DOM and timers

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="
https://ajax.googleapis.com/ajax/libs/prototype/1.7.0.0/prototype.js "
type="text/javascript"></script>
```
JS

- 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


```JS
$("id")
```

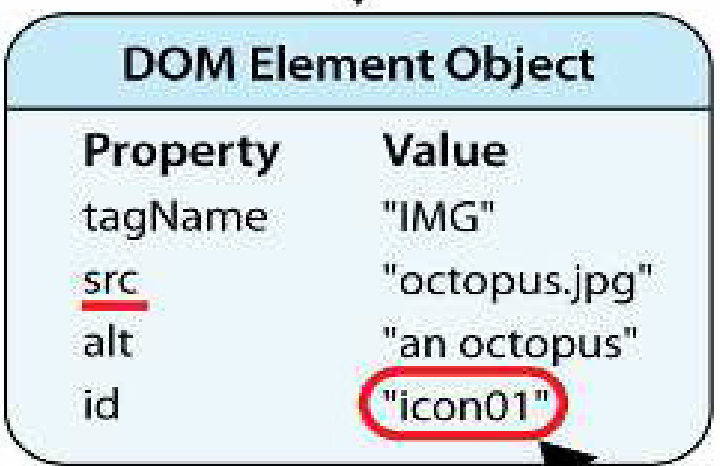
- returns the DOM object representing the element with the given id
- short for document.getElementById("id")
- often used to write more concise DOM code:

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

DOM element objects

HTML

```
<p>
  Look at this octopus:
  <img src="octopus.jpg" alt="an octopus" id="icon01" />
  Cute, huh?
</p>
```



JavaScript

```
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```

DOM object properties


```
<div id="main" class="foo bar">
<p>Hello, I am <em>very</em> happy to see you!</p>
<img id="icon" src="images/potter.jpg" alt="Potter" />
</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, ve...
src	URL target of an image	$("icon").src is "images/potter.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

Adjusting styles with the DOM


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

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

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

Common DOM styling errors

- forgetting to write .style when setting styles:

```
var clickMe = document.getElementById("clickme");
clickMe.color = "red";
clickMe.style.color = "red";                                    JS
```

- style properties are capitalized likeThis, not like-this:

```
clickMe.style.font-size = "14pt";
clickMe.style.fontSize = "14pt";                                JS
```

- style properties must be set as strings, often with units at the end:

```
clickMe.style.width = 200;
clickMe.style.width = "200px";
clickMe.style.padding = "0.5em";                                JS
```

Unobtrusive styling


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

```
.highlighted { color: red; }
```
CSS

- 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

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

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

setInterval example


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

```
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

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

- any parameters after the delay are eventually passed to the timer function
- why not just write this?

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

Common timer errors


```
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!