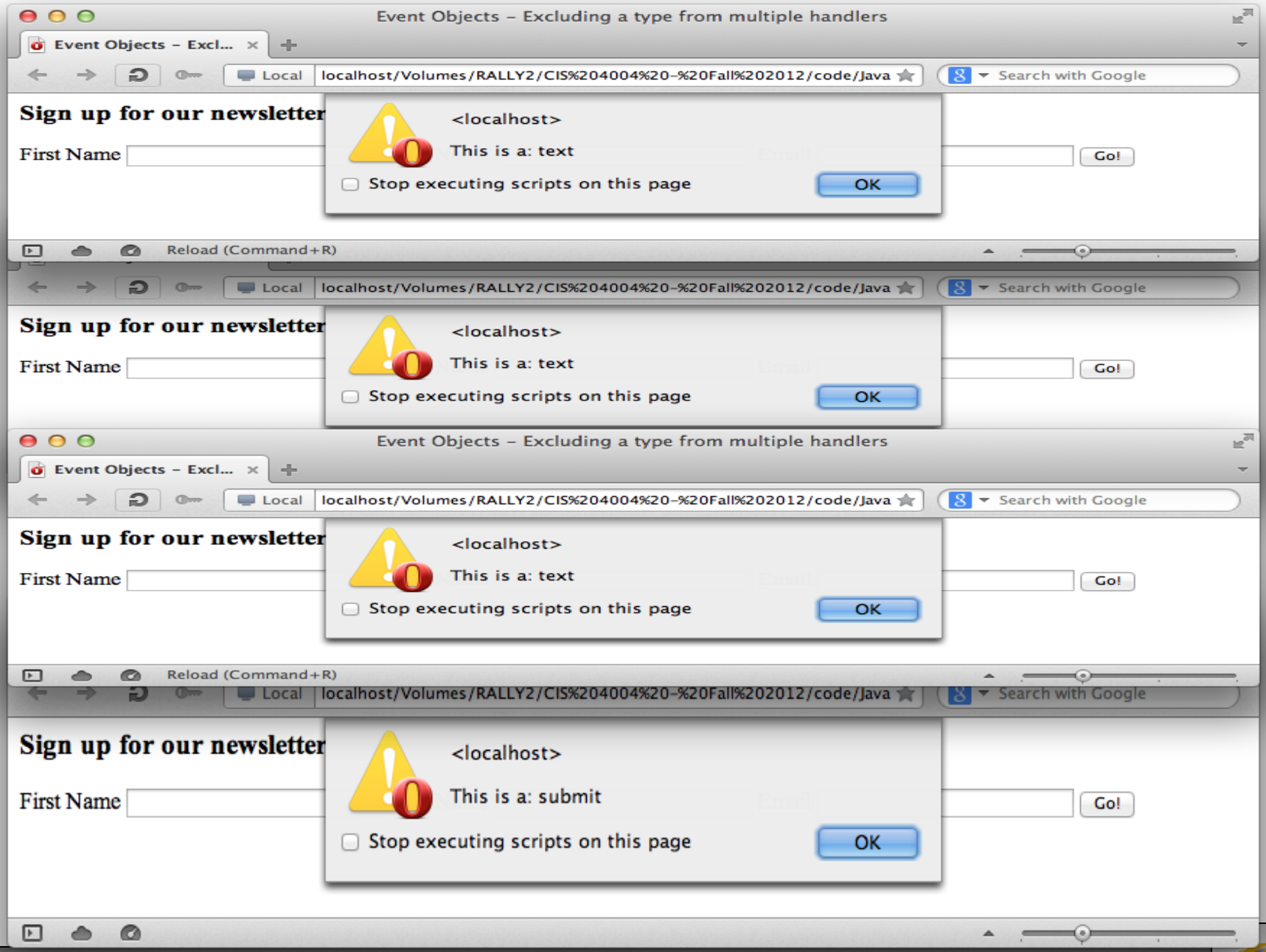


Attaching Event Handlers To Multiple Fields

- What makes the submit button different from the text inputs is that its type attribute is submit and not text.
- First we'll create a simple if statement filter based on this difference to identify and exclude the type submit from our highlighting.
- As before, we'll add alert pop-up windows to help us test the code.
- The code is shown on the next page and an illustration of it running on the following page.

