Lecture Handout
Week 4: Sept 21-23, 2015

Let's talk about Strings!

```
var coffee = "pumpkin spice latte";
```

To a computer, this is represented as an Array of Characters:

	p	u	m	p	k	i	n		s	p	i	c	e		l	a	t	t	e
Index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Puzzle #1: Since the **data type** of `coffee` is _string_:

Method / Property	Value
`coffee.length`	19
`coffee[2]`	"m"
`Coffee[11]`	"c"
`coffee.charAt(10)`	"i"

A new variable approaches:

```
var b10 = ["UIUC", "IU", "Iowa", "UMD", "UMich", "MSU",
           "Minnesota", "Nebraska", "Northwestern",
           "OSU", "Penn State", "Purdue", "Rutgers" ];
```

Puzzle #2: Since the **data type** of `b10` is _____**array of strings**_____:

Method / Property	Value
`b10.length`	13
`b10[2]`	"Iowa"
`b10[11]`	"Purdue"

Puzzle #3: Since the **data type** of `b10[0]` is _____**string**_____:

Method / Property	Value
`b10[0].length`	4
`b10[0][1]`	"I"

Consider the following code:

```
1  var count = 0;
2  for (var i = 0; i < b10.length; i++) {
3    var b10school = b10[i];
4    if (b10school[0] == "U") {
5      count = count + 1;
6    }
7  }
```

What is the value of `count` after:

[**Puzzle #4**]: …the for-loop runs once? _____1_____

[**Puzzle #5**] … the for-loop runs six times? _____3_____

[**Puzzle #6**] … the program completes? _____3_____

Suppose we want to create a meme-generator on Big 10 schools. We will call the function `b10meme` and it takes one parameter, the name of a school.

Puzzle #7: How do we start to define the function?

```
var b10meme = function (school) {
}
```

Puzzle #8: How would we call the function with the school "UIUC"?

```
b10meme("UIUC");
```

Puzzle #9: How would we call the function with the school "Michigan"?

```
b10meme("Michigan");
```

Puzzle #10: How do we define this function in full?

```
var b10meme = function(school) {
   if (school == "UIUC") {
      return "UIUC #1";
   }
   else {
      return ("Boo "+school);
   }
};
```

So far, we have seen five **data types**:

- Strings
- Numbers
- Functions
- Arrays
- SimpleImage

Today you will see the final **primitive** data type: _____**Objects**_____.

_____Object_____ allow us to associate data in a ____**dictionary**_____.
All objects are a collection of **key-value** pairs:

Key	Value
`Name`	`"Wade"`
`netId`	`"waf"`
`Office`	`"2215 SC"`

You can then define `wade` as:

```
var wade = {
name: "Wade",
netId: "waf",
office: "2215 SC"
}
```

You can refer to `wade` properties with the **dot operator**:

Method / Property	Value
`wade.name`	`"Wade"`
`wade.netId`	`"waf"`
`wade.office`	`"2215 SC"`

Until now, we have provided the HTML to run your JavaScript for you. How did we do this?

- HTML in an acronym for **H**yper**T**ext **M**arkup **L**anguage
- HTML is **not** a _____**programming language**_____.
  - Does **not** contain: **conditionals, loops, variables**

HTML documents contain structured **tags**:

```
1  <html>
2    <head>
3      <title>CS 105</title>
4    </head>
5    <body>
6      <div>
7        Hello, world!
8      </div>
9    </body>
10 </html>
```

The single most flexible tag in HTML is the `<div>` tag:

- tags can _____.
- tags can _____.

HTML tags can also have properties:

```
<div class="cs105" onclick="jsFunction();"> ... </div>
```

The above HTML has two properties:

Name	Value	Functionality
`class`		
`onclick`		

Puzzle #11: