中山大學
SUN YAT-SEN UNIVERSITY

# Lecture 15
# Advanced JavaScript and DOM

**SE-805 Web 2.0 Programming (supported by Google)**

http://my.ss.sysu.edu.cn/courses/web2.0/

**School of Software, Sun Yat-sen University**

# Important tools to have

*"Mozilla Firefox is a free and open source web browser descended from the Mozilla Application Suite, managed by the Mozilla Corporation. Firefox had 19.73% of the recorded usage share of web browsers as of August 2008, making it the second-most popular browser in current use worldwide."*

www.firefox.com


*"Firebug integrates with Firefox to put a wealth of web development tools at your fingertips while you browse. You can edit, debug, and monitor CSS, HTML, and JavaScript live in any web page."*

www.getfirebug.com


*"The Aptana Studio Community Edition provides a full-featured web development environment. The Community Edition represents the core pieces of the Aptana frameworks for editing, debugging, synchronization, and project management."*

www.aptana.com

# Outline

- **Global DOM Objects**
- Unobtrusive JavaScript
- JavaScript tips

# Global DOM Objects

- Every browsers' Javascript program can refer to the following global objects:

| Name | Description |
|---|---|
| document | current HTML page and its content |
| history | list of pages the user has visited |
| location | URL of the current HTML page |
| navigator | info about the web browser you are using |
| screen | info about the screen area occupied by the browser |
| window | the browser window |

# The window Object

- The entire browser window; the top-level object in DOM hierarchy
- Technically, all global code and variables become part of the window object
- Properties:
  - document, history, location, name
- Methods:
  - alert, confirm, prompt (popup boxes)
  - setInterval, setTimeout clearInterval, clearTimeout (timers)
  - open, close (popping up new browser windows)
  - blur, focus, moveBy, moveTo, print, resizeBy, resizeTo, scrollBy, scrollTo

# The document Object

- The current web page and the elements inside it
- Properties:
  - anchors, body, cookie, domain, forms, images, links, referrer, title, URL
- Methods:
  - getElementById
  - getElementsByName
  - getElementsByTagName
  - close, open, write, writeln
- complete list

# The location Object

- The URL of the current web page
- Properties:
  - host, hostname, href, pathname, port, protocol, search
- Methods:
  - assign, reload, replace
- Complete list

# The navigator Object

- Information about the web browser application
- Properties:
  - appName, appVersion, browserLanguage, cookieEnabled, platform, userAgent
  - complete list
- Some web programmers examine the navigator object to see what browser is being used, and write browser-specific scripts and hacks:

```
if (navigator.appName === "Microsoft Internet Explorer") { ...
```
JS

  - (This is poor style; you should not need to do this)

# The screen Object

- Information about the client's display screen
- Properties:
  - availHeight, availWidth, colorDepth, height, pixelDepth, width
  - complete list

# The history Object

- The list of sites the browser has visited in this window
- Properties:
  - length
- Methods:
  - back, forward, go
- Complete list
- Sometimes the browser won't let scripts view history properties, for security

# Outline

- Global DOM Objects
- **Unobtrusive JavaScript**
- JavaScript tips

# Unobtrusive JavaScript

- The JavaScript event code seen previously 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 id="ok" onclick="okayClick();">OK</button>
```
HTML

```
// called when OK button is clicked
function okayClick() {
  alert("booyah");
}
```
JS


output

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

# Attaching an Event Handler in JavaScript Code

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

```
$("ok").onclick = okayClick;
```
JS

OK

output

- 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?

# When does My Code Run?

```HTML
<head>
  <script src="myfile.js" type="text/javascript"></script>
</head>

<body> ... </body>
```

```JS
// 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

# A Failed Attempt at Being Unobtrusive

```HTML
<head>
  <script src="myfile.js" type="text/javascript"></script>
</head>

<body>
  <div><button id="ok">OK</button></div>
```

```JS
// global code
$("ok").onclick = okayClick;   // error: $("ok") is null
```

- Problem: global 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
- 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
```
JS

- 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

# An Unobtrusive Event Handler

```html
<!-- look Ma, no JavaScript! -->
<button id="ok">OK</button>
```
HTML

```js
// called when page loads; sets up event handlers
function pageLoad() {
  $("ok").onclick = okayClick;
}

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

window.onload = pageLoad;  // global code
```
JS


output

# Common Unobtrusive JS Errors

- Many students mistakenly write () when attaching the handler

```
window.onload = pageLoad();
window.onload = pageLoad;

okButton.onclick = okayClick();
okButton.onclick = okayClick;
```
JS

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

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

# Outline

- Global DOM Objects
- Unobtrusive JavaScript
- **JavaScript tips**

# Anonymous Functions

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

- JavaScript allows you to declare anonymous functions
- Quickly creates a function without giving it a name
- Can be stored as a variable, attached as an event handler, etc.

# Anonymous Function Example

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

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

JS


output

- Or the following is also legal (though harder to read and bad style):

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

JS

# The Keyword this

```
this.fieldName                          // access field
this.fieldName = value;                 // modify field

this.methodName(parameters);            // call method
```
JS

- All JavaScript code actually runs inside of an object
- By default, code runs inside the global window object
  - all global variables and functions you declare become part of window
- The this keyword refers to the current object

# Event Handler Binding

```
function pageLoad() {
  $("ok").onclick = okayClick;   // bound to okButton here
}

function okayClick() {           // okayClick knows what DOM object
  this.innerHTML = "booyah";     // it was called on
}

window.onload = pageLoad;
```

JS

OK

output

- Event handlers attached unobtrusively are **bound** to the element
- Inside the handler, that element becomes this (rather than the window)

# Fixing Redundant Code with this

```HTML
<fieldset>
  <label><input type="radio" name="ducks" value="Huey"  /> Huey</label>
  <label><input type="radio" name="ducks" value="Dewey" /> Dewey</label>
  <label><input type="radio" name="ducks" value="Louie" /> Louie</label>
</fieldset>
```

```JS
function processDucks() {
  if ($("huey").checked) {
    alert("Huey is checked!");
  } else if ($("dewey").checked) {
    alert("Dewey is checked!");
  } else {
    alert("Louie is checked!");
  }
  alert(this.value + " is checked!");
}
```

- If the same function is assigned to multiple elements, each gets its own bound copy

# Summary

- Global DOM Objects
  - window, document, location
  - navigator, screen, history
- Unobtrusive JavaScript
  - Browsers' loading & running JS
  - Add event handler at runtime – window.onload event
- JavaScript tips
  - Anonymous functions
  - this
  - Reuse event handlers

# Exercises

- Write a JavaScript script alters all links within an html page
  - Casually get a html page from the Web, and then store it locally
  - Bind your JavaScript file to this locally stored page
  - Use JavaScript alters all links within the page
    - Adding prefix “link to [” and suffix “] was cancelled” to texts of these links
    - Showing destinations of all links in the current page in a message box

# Further Readings

- W3School DOM node reference
  http://www.w3school.com/dom/dom_node.asp/
- W3School DOM tutorial
  http://www.w3schools.com/htmldom/
- Quirksmode DOM tutorial
  http://www.quirksmode.org/dom/intro.html
- Unobtrusive JavaScript
  http://en.wikipedia.org/wiki/Unobtrusive_JavaScript
- Unobtrusive JavaScript – Rules to work by
  http://ajaxian.com/archives/unobtrusive-javascript-rules-to-work-by

# Thank you!