hilite_field_basic9.html

```
    obj['e'+type+fn] = fn;
    obj[type+fn] = function(){obj['e'+type+fn]( window.event );}
    obj.attachEvent( 'on'+type, obj[type+fn] );
  } else
    obj.addEventListener(type, fn, false);
}

window.onload=setUpFieldEvents;

function setUpFieldEvents() {
        var emailField=document.getElementById("email_form"); // get the field
        var theInputs=document.getElementsByTagName("input");
        var inputCount=theInputs.length;
        for (i=0; i < inputCount; i++) {
                alert ("This is a: " + theInputs[i].getAttribute("type")); //testing
alerts
                addEvent(theInputs[i], 'focus', checkHighlight); // add focus event
                addEvent(theInputs[i], 'blur', checkHighlight);  //  add blur event
        }
}
function checkHighlight(e) {
        var evt = e || window.event; // sets evt variable to either W3C or Microsoft event
obj
        var evtTarget = evt.target || evt.srcElement; // gets the target of the event
        switch (evt.type) { // type is the same name in both objs
                case "focus":
                        evtTarget.style.backgroundColor="#6F3";
                        break;
                case "blur":
```

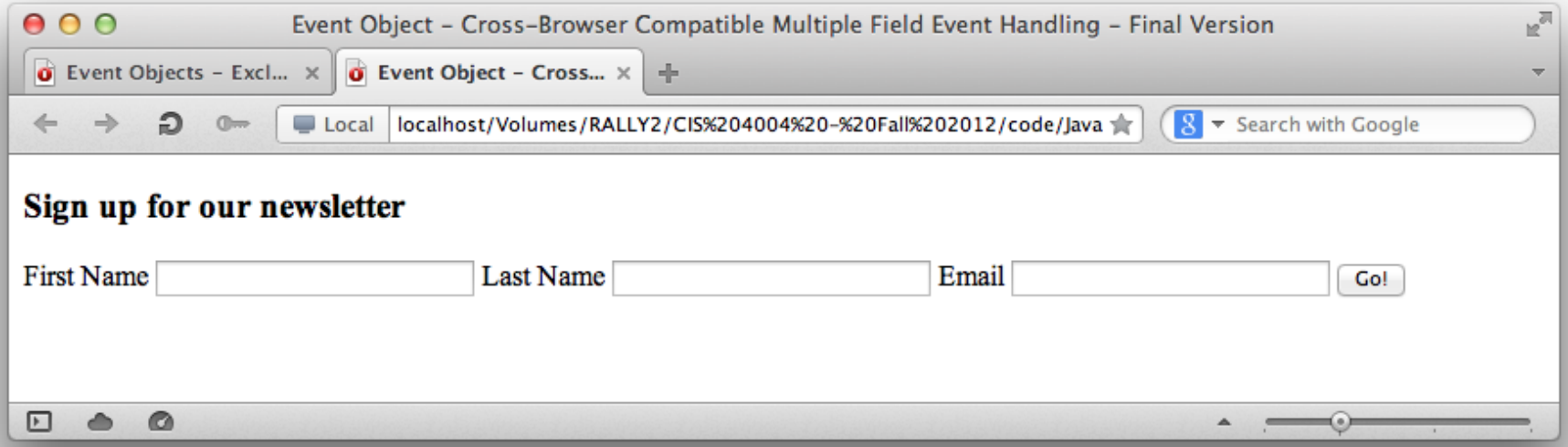


Event Objects – Excluding a type from multiple handlers
Event Objects – Excl...
Local
localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/Java
Search with Google
Sign up for our newsletter
First Name
Go!
<localhost>
This is a: text
Stop executing scripts on this page
OK
Reload (Command+R)
Event Objects – Excluding a type from multiple handlers
Event Objects – Excl...
Local
localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/Java
Search with Google
Sign up for our newsletter
First Name
Go!
<localhost>
This is a: text
Stop executing scripts on this page
OK
Reload (Command+R)
Local
localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/Java
Search with Google
Sign up for our newsletter
First Name
Go!
<localhost>
This is a: submit
Stop executing scripts on this page
OK

