1 Unobtrusive JavaScript
# The six global DOM objects

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document</td>
<td>current HTML page and its content</td>
</tr>
<tr>
<td>history</td>
<td>list of pages the user has visited</td>
</tr>
<tr>
<td>location</td>
<td>URL of the current HTML page</td>
</tr>
<tr>
<td>navigator</td>
<td>info about the web browser you are using</td>
</tr>
<tr>
<td>screen</td>
<td>info about the screen area occupied by the browser</td>
</tr>
<tr>
<td>window</td>
<td>the browser window</td>
</tr>
</tbody>
</table>
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
The location object

- **the URL of the current web page**
- **properties:**
  - host, hostname, href, pathname, port, protocol, search
- **methods:**
  - assign, reload, replace
The navigator object

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

```javascript
if (navigator.appName === "Microsoft Internet Explorer") {
  ...
}
```
The screen object

- information about the client's display screen
- properties:
  - availHeight, availWidth, colorDepth, height, pixelDepth, width
The history object

- the list of sites the browser has visited in this window
- properties:
  - length
- methods:
  - back, forward, go
- sometimes the browser won't let scripts view history properties, for security
Unobtrusive JavaScript

- 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
Unobtrusive JavaScript

- 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)

- 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

- 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?

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

$("ok").onclick = okayClick;
```
When does my code run?

- 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
When does my code run?

- 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

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

```javascript
// global code
var x = 3;
function f(n) { return n + 1; }
function g(n) { return n - 1; }
x = f(x);
```
A failed attempt at being unobtrusive

- 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

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

```js
// 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
An unobtrusive event handler

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

// called when page loads; sets up event handlers
function pageLoad() {
    $("ok").onclick = okayClick;
}
function okayClick() {
    alert("booyah");
}
window.onload = pageLoad; // global code
Common unobtrusive JS errors

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

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

- 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

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

function okayClick() {
    alert("booyah");
}
```
The keyword `this`

```javascript
this.fieldName // access field
this.fieldName = value; // modify field
this.methodName(parameters); // call method
```

- 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
The keyword `this` 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!");
}
```
Example: Tip Calculator

```html
<h1>Tip Calculator</h1>
<div>
  $<input id="subtotal" type="text" size="5" /> subtotal
  <br />
  <button id="tenpercent">10%</button>
  <button id="fifteenpercent">15%</button>
  <button id="eighteenpercent">18%</button>
  <span id="total"></span>
</div>
```

```js
window.onload = function() {
  $("tenpercent").onclick = computeTip;
}
function computeTip{
  var subtotal = parseFloat($("subtotal").value);
  var tipAmount = subtotal*0.1;//Add this code
  $("total").innerHTML = "Tip: $" + tipAmount;
}
```