# Extra Slides, week 3

## CSE 190 M (Web Programming), Spring 2008
### University of Washington

**Reading: Chapter 3**



# Additional JavaScript

## Commands and syntax you won't need on your homework

## JavaScript in HTML body (example)

```HTML
<script type="text/javascript">
  JavaScript code
</script>
```

- JS code can be embedded within your HTML page's `head` or `body`
- runs as the page is loading
- this is considered *bad style* and shouldn't be done in this course
  - mixes HTML content and JS scripts (bad)
  - can cause your page not to validate

## Injecting Dynamic Text: `document.write`

```JS
document.write("message");
```

- prints specified text into the HTML page
- this is very bad style; this is how newbs program JavaScript:
  - putting JS code in the HTML file's `body`
  - having that code use `document.write`
  - (this is awful style and a poor substitute for server-side PHP programming, which we'll learn later)

# The `typeof` function

```
typeof(value)
```
JS

- given these declarations:
  - `function foo() { alert("Hello"); }`
  - `var a = ["Huey", "Dewey", "Louie"];`
- The following statements are `true`:
  - `typeof(3.14) === "number"`
  - `typeof("hello") === "string"`
  - `typeof(true) === "boolean"`
  - `typeof(foo) === "function"`
  - `typeof(a) === "object"`
  - `typeof(null) === "object"`
  - `typeof(undefined) === "undefined"`

# The `arguments` array

```
function example() {
  for (var i = 0; i < arguments.length; i++) {
    alert(arguments[i]);
  }
}
```
JS

```
example("how", "are", "you");   // alerts 3 times
```
JS

- every function contains an array named `arguments` representing the parameters passed
- can loop over them, print/alert them, etc.
- allows you to write functions that accept varying numbers of parameters

# The "for each" loop

```
for (var name in arrayOrObject) {
  do something with arrayOrObject[name];
}
```
JS

- loops over every index of the array, or every property name of the object
- using this is actually discouraged, for reasons we'll see later

# Arrays as maps

```
var map = [];
map[42] = "the answer";
map[3.14] = "pi";
map["champ"] = "suns";
```
JS

- the indexes of a JS array need not be integers!
- this allows you to store *mappings* between an index of any type ("keys") and value
- similar to Java's `Map` collection or a hash table data structure

# Date object

```
var today = new Date();                // today
var midterm = new Date(2007, 4, 4);    // May 4, 2007
```
JS

- methods
  - `getDate`, `getDay`, `getMonth`, `getFullYear`, `getHours`, `getMinutes`, `getSeconds`, `getMilliseconds`, `getTime`, `getTimezoneOffset`, `parse`, `setDate`, `setMonth`, `setFullYear`, `setHours`, `setMinutes`, `setSeconds`, `setMilliseconds`, `setTime`, `toString`
- quirks
  - `getYear` returns a 2-digit year; use `getFullYear` instead
  - `getDay` returns day of week from 0 (Sun) through 6 (Sat)
  - `getDate` returns day of month from 1 to (# of days in month)
  - `Date` stores month from 0-11 (not from 1-12)

# The `eval` (evil?) function

```
eval("JavaScript code");
```
JS

```
eval("var x = 7; x++; alert(x / 2);");  // alerts 4
```
JS

- `eval` treats a String as JavaScript code and runs that code
- this is occasionally useful, but usually a very *bad idea*
  - if the string's contents come from user input, the user can cause arbitrary code execution
  - can lead to security problems and bugs