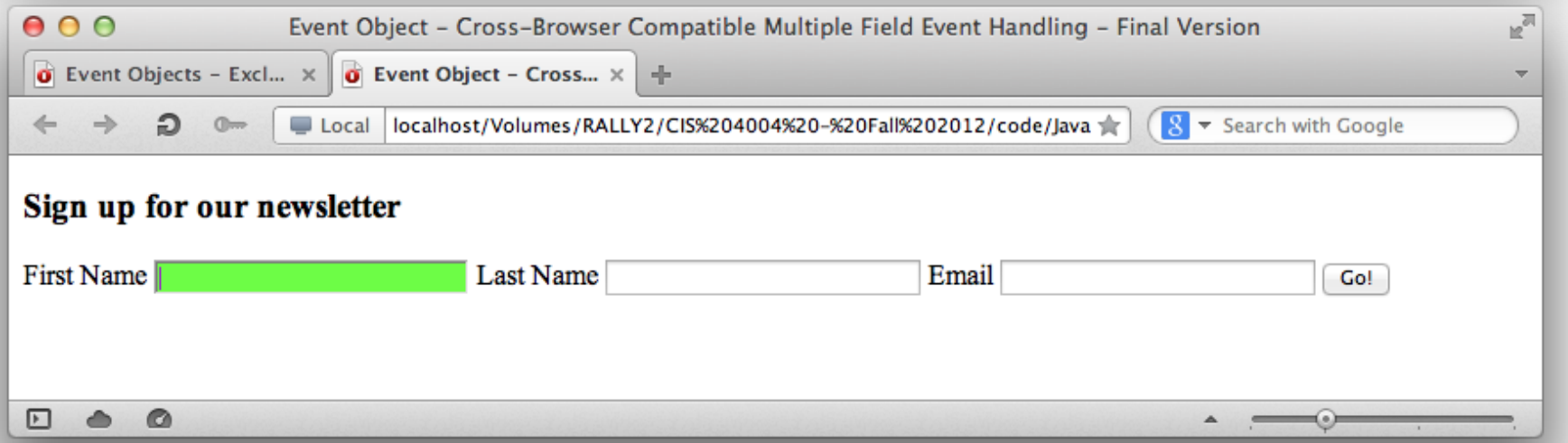
Attaching Event Handlers To Multiple Fields

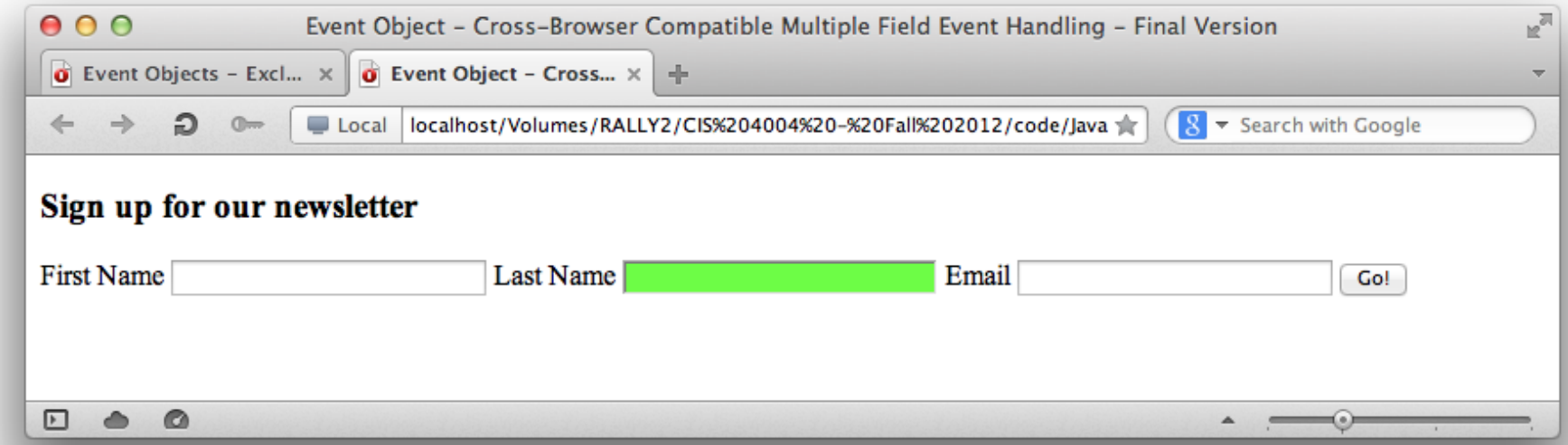
- Now that we know that we can distinguish between the submit input type and the text input type, all that is left to do is work this test code into the JavaScript to be inside the for loop driving through the input fields.
- This will produce the final version of this markup/JavaScript into a complete working example that will attach event handlers to multiple elements. The markup is also cross-browser compatible due to our technique of being able to distinguish between the W3C event object model and the Microsoft event object model.
- The complete markup/JavaScript file is available on the course webpage for you to download and experiment with. The following page illustrates the final version of the relevant portion of the JavaScript and the subsequent pages illustrate its rendering.

