CS 22A JavaScript for Programmers

Today (Day 12)

- Announcements
- Events & Event Handlers
- ICE02: Objects
- Lab

Midterm Survey

- Get your quick and easy 5 points!
- Problems? Suggestions? You can also use the anonymous contact form linked from the syllabus.

Events & Event Handlers

- For Plain Old JavaScript

What is an Event?

- An **event** is something that happens when the viewer of the page performs some sort of action, such as clicking a mouse button, clicking a button on the page, changing the contents of a form element, or moving the mouse over a link on the page.
  - Events can also occur simply by the page loading or other similar actions.
- JavaScript events enables your code to react to an action by the user so you can make scripts that are interactive, and more useful to you and to the user.

What is an Event Handler?

- An **event handler** is predefined JavaScript property of an object (which is usually an element in the document). Programmers bind that property to a function that is used to handle an event on a web page. This is called **listening to an event**, **binding**, or **attaching**.
  - Event handler functions are also called **callbacks** because they are called-back/run when the particular event happens
- When events occur, you are able to use JavaScript event handlers to listen to them and then react to perform a specific task or set of tasks.

Event Handler Example

```
var pageLoadedHandler = function(){
  alert("I'm alive!");
};
window.onload = pageLoadedHandler;
```

After the programmer binds the window.onload to the pageLoadedHandler, whenever a page load event is generated, the pageLoadedHandler function is called.

Event Handler Example

```
var pageLoadedHandler = function(){
  alert("I'm alive!");
};
window.onload = pageLoadedHandler;
```

WATCH-IT!:

Sometimes when people say "event handler" they mean the event handler **property** or sometimes they mean the value of an event handler property AFTER it is bound, in other words the event handler **function**.

Why Event Handlers Are Useful

```
var pageLoadedHandler = function(){
  alert("I'm alive!");
};
window.onload = pageLoadedHandler;
```

Event handlers are useful because they enable you to gain access to the events that may occur on the page.

Understanding Event Handlers Locations and Uses

- To see how event handlers work, you need to know where you can place them in a document and how to use them to add JavaScript code for an event.
- Event handlers can be used in a number of locations.
  - They can be used directly within HTML elements by adding special attributes to those elements.
  - They can also be used within the <script> and </script> tags or in an external JavaScript file.

Using an Event Handler in an HTML Element (inline)

```
<input type="button" value="Click Me!"
onclick="alert('You clicked!');">
```

Using Event Handlers in the <script> Element

```
<input type="button" value="Click Me!" id="yo">

<script>
var myFunction = function(event){
        alert("Yo!");
}
var yoButton = document.getElementById("yo");
yoButton.onclick = myFunction;
</script>
```

The **event object** contains the detected event's properties and methods.

Using Event Handlers in the <script> Element

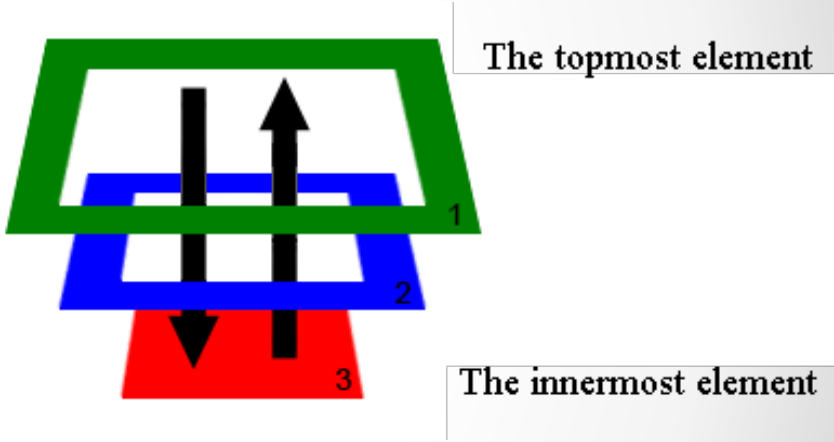
```
<input type="button" value="Click Me!" id="yo">

<script>
var myFunction = function(event){
	alert("Yo!");
}
document.getElementById("yo").onclick = myFunction;
</script>
```

