Problems with JavaScript

JavaScript is a powerful language, but it has many flaws:

- the DOM can be clunky to use
- the same code doesn't always work the same way in every browser
  - code that works great in Firefox, Safari, ... will fail in IE and vice versa
- many developers work around these problems with hacks (checking if browser is IE, etc.)
Prototype framework

- the Prototype JavaScript library adds many useful features to JavaScript:
  - many useful extensions to the DOM
  - added methods to String, Array, Date, Number, Object
  - improves event-driven programming
  - many cross-browser compatibility fixes
  - makes Ajax programming easier (seen later)

The $ function (9.1.3)

$ ("id")

- returns the DOM object representing the element with the given id
- short for `document.getElementById("id")`
- often used to write more concise DOM code:

```javascript
$("footer").innerHTML = $("username").value.toUpperCase();
```
DOM element objects (7.2.5)

HTML

```html
<p>Look at this octopus:
<img src="octopus.jpg" alt="an octopus" id="icon1" />
Cute, huh?</p>
```

```
DOM Element Object

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tagName</td>
<td>&quot;IMG&quot;</td>
</tr>
<tr>
<td>src</td>
<td>&quot;octopus.jpg&quot;</td>
</tr>
<tr>
<td>alt</td>
<td>&quot;an octopus&quot;</td>
</tr>
<tr>
<td>id</td>
<td>&quot;icon1&quot;</td>
</tr>
</tbody>
</table>
```

JavaScript

```javascript
var icon = document.getElementById("icon1");
icon.src = "kitty.gif";
```

- every element on the page has a corresponding DOM object
- access/modify the attributes of the DOM object with `objectName.attributeName`

DOM object properties (7.2.5)

```html
<div id="main" class="foo bar">
  <p>Hello, <em>very</em> happy to see you!</p>
  <img id="icon" src="images/borat.jpg" alt="Borat" />
</div>
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>tagName</td>
<td>element's HTML tag</td>
<td><code>$(&quot;main&quot;).tagName is &quot;DIV&quot;</code></td>
</tr>
<tr>
<td>className</td>
<td>CSS classes of element</td>
<td><code>$(&quot;main&quot;).className is &quot;foo bar&quot;</code></td>
</tr>
</tbody>
</table>
| innerHTML    | content inside element       | `$("main").innerHTML is \n <p>Hello, <em>ve...`
| src | URL target of an image | \$\text{("icon").src} \text{is} \ "\text{images/borat.jpg}" |
DOM properties for form controls

```
<input id="sid" type="text" size="7" maxlength="7" />
<input id="frosh" type="checkbox" checked="checked" /> Freshman?
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>the text in an input control</td>
<td><code>&quot;sid&quot;.value</code> could be &quot;1234567&quot;</td>
</tr>
<tr>
<td>checked</td>
<td>whether a box is checked</td>
<td><code>&quot;frosh&quot;.checked</code> is true</td>
</tr>
<tr>
<td>disabled</td>
<td>whether a control is disabled (boolean)</td>
<td><code>&quot;frosh&quot;.disabled</code> is false</td>
</tr>
<tr>
<td>readOnly</td>
<td>whether a text box is read-only</td>
<td><code>&quot;sid&quot;.readOnly</code> is false</td>
</tr>
</tbody>
</table>

Abuse of `innerHTML`

```
// bad style!
var paragraph = document.getElementById("welcome");
paragraph.innerHTML = "<p>text and <a href="page.html">link</a>";
```

- `innerHTML` can inject arbitrary HTML content into the page
- however, this is prone to bugs and errors and is considered poor style
- we forbid using `innerHTML` to inject HTML tags; inject plain text only
  - (later, we'll see a better way to inject content with HTML tags in it)
Adjusting styles with the DOM (8.2.2)

```html
<button id="clickme">Color Me</button>
```

```js
window.onload = function() {
    document.getElementById("clickme").onclick = changeColor;
};

function changeColor() {
    var clickMe = document.getElementById("clickme");
    clickMe.style.color = "red";
}
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>style</td>
<td>lets you set any CSS style property for an element</td>
</tr>
</tbody>
</table>

- contains same properties as in CSS, but with camelCasedNames
  - examples: backgroundColor, borderLeftWidth, fontFamily

Common DOM styling errors

- many students forget to write `.style` when setting styles
  ```js
  var clickMe = document.getElementById("clickme");
  clickMe.color = "red";
  clickMe.style.color = "red";
  ```

- style properties are capitalized like This, not like-this
  ```js
  clickMe.style.fontSize = "14pt";
  clickMe.style.fontSize = "14pt";
  ```

- style properties must be set as strings, often with units at the end
  ```js
  clickMe.style.width = 200;
  clickMe.style.width = "200px";
  clickMe.style.padding = "0.5em";
  ```
  - write exactly the value you would have written in the CSS, but in quotes
Unobtrusive styling (8.2.3)

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

```css
.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

Timer events (9.2.6)

<table>
<thead>
<tr>
<th>method</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setTimeout(function, delayMS);</code></td>
<td>arranges to call given function after given delay in ms</td>
</tr>
<tr>
<td><code>setInterval(function, delayMS);</code></td>
<td>arranges to call function repeatedly every <code>delayMS</code> ms</td>
</tr>
<tr>
<td><code>clearTimeout(timerID);</code></td>
<td>stops the given timer so it will not call its function</td>
</tr>
<tr>
<td><code>clearInterval(timerID);</code></td>
<td></td>
</tr>
</tbody>
</table>

- both `setTimeout` and `setInterval` return an ID representing the timer
  - this ID can be passed to `clearTimeout/Interval` later to stop the timer
setTimeout example

```html
<button onclick="delayMsg();">Click me!</button>
<span id="output"></span>

function delayMsg() {
    setTimeout(booyah, 5000);
    $('output').innerHTML = "Wait for it...";
}

function booyah() {  // called when the timer goes off
    $('output').innerHTML = "BOOYAH!";
}
```

setInterval example

```javascript
var timer = null;  // stores ID of interval timer

function delayMsg2() {  
    if (timer == null) {
        timer = setInterval(rudy, 1000);
    } else {
        clearInterval(timer);
        timer = null;
    }
}

function rudy() {  // called each time the timer goes off
    $('output').innerHTML += " Rudy!";
}
```
Passing parameters to timers

```javascript
function delayedMultiply() {
    // 6 and 7 are passed to multiply when timer goes off
    setTimeout(multiply, 2000, 6, 7);
}

function multiply(a, b) {
    alert(a * b);
}
```

- any parameters after the delay are eventually passed to the timer function
  - doesn't work in IE6; must create an intermediate function to pass the parameters
- why not just write this?

```
setTimeout(multiply(6 * 7), 2000);
```

Common timer errors

- many students mistakenly write () when passing the function

```
setTimeout(booyah(), 2000);
setTimeout(booyah, 2000);
setTimeout(multiply(num1 * num2), 2000);
setTimeout(multiply, 2000, num1, num2);
```

  - what does it actually do if you have the ()?
  - it calls the function immediately, rather than waiting the 2000ms!