intro to JavaScript

class three

course adapted from:
Teaching Materials: http://www.teaching-materials.org/javascript/
Oz Girl Develop It: http://cathylill.net/courses/js101/
Girl Develop It: http://www.girldevelopit.com/materials

homework!

hello world translator

Write a function named `helloWorld` that:

- takes 1 argument, a language code (e.g. "es", "de", "en")
- returns "Hello, World" for the given language in an alert, for at least 3 languages (try using Google Translate). It should default to returning English.

Prompt the user for a language code and then call the function using the user's input.

BONUS: alert an error message if the user has entered something that isn't one of your language codes.

a little review

The story thus far:

- We've learned about ways to name and process data - we can declare **variables** (that have certain **data types**) that can be manipulated in predefined ways. Super simple.
- We made **arrays** - a more complex form of data that holds multiple pieces of data. Still pretty simple.
- We learned **if/else** statements which are simple sets of instructions that execute under certain conditions.

loops

You can repeat a piece of code multiple times.

Why would you do this?

- timer count down
- showing search results
- adding images to a slideshow

while loop

Tells JS to repeat the loop until a condition is true.

```
while (expression) {
    //statements to repeat
}
```

```
var coffee=6;
while (coffee > 0) {
    alert("I've got " + coffee + ' cups of coffee!');
    coffee--;
}
```

REVIEW: this means “coffee - 1”! What would happen if we didn’t have it here?

BEWARE THE INFINITE LOOP!

for loop

Another way of repeating statements, but with all our conditions up at the top.

```
for (initialize; condition; update) {
   //statements to repeat
}
```

```
for (coffee=6; coffee > 0; coffee--) {
   alert('I still have ' + coffee + ' cups of coffee!');
}
```

break

To exit from a loop, use break.

```
for (x = 100; x < 200; x++) {
   console.log(x);
   if (x % 7 === 0) {
      console.log('Found it!');
      break;
   }
}
```

Check out the `if()` *inside* the `for()`! DID YOUR MIND JUST GET BLOWN?

continue

You can use continue to skip an iteration and keep looping.

```
for (i = 100; i < 200; i++) {
   if (i % 7 === 0) {
      console.log(i + ' is divisible by 7!');
      continue;
   }
   console.log(i);
}
```

exercise time!

even/odd reporter

Write a loop that iterates from 1 to 20. Each time it goes through the loop, check if the current number is odd or even, and report that to the screen, like “`3 is odd.`”

array

An array is a datatype that holds an ordered list of values.

```
var arrayName = [thing0, thing1, thing2];
```

```
var coffeeSpecies = ['arabica', 'robusta',
'liberica'];
var waysToBrew = ['drip', 'pour over', 'aero
press', 'chemex'];
var numCupsToDrink = [3, 4, 5, 6, 7, 8];
```

find the length of an array with the `.length` property:

```
console.log(numCupsToDrink.length);
```

finding values in an array

Use bracket notation to access a value in an array. The number inside the bracket is called the “index”, and numbering in JS starts at 0.

```
var arrayItem = arrayName[indexNum];
```

```
var rainbowColors = ['red', 'orange', 'yellow',
'green', 'blue', 'indigo', 'violet'];
var firstColor = rainbowColors[0];
var lastColor = rainbowColors[6];
```

updating and adding

Bracket notation also lets you change the values in an array:

```
var awesomeAnimals = ['ferret', 'pangolin',
'cassowary'];
awesomeAnimals[0] = 'red panda';
```

Bracket notation can also add new values to an array:

```
awesomeAnimals[3] = 'tiger shrimp';
```

Or use the `push` method:

```
awesomeAnimals.push('star-nosed mole');
```

loops and arrays

Use a for loop to easily look at each item in an array:

```
for (x=0; x<awesomeAnimals.length; x++) {
   console.log(awesomeAnimals[x]);
}
```

let’s get complicated

Single values, like this:

```
var coffee = true;
```

are **primitive** values.

It’s one piece of data with one value.

To do real work in JS, we need **objects**. They have a more complex structure - like an object in the real world. **Arrays** and **functions** are both types of objects.

objects

Objects can hold properties and methods.

```
var cody = {
   gender: 'male', // key: property
   age: 33, // key: property
   drinkCoffee: function () { //method: property
      console.log('That’s the ticket!');
   },
   cupsCoffee: function(cupNum) { //method:
property
      alert('Drank ' + cupNum + ' cups today!');
   }
};
```

using objects

Access a property or method using dot notation:

```
var coffee = {
   temp: 'hot',
   cream: function() {
      console.log('Why would you do that?');
      }
};

var myTemp = coffee.temp;
coffee.cream();
```

changing objects

Use dot notation to change object properties:

```
var coffee = {
   temp: 'hot',
   cream: false
};
coffee.cream = 'gross';
```

or add new properties:

```
coffee.color = 'brown';
```

or delete properties:

```
delete coffee.temp;
```

arrays of objects?

Yes, you can do that. Arrays can hold any data type:

```
var myFam = [
   {name: 'Graham',
   age: 27},
   {name: 'Tiffany',
   age: 42}
];

for (var i = 0; i < myFam.length; i++) {
   var fam = myFam[i];
   console.log(fam.name + ' is ' + fam.age);
}
```

objects as arguments

objects can also be passed into functions:

```
var myDog = {
name: 'Francie',
age: 5,
breed: 'super mutt'};

function describeDog(dog) {
   console.log("My dog is named " + dog.name + ".
She is " + dog.age + " years old and a " + dog.breed
+ "!");
}

describeDog(myDog);
```

wait - console.log?

`log` is a **method** that belongs to an **object** named `console`!

We didn't make `console` - it's an object that is built into the browser. JS also has some objects that are built-in (or "global"), with their own numbers and properties, such as `array`, `boolean`, `number`, `string`, `regExp`, `date`, and `math`.

(for a full list, see here)

homework!

recipe card

- Create an object to hold information on your favorite recipe. It should have properties for title (a string), servings (a number), and ingredients (an array of strings).
- On separate lines (one console.log statement for each), log the recipe information so it looks like:
  - Mole
  - Serves: 2
  - Ingredients:
  - cinnamon
  - cumin
  - cocoa

reading list

- Create an array of objects named `readingList`, where each object describes a book and has properties for the title (a string), author (a string), and readYet (a boolean indicating if you read it yet).
- Create function named `myReadingList` and iterate through the array of books. For each book, `alert` the book title and book author like so: "The Hobbit by J.R.R. Tolkien".
- Now use an if/else statement to change the output depending on whether you read it yet or not. If you read it, alert a string like 'You already read "The Hobbit" by J.R.R. Tolkien', and if not, alert a string like 'You still need to read "The Lord of the Rings" by J.R.R. Tolkien.'