The Three Phases of Event Dispatch

Capturing – The event travels downward (or inward) from the document object to the target object
Target – The event triggers on the target object
Bubbling – The event targets upward (or outward) from the target object to the document object
NOTE: Some versions of Internet Explorer don't support the capturing phase, so if you want to support Internet Explorer browser users you should limit your event handlers to the target and bubbling phases.

```
<div class="d1">1  <!-- topmost -->
 <div class="d2">2
  <div class="d3">3 <!-- innermost ->
   Click me!
  </div>
 </div>
</div>
```



For more info see:
http://javascript.info/tutorial/bubbling-and-capturing

The **addEventListener()** Method: How-to bind **any** event using plain old JS

The **addEventListener()** method allows you to specify an event, a function to execute for the event, and a value of true or false depending on how you want the event handler function to be executed in the *capturing* (true) or *bubbling* (false) phase. The general format looks like this:

```
element.addEventListener('event_type', listener_function_name, useCapture_true_or_false);
```

For example: if you want a button to open a link when clicked you could write JavaScript code like this:

```
var page1 = "http://www.yahoo.com";
var b1 = document.getElementById("btn1");
b1.addEventListener('click', function(event) {
  console.log(event.type); // event.type should be equal to 'click'
  window.location = page1;
}, false);
```

The **event object** contains the detected event's properties and methods.

For example, the **type** property of an event object contains the type of event that occurred.

The event Object's Contents

Property/Method	Description
bubbles	Whether or not the event bubbles
cancelable	Whether or not the default action of the event can be canceled
cancelBubble	Cancels event bubbling when set to false (Internet Explorer)
currentTarget	The element that is currently handling the event
defaultPrevented	Whether or not preventDefault() has been called
detail	Additional information about the event
eventPhase	The phase in which the event handler was called: 1 = capturing, 2 = at target, 3 = bubbling
preventDefault()	Prevents the default action of the event from occurring
returnValue	Prevents the default action of the event from occurring (Internet Explorer)
srcElement	The element that is the target of the event (Internet Explorer)
stopImmediatePropagation()	Ends all capturing and bubbling on the event, and stops other event handlers from being called
stopPropagation()	Ends all capturing and bubbling on the event
target	The element that is the target of the event
trusted	Whether or not the event was initiated by the browser or the programmer
type	The event type (for example, click, mouseover, and so on)
view	The window object where the event happened

NOTE: Some of these events, such as the copy event, will only work with certain browsers (which may need to be running in their latest versions). There are also events that work only in Internet Explorer or that are not necessarily cross browser as of yet (see https://www.quirksmode.org/dom/events/ & http://msdn.microsoft.com/en-us/library/ms533051(VS.85).aspx) .

Event	Event Handler	Event Trigger
Abort	onabort	An image is stopped from loading before loading has completed
Blur	onblur	Viewer removes focus from an element
Change	onchange	Viewer changes the contents of a form element
Click	onclick	Viewer clicks an element
ContextMenu	oncontextmenu	Viewer opens the context menu
Copy	oncopy	Viewer uses the copy command on part of a page
Cut	oncut	Viewer uses the cut command on part of a page
Dblclick	ondblclick	Viewer double-clicks the mouse
Error	onerror	Viewer's browser gets a JavaScript error or an image that does not exist
Focus	onfocus	Viewer gives focus to an element
Keydown	onkeydown	Viewer presses down a key on the keyboard
Keypress	onkeypress	Viewer presses a key on the keyboard, and releases or holds the key down
Keyup	onkeyup	Viewer releases a key on the keyboard
Load	onload	Web page finishes loading
Mousedown	onmousedown	Viewer presses the mouse button
Mousemove	onmousemove	Viewer moves the mouse (moves the cursor)
Mouseout	onmouseout	Viewer moves the mouse away from an element
Mouseover	onmouseover	Viewer moves the mouse over an element
Mouseup	onmouseup	Viewer releases the mouse button
Paste	onpaste	Viewer uses the paste command on part of the page
Reset	onreset	Viewer resets a form on the page
Resize	onresize	A window is resized
Scroll	onscroll	Viewer scrolls an area which is scrollable
Select	onselect	User makes a selection
Submit	onsubmit	Viewer submits a form on the page
Unload	onunload	Viewer leaves the current page

To Do:Exercise02

- Instructions in our Canvas site
- Watch the demonstration of how Exercise02 should work in the **FirefoxDeveloperEdition** browser in a video in the Day12 module