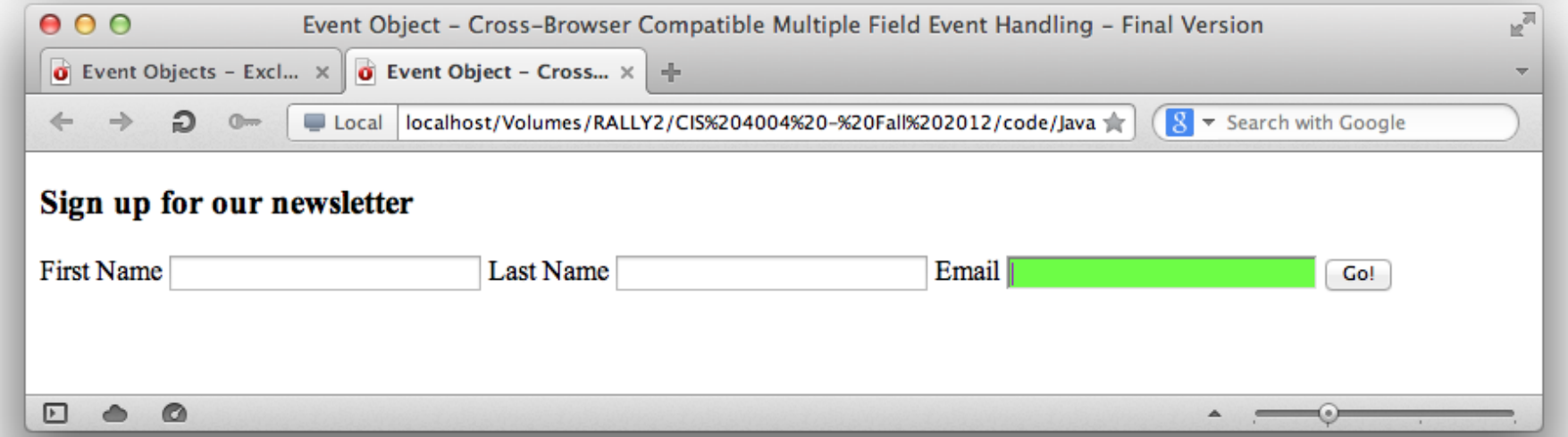
hilite_field_basic10.html — Edited

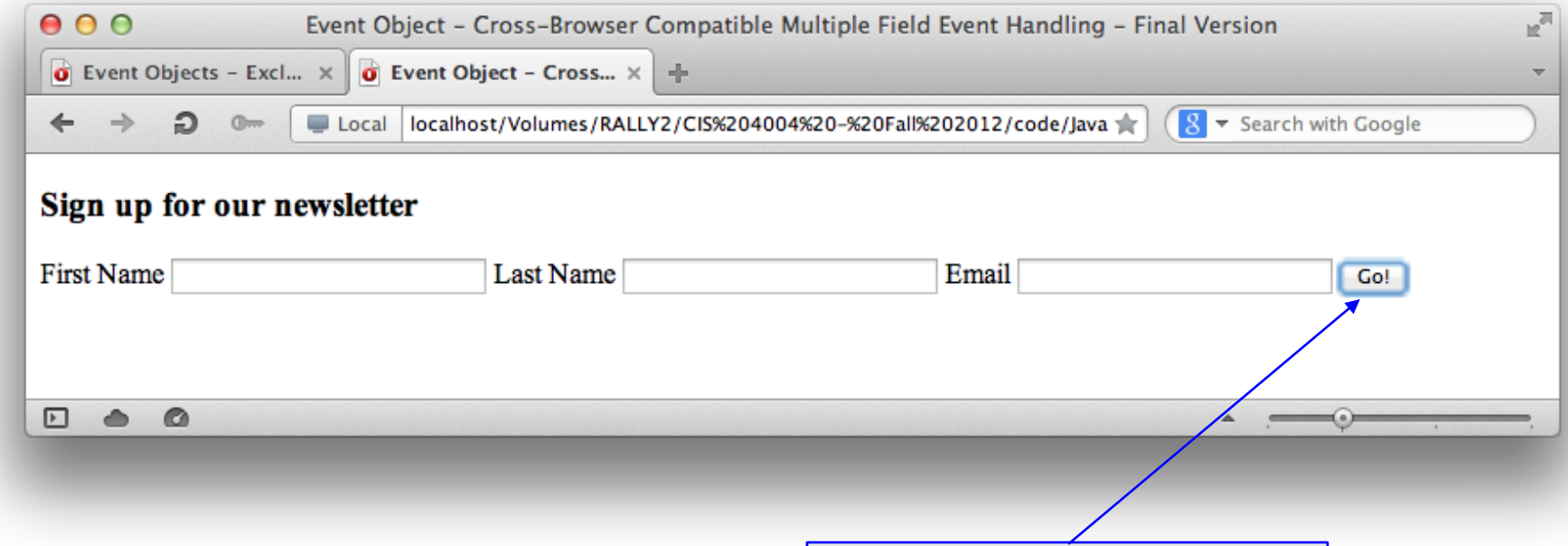
```
}

window.onload=setUpFieldEvents;

function setUpFieldEvents() {
        var emailField=document.getElementById("email_form"); // get the field
        var theInputs=document.getElementsByTagName("input");
        var inputCount=theInputs.length;
//alert(theInputs[0].getAttribute("type"));
        for (i=0; i < inputCount; i++) {
                var theInputType=theInputs[i].getAttribute("type");
                if (theInputType==="text") {
                        addEvent(theInputs[i], 'focus', checkHighlight); // add focus
event
                        addEvent(theInputs[i], 'blur', checkHighlight); //  add blur event
                }
        }
}
function checkHighlight(e) {
        var evt = e || window.event; // picks either W3C or Microsoft event obj
        var evtTarget = evt.target || evt.srcElement; // gets the target of the event
        switch (evt.type) { // type is the same name in both objs
                case "focus":
                        evtTarget.style.backgroundColor="#6F3";
                        break;
                case "blur":
                        evtTarget.style.backgroundColor="";
                        break;
        }
}
```



Event Object – Cross-Browser Compatible Multiple Field Event Handling – Final Version
Event Objects – Excl...
Event Object – Cross...
Local
localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/Java
Search with Google
Sign up for our newsletter
First Name
Last Name
Email
Go!

