Web Programming Step by Step

Lecture 14
DOM and Timers
Reading: 7.2 - 7.3; 8.2; 9.2.6

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

```
<script src="http://www.cs.washington.edu/education/courses/cse190m/09sp/prototype
 type="text/javascript"></script>
```

```
<!-- or link to Prototype home site -->
<script src="http://prototypejs.org/assets/2008/9/29/prototype-1.6.0.3.js"
 type="text/javascript"></script>
```

- 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)

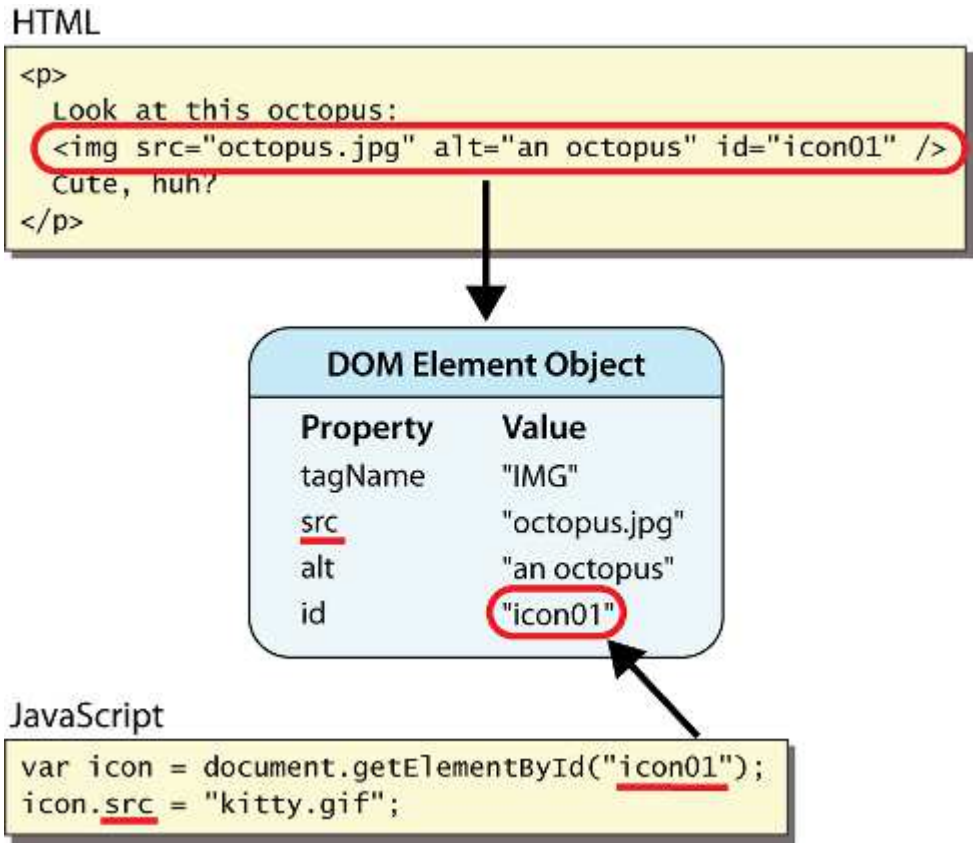
The $ function (9.1.3)

```
$("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:

```
$("footer").innerHTML = $("username").value.toUpperCase();
```

DOM element objects (7.2.5)



- 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)

```
<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>
```
HTML

Property	Description	Example
`tagName`	element's HTML tag	`$("main").tagName` is `"DIV"`
`className`	CSS classes of element	`$("main").className` is `"foo bar"`
`innerHTML`	content inside element	`$("main").innerHTML` is `"\n <p>Hello, <em>ve...`

`src`	URL target of an image	`$("icon").src` is `"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?
```
HTML

Property	Description	Example
`value`	the text in an input control	`$("sid").value` could be `"1234567"`
`checked`	whether a box is checked	`$("frosh").checked` is `true`
`disabled`	whether a control is disabled (boolean)	`$("frosh").disabled` is `false`
`readOnly`	whether a text box is read-only	`$("sid").readOnly` is `false`

Abuse of `innerHTML`

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

- `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";
}
```

output

Color Me

Property	Description
`style`	lets you set any CSS style property for an element

- 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 `likeThis`, not `like-this`

```JS
~~clickMe.style.font-size = "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)

```JS
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)

method	description
`setTimeout(`*function*`, `*delayMS*`);`	arranges to call given function after given delay in ms
`setInterval(`*function*`, `*delayMS*`);`	arranges to call function repeatedly every *delayMS* ms
`clearTimeout(`*timerID*`);` `clearInterval(`*timerID*`);`	stops the given timer so it will not call its function

- 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

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

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

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

Click me!

output

`setInterval` example

```
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!";
}
```
JS

Click me!

output

Passing parameters to timers

```
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);
}
```
JS

Click me

output

- 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);
```
JS

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);
```
JS

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