

JavaScript Workshop

Unobtrusive JavaScript

What is Unobtrusive JavaScript?

- Separation of structure and behavior. Scripts should not be embedded inside the HTML (Structure layer) and not define any look and feel (Presentation layer), leave that to the CSS.
- Using W3C DOM standards, not browser specific dialects. Scripts should never use `document.layers` or `document.all`.
- Graceful degradation. Scripts should still work for older browsers that don't support a sophisticated interface. We should only use JavaScript to enhance a functionality that is already given, and not rely on it.

Separation of structure and behavior

- Apart from keeping your JavaScript in separate files, the next step will be removing inline event calls in your web pages.
- Events were first discussed last season and used with topics such as the Image and Form objects. All of our previous examples using events were embedded within the HTML tags.

```
<a href = "#" onclick = "JavaScript: function()"; >click me</a>
```

- This mixed the Structure layer (HTML) and the Behavior layer (JavaScript), which should be kept separate to make it easier to maintain, standardize, and be accessible if JavaScript is not available (old browser, firewall restrictions, screen readers).

Step 1 - Register an event handler

- Allows you to attach an event to an element without having the event call in your HTML. This registers the event with the custom JavaScript function you create.

element.event = function name(); // then define your function code

- Another way (or shorthand) would be to create an anonymous function which defines the function code without a function name. This technique is used in many book examples since it is more efficient in many ways.

element.event = function() { ---> function has no name

enter your function code here...

}

Step 2 - Create a "hook" for attached events

- A class or id needs to be used to "hook" the behavior to the markup without having to intermingle inline function calls.

```
<a class = "popup" href = "http://www.pacsnet.org">PACS</a>
```

```
var links = document.getElementsByTagName("a");
```

```
for (var i=0; i < links.length; i++) {
```

```
    if (links[i].getAttribute("class") == "popup") {
```

```
        links[i].onclick = function();
```

```
        return false;
```

```
    }
```

```
}
```

Step 3 - Execute function after DOM loads

- Now that we have registered an event and hooked it to specific elements, we still need a way of running the function. The normal way of executing a function will not work because the code must be executed after the document has finished loading.
- This code can go in a .js file or in the <HEAD> section.

```
window.onload = function () {  
  
    firstFunction();  
    secondFunction();  
}
```

Note: An anonymous function must be used since the onload event will only run once.

Before DOM there was DHTML

- Before there was W3C DOM support in browsers (IE 4 and NS 4), Dynamic HTML allowed for increased user interaction on web pages. DHTML offered similar enhancements like DOM Scripting, but each browser required custom coding (browser sniffing, etc.) which led to its demise.
- With current browsers (IE 6, FF 1.x and Opera 7) offering W3C DOM compliancy, dynamic web pages have sprung back to life.

if (document.all || document.layers) ---> DHTML (browser detection)

if (document.getElementById) ---> DOM Scripting (DOM detection)

Graceful degradation

- We cannot control when viewers browser web pages.
 - 1) Using an older browser
 - 2) Has client-scripting disabled
 - 3) Employer has firewall with specific security settings
 - 4) User is using a screen reader
- Which is why your JavaScript should degrade gracefully so your web pages are still usable and accessible. Keeping the Behavior layer separate is the first step, but should also include checking for specific objects before accessing them.

if (!document.getElementById) return false;

- JavaScript Workshop website:
<http://www.javascriptworkshop.com>
- Any questions?
frank@javascriptworkshop.com - Frank Stepanski
- Recommended reading:
[DOM Scripting: Web Design with JavaScript](#)