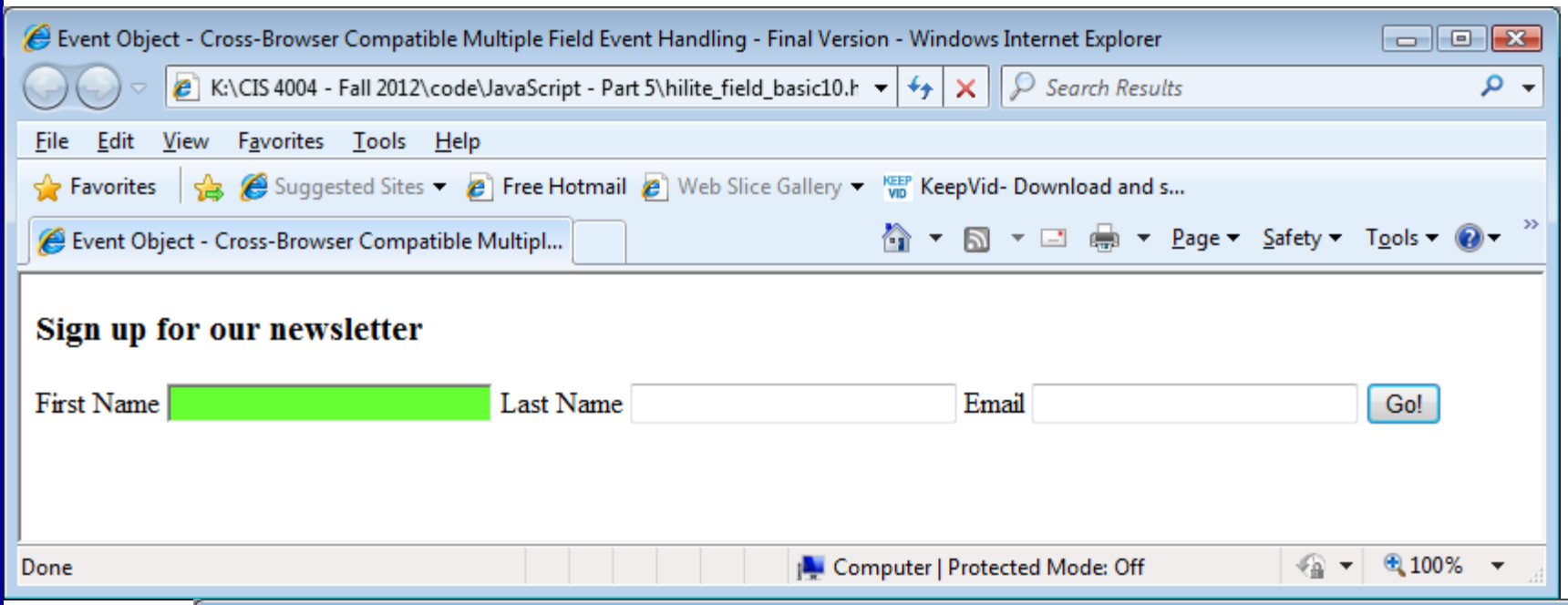

Event Object – Cross-Browser Compatible Multiple Field Event Handling – Final Version
Event Objects – Excl...
Event Object – Cross...
Local
localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/Java
Search with Google
Sign up for our newsletter
First Name
Last Name
Email
Go!

