JavaScript Data

Variables, Arrays, Objects, JSON, & Web Storage

Variables

We've been using them, but let's look closer:

- Declared variable. made with keyword var
- Undeclared variables. just used
- JavaScript variable declarations are processed first (hoisting).
- JavaScript variables either have global scope or function scope.

Declared Variables

Made when you use the keyword **var.**

Typically we make them and give them a value all at the same time:

```
var cat = "Abby";
```

But you can also just define them:

```
var cat, dog, fish, bird;
```

Hoisting

Because variable declarations are processed first, the following are basically equivalent:

```
cat = "Abby";
var cat;
```

AND

```
var cat;
cat = "Abby";
```

Don't abuse this! Always declare variables at the top of their scope.

Scope

Variables declared outside a function are global, but variables declared within a function exist only within that function.

```
var cat = "Abby";
var meow = function() {
  var sound = "meow";
  console.log(cat + " says " + sound);
}
```

IIFE

Immediately Invoked Function Expression

```
// Make our IIFE
(function(){

  // Put all your code here

}())  // End the definition and
      // Call the function to run the code.
```

Undeclared Variables

Variables created without using var.

- They are always global
- Not hoisted, they don't exist until they are assigned a value.

Mixing declared and undeclared variable can get really confusing.

Don't do it! Always use var.

Arrays

Working with arrays is extremely common.

Make sure you have down the basics:

- basic creation and positional access
- using a for loop to do something to each
- multi-dimensional arrays
- array methods: push, pop, join

```
// Make an empty array
var emptyArray = [];

// Declare with data
var myCats = ["Bart", "Abby", "Kyo", "Fred"];

// Get count of items
var catCount = myCats.length;


// Get something out by position
var oldest = myCats[0];
```

Simple array examples

```
// Make a cats array
var myCats = ["Bart", "Abby", "Kyo", "Fred"];

// Make a dogs array
var myDogs = ["Skadi", "Loki"];

// Make a Multi-dimensional pets array
var pets = [myCats, myDogs];

// Get a cat and a dog
var cat = pets[0][0];
var dog = pets[1][0];
```

Multi-dimensional array examples

```
// Add a cat (push adds to end)
myCats.push("Ginger");

// Remove a cat (pop removes from the end)
var ginger = myCats.pop();

// Unshift & Shift add/remove from the beginning
myCats.unshift("Chai");
var chai = myCats.shift();

// Make a comma separated list
var prettyList = myCats.join(', ');
```

Using array methods

Objects

Objects are a way to group and organize data in a more conceptual way.

Objects can group both data and behaviors:

- Data are called **properties**
- Behaviors are called **methods**

```
// Some multi-dimensional array of names?
var names = [
   ["Bart", "Abby", "Kyo", "Fred"],
   ["Skadi", "Loki"]
];
// A nice clear object
var pets = {
  cats : ["Bart", "Abby", "Kyo", "Fred"],
  dogs : ["Skadi", "Loki"]
};
```

Multi-dimensional array vs an object

JSON

JavaScript Simple Object Notation

Used in web applications to exchange data:

- JavaScript native: Easily convert between text and JS Object
- Human readable
- Easy to read & write (computers & people)
- Simpler and smaller than equivalent XML

```
{
  "cats" : [
    { "name" : "Bart", "weight" : 16 },
    { "name" : "Abby", "weight" : 13 },
    { "name" : "Kyo", "weight" : 10 },
    { "name" : "Fred", "weight" : 11 }
  ]
}
```

JSON for some cats

Web Data Storage

Historically web applications stored data in the browser using cookies.

Problems w/ use of cookies:

- Cookies are not very *friendly*.
- Often blocked to avoid tracking and malware.

HTML 5 Storage Options

HTML5 introduces new, friendlier ways to store both sort and long term data.

LocalStorage - for long term data

SessionStorage - for short tem data, gone when browser is closed.

Simple API

With either local or session storage, you use the same set of methods:

- setItem
- getItem
- removeItem

```
// Store the use name locally
// Data stored must be a string
localStorage.setItem("userName", "Mary");

// Get the name from storage
localStorage.getItem("userName");

// Remove the name from storage
localStorage.removeItem("userName");
```

JSON methods: stringify & parse

JSON Conversion

The data stored in the browser must be a string, so there are methods provided to convert between an object and string.

- JSON.stringify(object)
- JSON.parse(string)

```
// Convert the object to a string for storage
var stringPets = JSON.stringify(myPets);
localStorage.setItem("myPets", stringPets);

// Get it back and convert to an object
var storedPets = localStorage.getItem("myPets");
var petObject = JSON.parse(storedPets);
```

Using local storage