CS50 Section. Week 10. 11/10/15.

Tuesdays 7:00-8:30pm, Science Center 309A. https://github.com/hathix/cs50-section.

Neel Mehta. neelmehta@college.harvard.edu. (215) 990-6434.

Get these handouts at https://github.com/hathix/cs50-section/tree/master/handouts.

JavaScript

Syntactically very similar to C and PHP. Like PHP, it's *interpreted*, has no explicit variable types (but you don't need that `$` at the start of every variable name), and is used to code the web (mostly the frontend but also the backend.)

Functions

There are 2 syntaxes you can use!

```
// "function declaration" syntax
function add(a, b) {
    return a + b;
}

// "function literal" syntax
var subtract = function(a, b) {
    return a - b;
};
```

Functions are data types just like integers, strings, etc!

```
function call(myFunction, x, y) {
    return myFunction(x, y);
}

// prints 7
console.log(call(subtract, 9, 2));

// prints 12
console.log(call(function(a, b){
    return a * b;
}, 3, 4));
```

Callbacks

Functions that you pass to other functions are called *callbacks*.

```
// prints "Hi!"
function callback(){
    console.log("Hi!");
};
waitForABitThenRun(callback);
```

Arrays

Like PHP arrays!

```
// mixed types in arrays
var numbers = [1, 2, 3];
var stuff = ["Cherry", 43.5, ["a", "b", "c"]];

// prints 1, 2, 3
for (var i = 0; i < numbers.length; i++) {
    console.log(numbers[i]);
}

// prints [1, 2, 3, 4]
numbers.push(4);
console.log(numbers);
```

Objects

Kind of like associative arrays in PHP, but cooler! Since functions are just like other data types, you can include them in objects too.

```
var movie = {
    title: "Moneyball",
    year: 2011,
    tags: ["baseball", "oakland"],
    synopsis: function() {
        // `this` gives access to the object's other fields
        console.log(this.title + ": " + this.year);
    }
};

// prints 2011
console.log(movie.year);

// prints "Harry Potter"
movie.title = "Harry Potter";
console.log(movie.title);

// prints "Harry Potter: 2011"
console.log(movie.synopsis());
```

jQuery

jQuery (`$`) lets you manipulate the DOM really easily and adds kinds of other cool features.

```
// changes the HTML of the element with id `danger-button`
$("#danger-button").html("Self destruct");

// makes all paragraphs red
$("p").attr("color", "red");

// hides elements of class `old` and shows those of class `new`
$(".old").hide();
$(".new").show();

// hides all paragraphs when button clicked
$("#danger-button").on("click", function() {
    $("p").hide();
});
```

Ajax

Ajax lets you query PHP files through JavaScript without refreshing the page.

```
var arguments = {
    key: "value"
};
// notice that jQuery lets you do Ajax, too!
$.getJSON("api-endpoint.php", arguments)
.done(function(data, textStatus, jqXHR) {
    // success!
    // data is a JavaScript object; ignore the other 2 variables
    console.log(data);
})
.fail(function(jqXHR, textStatus, errorThrown) {
    // failure!
    // errorThrown provides error info; ignore the other 2 variables
    console.log(errorThrown.toString());
});
```

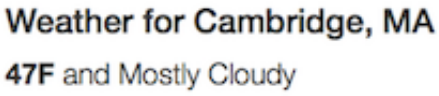
Templating

Underscore (`_`) is a utility library that lets you template a la PHP (cf. `<?= $variable; ?>`.)

```
// prints "<a href='http://yale.edu' class='btn btn-danger'>Safety school</a>"
var templateFunction = _.template("<a href='<%- url %>' class='btn btn-<%- btnClass %>'><%- text %></a>");
var htmlString = templateFunction({
    url: "http://yale.edu",
    btnClass: "danger",
    text: "Safety school"
});
console.log(htmlString);
```

Challenge: Weather

We're going to make a simple weather app!

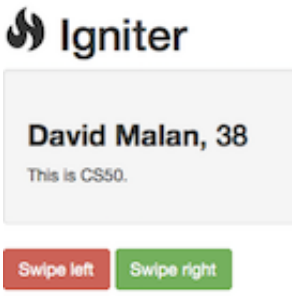


Give it a shot: http://is.gd/cs50_weather

Solution: https://github.com/hathix/cs50-section/blob/master/code/10/weather-soln.js

Challenge: Ignite

We're going to make a dating app where you can see people's bios and "swipe left" or "swipe right" on them. We're gonna be millionaires!



Give it a shot: http://is.gd/cs50_ignite

Solution: https://github.com/hathix/cs50-section/blob/master/code/10/igniter-soln.js