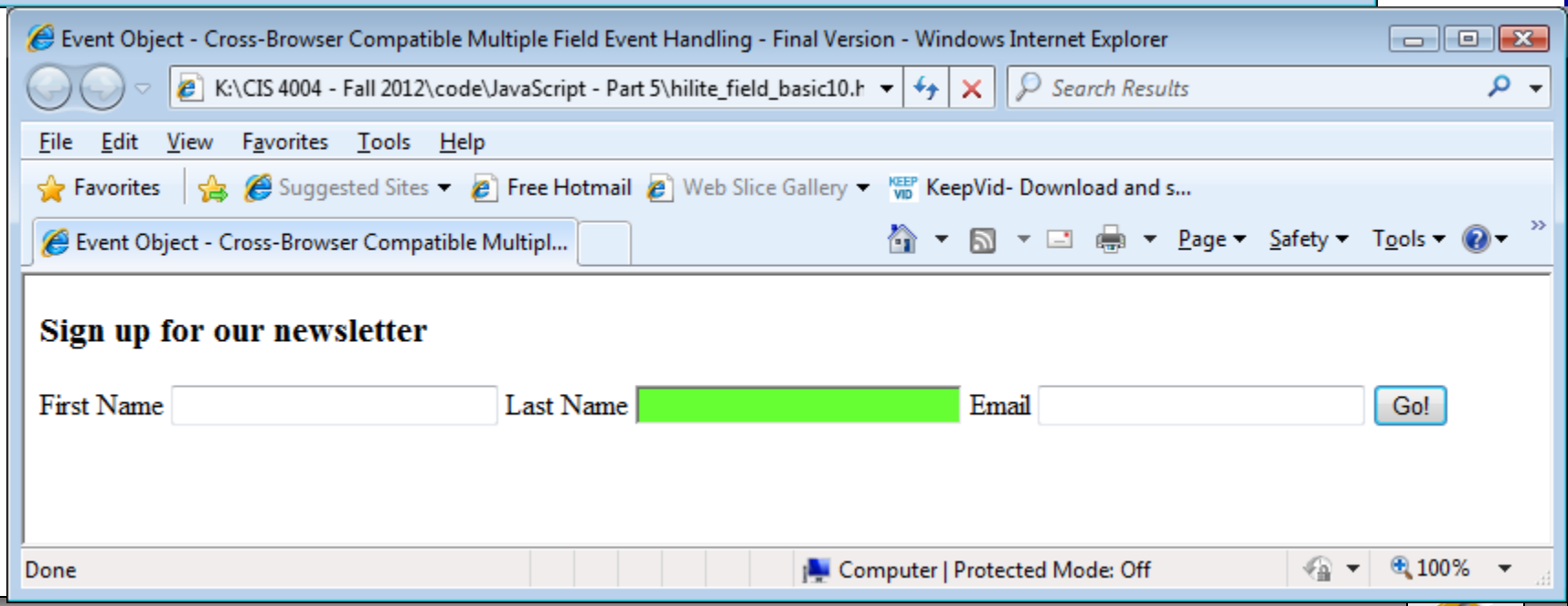

Event Object – Cross-Browser Compatible Multiple Field Event Handling – Final Version
Event Objects – Excl...
Event Object – Cross...
Local
localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/Java
Search with Google
Sign up for our newsletter
First Name
Last Name
Email
Go!

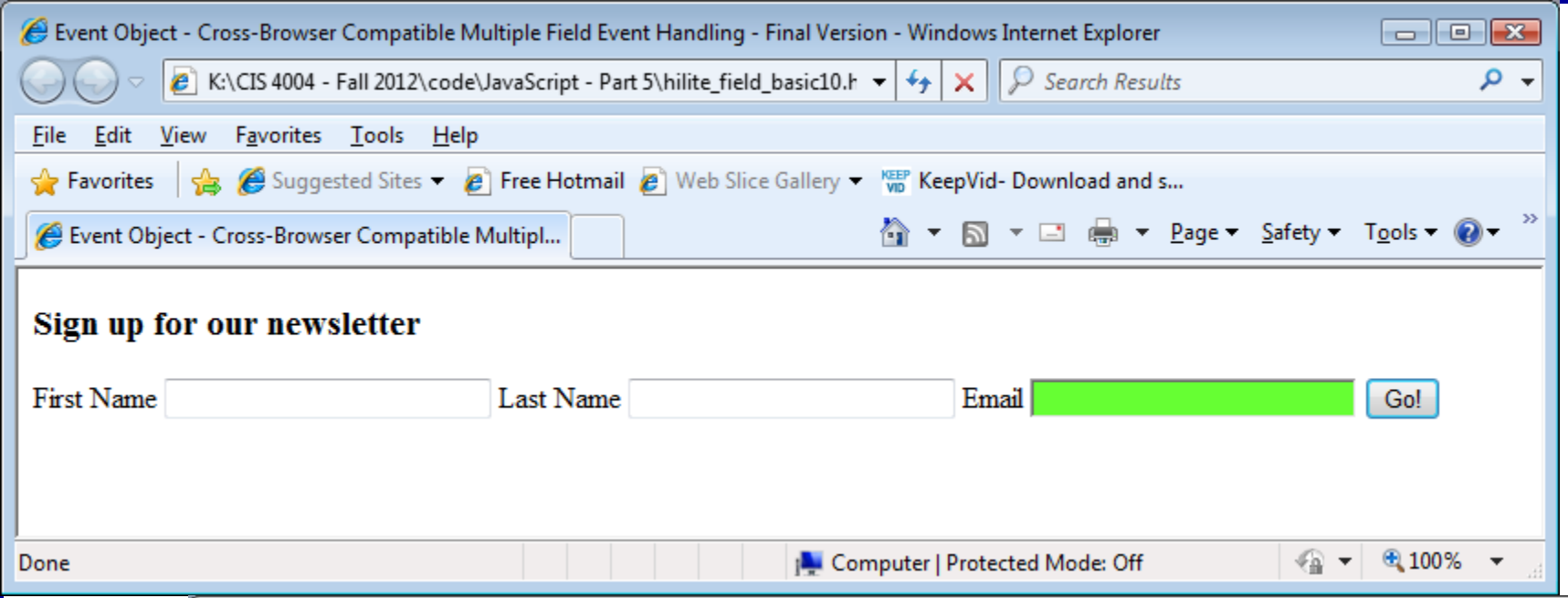

Event Object – Cross-Browser Compatible Multiple Field Event Handling – Final Version
Event Objects – Excl...
Event Object – Cross...
Local
localhost/Volumes/RALLY2/CIS%204004%20-%20Fall%202012/code/Java
Search with Google
Sign up for our newsletter
First Name
Last Name
Email
Go!

