CS472 WAP

Unobtrusive JavaScript

JavaScript Programming Environment

Maharishi International University Fairfield, Iowa © 2020

Main Point Preview

- Unobtrusive JavaScript promotes separation of web page content into 3 different concerns: content (HTML), presentation (CSS), and behavior(JS) (ala MVC, knower, known, process of knowing)

- JavaScript code runs when the page loads it. Event handlers cannot be assigned until the target elements are loaded. In intelligent systems certain events must happen in a particular order. Creative intelligence proceeds in an orderly sequential manner.

Unobtrusive JavaScript

- JavaScript event code seen yesterday was *obtrusive*, in the HTML; this is bad style
- now we'll see how to write *unobtrusive* JavaScript code
  - HTML with minimal JavaScript inside
  - uses the DOM to attach and execute all JavaScript functions
- allows separation of web site into 3 major categories:
  - **content** (HTML) - what is it?
  - **presentation** (CSS) - how does it look?
  - **behavior** (JavaScript) - how does it respond to user interaction?

Obtrusive event handlers(bad)

```
<button onclick="okayClick();">OK</button>

function okayClick() {
 alert("booyah");
}
```

- this is bad style (HTML is cluttered with JS code)
- goal: remove all JavaScript code from the HTML body

Unobtrusive JavaScript

```
// where element is a DOM element object
element.onevent = function;
```

```
<button id="ok">OK</button>
var okButton = document.getElementById("ok");
okButton.onclick = okayClick;
```

- it is legal to attach event handlers to elements' DOM objects in your JavaScript code
  - notice that you do **not** put parentheses after the function's name
- this is better style than attaching them in the HTML
- Where should we put the above code?

Linking to a JavaScript file: script

- JS code can be placed directly in the HTML file's body or head (like CSS)
  - but this is bad style (should separate content, presentation, and behavior)
- script tag should be placed in HTML page's head
- script code should be stored in a separate .js file (like CSS)

```
<script src="example.js" ></script>
```

When does my code run?

```
<html>
 <head>
 <script src="myfile.js"></script> </head>
 <body> ... </body> </html>

// global code
var x = 3;
function f(n) { return n + 1; }
function g(n) { return n - 1; }
x = f(x);
```

- your file's JS code runs the moment the browser loads the script tag
  - any variables are declared immediately
  - any functions are declared but not called, unless your global code explicitly calls them
- at this point in time, the browser has not yet read your page's body
  - none of the DOM objects for tags on the page have been created yet

See example: lecture06_examples/runjs.html

A failed attempt at being unobtrusive

```
<html>
 <head>
 <script src="myfile.js" type="text/javascript"></script> </head>
 <body> ... </body> </html>
<div><button id="ok">OK</button></div>

// code in myfile.js
document.getElementById("ok").onclick = okayClick; // error: cannot set property
 onclick of null
```

- problem: myfile.js code runs the moment the script is loaded
- script in head is processed before page's body has loaded
  - no elements are available yet or can be accessed yet via the DOM
  - See lecture06_examples/failedattempt.html *
- we need a way to attach the handler after the page has loaded...

The window.onload event

```
// this will run once the page has finished loading
function functionName() {
 element.event = functionName;
 element.event = functionName;
 ...
}
window.onload = functionName; // global code
```

- we want to attach our event handlers right after the page is done loading
  - there is a global event called window.onload event that occurs at that moment
- in window.onload handler we attach all the other handlers to run when events occur

See example: lecture06_examples/unobtrusivehandler1.html (does it work?) *

Common unobtrusive JS errors

- many students mistakenly write () when attaching the handler

```
window.onload = pageLoad();
window.onload = pageLoad;
okButton.onclick = okayClick();
okButton.onclick = okayClick;
```

  - IMPORTANT FUNDAMENTAL CONCEPT !!!
    - Function reference versus evaluation

- event names are all lowercase, not capitalized like most variables

```
window.onLoad = pageLoad;
window.onload = pageLoad;
```

Anonymous functions

```
function(parameters) {
  statements;
}
```

- JavaScript allows you to declare **anonymous functions**
- creates a function without giving it a name
- can be stored as a variable, attached as an event handler, etc.
- Important in JavaScript because of event handling nature and minimizing namespace clutter

Anonymous function example

```
window.onload = function() {
 var okButton = document.getElementById("ok");
 okButton.onclick = okayClick;
};
function okayClick() {
 alert("booyah");
}
```

or the following is also legal (though harder to read):

```
window.onload = function() {
 var okButton = document.getElementById("ok");
 okButton.onclick = function() {
  alert("booyah");
 }; };
```

- this is probably only place will ever need okayClick function,
  - so JS convention is to use anonymous function style and not clutter up namespace

Unobtrusive styling

```
function okayClick() {
 this.style.color = "red";
 this.className = "highlighted";
}
.highlighted { color: red; }
```

- Well-written JavaScript code should contain as little CSS as possible
- use JS to set CSS classes/IDs on elements
- define the styles of those classes/IDs in your CSS file
- What about cssZenGarden?

Main Point

- Unobtrusive JavaScript promotes separation of web page content into 3 different concerns: content (HTML), presentation (CSS), and behavior(JS) (ala MVC, knower, known, process of knowing)

- JavaScript code runs when the page loads it. Event handlers cannot be assigned until the target elements are loaded. In intelligent systems certain events must happen in a particular order. Creative intelligence proceeds in an orderly sequential manner.


MAHARISHI
HIGHER EDUCATION
FOR HIGHER CONSCIOUSNESS
INTERNATIONAL UNIVERSITY