window.onload Conflict

- Can only set `window.onload = function` once
  - What if you have multiple files for handlers?
  - What if you're using other libraries?
- Alternative is to add a `listener`

```javascript
window.addEventListener(eventName, function, false)
```

- `eventName` is a String (e.g., “load”)
- `function` is a reference to the handler
- `false` is for the `useCapture` parameter
  - `true` grabs event at a different point in its DOM traversal
- Can use listeners for other elements and events, too!
Event

- Contains information about a JavaScript event
- Accessed differently in earlier versions of IE

```javascript
Firefox
function handler(event) {
    ...
}

IE8 or Earlier
function handler() {
    var event = window.event;
    ...
}

Both
function handler(event) {
    var theEvent = event;
    if (! theEvent) {
        theEvent = window.event;
    }
    ...
}
```

### Some Event Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Event Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Event type (e.g., &quot;click&quot;)</td>
<td>All</td>
</tr>
<tr>
<td>target</td>
<td>Element</td>
<td>Element subject to the event</td>
<td>All</td>
</tr>
<tr>
<td>clientX, clientY</td>
<td>Number</td>
<td>Coordinates of the mouse cursor in the browser window</td>
<td>Mouse</td>
</tr>
<tr>
<td>screenX, screenY</td>
<td>Number</td>
<td>Coordinates of mouse cursor on the screen</td>
<td>Mouse</td>
</tr>
<tr>
<td>keyCode</td>
<td>Number</td>
<td>The ASCII value of the key that was pressed</td>
<td>Keyboard</td>
</tr>
<tr>
<td>altKey, ctrlKey, shiftKey</td>
<td>Boolean</td>
<td>Whether the ALT, CTRL, or Shift keys were also pressed</td>
<td>Keyboard</td>
</tr>
</tbody>
</table>
The DOM Tree

- A tree representation of the HTML document
  - All nodes are HTMLElement
  - Root is the html tag
  - Child nodes represent one of three items
    - An HTML element
    - An attribute and associated value of an element
    - Some text within an element
  - All internal nodes represent HTML elements
    - Attributes and text form the leaf nodes
- Parent nodes mirror containing elements
  - Child nodes mirror sub-elements

```html
<html>
  <head>
    <title>...</title>
    <meta>...</meta>
  </head>
  <body>
    <h1>H</h1>
    <p>T
      <ol>
        <li>E</li>
        <li>X</li>
      </ol>
    T
  </p>
</body>
</html>
```
Tree Traversal

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parentNode</td>
<td>Element</td>
<td>The parent node of this node</td>
</tr>
<tr>
<td>childNodes[]</td>
<td>Array</td>
<td>An array of the node's children</td>
</tr>
<tr>
<td>nextSibling,</td>
<td>Element</td>
<td>The next and previous nodes with the same parent node</td>
</tr>
<tr>
<td>previousSibling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>firstChild, lastChild</td>
<td>Element</td>
<td>The first and last child nodes of this node</td>
</tr>
</tbody>
</table>

Undeclared Functions

- Can create functions without names
  - Also known as *anonymous* functions
  - Can assign to a variable or use as an argument

- Why do this?
  - Can be a shortcut when creating/assigning handlers
  - Required in a few situations

```javascript
function() {
    ...
}
```
jQuery

- JavaScript library with useful methods
  - Helpful with the DOM, events, animation and AJAX
- Created by John Resig
  - Announced at “Barcamp NYC” in 2006
- Very popular
  - Used by Google, Mozilla, WordPress, Apple, ...
- Include it like any other JavaScript file
  - Hosted on several content delivery networks (CDNs)
  - Use URLs as the `src` in the `script` tag

```html
<script src="https://code.jquery.com/jquery-3.2.1.min.js"
      integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="
      crossorigin="anonymous"></script>
```

Using jQuery

- Centered around the jQuery class
  - Identified in several ways
    - `jQuery` ↔ `window.jQuery`
    - `$` ↔ `window.$`
- Based around selectors
  - Usually strings used to identify elements in the DOM
  - Several varieties
    - "tag" selects the elements with the given tag
    - "#id" selects the element with the given id
    - ".class" selects elements with the given class
    - ... and more
Using jQuery

- Pass selectors to a jQuery constructor

  \[
  $(\text{selector})
  \]

  - Returns a jQuery object
  - Contains all the elements that match the selector

- Call methods on the jQuery object to alter the elements
  - Full API listing on the jQuery website

  \[
  $(\text{document}).ready(function);
  \]

  - Alternative to \$(\text{window}).load(function);
  - Called once entire page is loaded (not just DOM)

Some Useful Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addClass(className), removeClass(className)</td>
<td>Adds/removes the CSS class(es) from the elements</td>
</tr>
<tr>
<td>prepend(content), append(content)</td>
<td>Adds content to the beginning/end of an element</td>
</tr>
<tr>
<td>text(), text(newText)</td>
<td>Returns/sets the text (without tags) within the element</td>
</tr>
<tr>
<td>height(), width()</td>
<td>Returns the height/width of the element</td>
</tr>
<tr>
<td>css(propertyName), css(propertyName, value)</td>
<td>Gets sets the CSS value for the given property</td>
</tr>
</tbody>
</table>
Integrating Timing

- There are many situations where timing is useful
  - Ensuring the user performs an action in a timely manner
  - Executing an action repeatedly over time
  - Displaying animation
- `Window` offers some global functions to integrate timing
- The two options are `timeout` and `interval`
  - Both set a timer that counts down
  - Both call a function once countdown is complete
  - Only interval repeats the process

## Timing Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setTimeout(anonFunction, delay)</code></td>
<td>Number</td>
<td>Starts a timer that calls <code>anonFunction</code> <em>once</em> after <code>delay</code> milliseconds; returns the timer's ID</td>
</tr>
<tr>
<td><code>clearTimeout(timerID)</code></td>
<td>None</td>
<td>Cancels the timer associated with a timeout and the given ID</td>
</tr>
<tr>
<td><code>setInterval(anonFunction, delay)</code></td>
<td>Number</td>
<td>Starts a timer that calls <code>anonFunction</code> <em>repeatedly</em> after <code>delay</code> milliseconds; returns the timer's ID</td>
</tr>
<tr>
<td><code>clearInterval(timerID)</code></td>
<td>None</td>
<td>Cancels the timer associated with an interval and the given ID</td>
</tr>
</tbody>
</table>