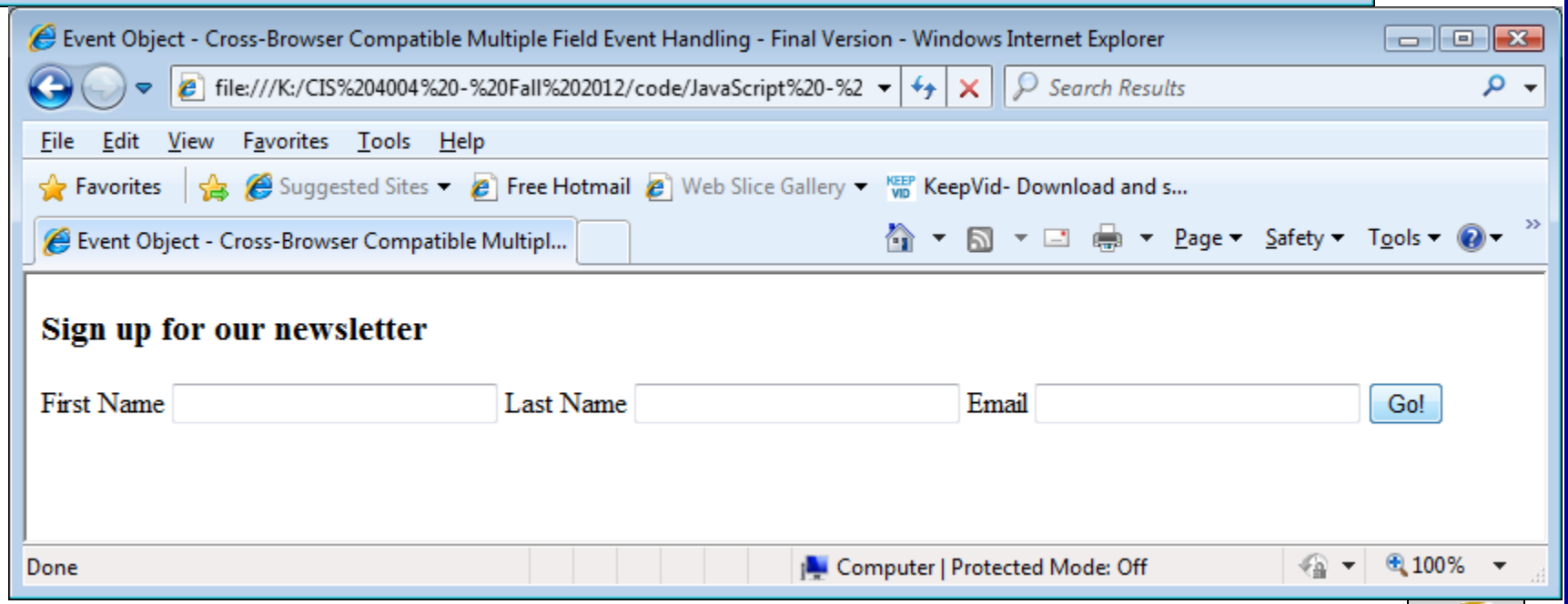


Notice that the submit button does not have the highlighting applied to it. It now renders properly by the JavaScript ignoring it.


Event Object - Cross-Browser Compatible Multiple Field Event Handling - Final Version - Windows Internet Explorer
K:\CIS 4004 - Fall 2012\code\JavaScript - Part 5\hilite_field_basic10.h
Search Results
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Free Hotmail
Web Slice Gallery
KeepVid- Download and s...
Event Object - Cross-Browser Compatible Multipl...
Page
Safety
Tools
Sign up for our newsletter
First Name
Last Name
Email
Go!
Done
Computer | Protected Mode: Off
100%


Event Object - Cross-Browser Compatible Multiple Field Event Handling - Final Version - Windows Internet Explorer
K:\CIS 4004 - Fall 2012\code\JavaScript - Part 5\hilite_field_basic10.h
Search Results
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Free Hotmail
Web Slice Gallery
KeepVid- Download and s...
Event Object - Cross-Browser Compatible Multipl...
Page
Safety
Tools
Sign up for our newsletter
First Name
Last Name
Email
Go!
Done
Computer | Protected Mode: Off
100%


Event Object - Cross-Browser Compatible Multiple Field Event Handling - Final Version - Windows Internet Explorer
K:\CIS 4004 - Fall 2012\code\JavaScript - Part 5\hilite_field_basic10.h
Search Results
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Free Hotmail
Web Slice Gallery
KeepVid- Download and s...
Event Object - Cross-Browser Compatible Multipl...
Page
Safety
Tools
Sign up for our newsletter
First Name
Last Name
Email
Go!
Done
Computer | Protected Mode: Off
100%


Event Object - Cross-Browser Compatible Multiple Field Event Handling - Final Version - Windows Internet Explorer
file:///K:/CIS%204004%20-%20Fall%202012/code/JavaScript%20-%2
Search Results
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Free Hotmail
Web Slice Gallery
KeepVid- Download and s...
Event Object - Cross-Browser Compatible Multipl...
Page
Safety
Tools
Sign up for our newsletter
First Name
Last Name
Email
Go!
Done
Computer | Protected Mode: Off
100%