Objects & Data in JavaScript

Some familiar pieces of the JavaScript language...

Variable `var cheese = "yummy";`

Array `var animals = ["chicken", "llama", "snake", "turtle"];`

Many languages, such as PHP, have associative arrays, but JavaScript does not. Instead it has objects.

```
<?php

    $person = [ "firstname" => "John", "lastname" => "Smith"];

?>
```

JavaScript Object Literal

```
var person = { firstname: "John", lastname: "Smith", age: 25 };

console.log( person.firstname );
```

Objects are often formatted like this.

```
var person = {
    firstname: "John",
    lastname: "Smith",
    age: 25
};

console.log( person.lastname );
```

JavaScript Object - Key / Value Pairs

Notice the key : value pairs

```
var person = {
    firstname: "John",
    lastname: "Smith",
    age: 25
};
```

The keys don't go inside quotes. Neither does 25, as it is a number and JavaScript knows what that means.

JavaScript Object Keyword

You can create an object like this too...

```
var person = new Object();

person.firstname = "John";
person.lastname = "Smith";
person.age = 25;
```

There is really no advantage to creating an object this way, and you are more likely to see it the previous way.

Objects can contain arrays and other objects

Notice the array and the object inside the object.

```
var person = {
    firstname: "John",
    lastname: "Smith",
    pets: ["dog", "cat", "chicken", "llama"],
    phonenumber: {
        home: "916-123-4567",
        work: "916-432-8765",
        cell: "916-456-7890"
    }
};

console.log( person.pets[2] );
console.log( person.phonenumber.home );
```

Objects can contain functions

Because functions are first class in JavaScript, you can do this...

```
var person = {
    firstname: "John",
    lastname: "Smith",
    greeting: function(){ console.log("hello!")}
};

person.greeting();
```

Functions inside objects are called methods

Notice the use of the keyword **this**. **This** refers to this object.

```
var person = {
    firstname: "John",
    lastname: "Smith",
    greeting: function(){
        console.log(`hello ${this.firstname} ${this.lastname}` )}
};

person.greeting();
```

Object constructor

Constructor functions are useful for creating reusable object templates...

```
function Person(first, last, age, eye) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eye;
}

var myFather = new Person("John", "Doe", 50, "blue");
var myMother = new Person("Sally", "Rally", 48, "green");
```

Note the use of the **new** keyword.

Putting Objects and Functions Together...

```
function printableMessage(){
    var message = 'hello';

    function printMessage(){
        console.log(message); }

    function setMessage(newMessage){
        message = newMessage; }

    return {
        printMessage: printMessage,
        setMessage: setMessage
    };
}
```

Pattern in use

```
var awesome1 = printableMessage();
awesome1.printMessage();

var awesome2 = printableMessage();
awesome2.setMessage('hi');
awesome2.printMessage();
```

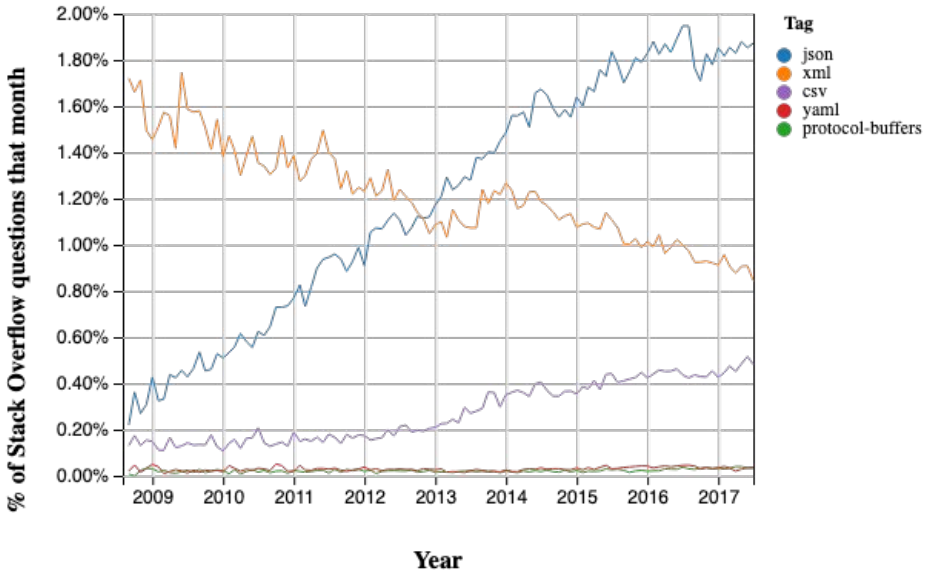
Since a new object is created every time the function printableMessage is assigned to a new variable, awesome1 is unaffected by awesome2...

```
awesome1.printMessage();
```

Data Formats - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
     <age>25</age>
     <first_name>Joe</first_name>
     <last_name>Jones</last_name>
     <social_media>
          <network>facebook</network>
          <network>Twitter</network>
          <network>Instagram</network>
     </social media>
     <online>true</online>
     <phone_numbers>
          <home>916-123-4567</home>
          <mobile>530-212-1111</mobile>
          <work>916-321-9876</work>
     </phone_numbers>
     <status>active</status>
</root>
```

XML remains a popular format for data, but has largely been surpassed by JSON.



JSON & Objects

```
var user = '{
     "first_name": "Joe",
     "last_name": "Jones",
     "age": 25,
     "social_media": [
          "facebook",
          "Twitter",
          "Instagram"
     ],
     "online": true,
     "phone_numbers": {
          "home": "916-123-4567",
          "work": "916-321-9876",
          "mobile": "530-212-1111"
     },
     "status": "active"
}';
```

```
var user = {
     first_name: "Joe",
     last_name: "Jones",
     age: 25,
     social_media: [
          "facebook",
          "Twitter",
          "Instagram"
     ],
     online: true,
     phone_numbers: {
          home: "916-123-4567",
          work: "916-321-9876",
          mobile: "530-212-1111"
     },
     status: "active"
};
```

Example File

Open the example file and experiment with the data.

The first thing to do, is to convert the JSON string into an object:

```
var user = JSON.parse(json);
```

Then let's see what each of these variables have in them.

```
console.log(json);
console.log(user);
```

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>JSON Data</title>
</head>

<body>

    <h1>Working with JSON Data</h1>

    <p>Usually JSON data will come from an external
    source, such as a database, or an API
    (Application Programming Interface), such as
    Google Maps or from Twitter. In this file, we are
    simply including the JSON directly.</p>

    <script>

        var json = '{ "first_name": "Joe",
        "last_name": "Jones", "age": 25,
        "social_media": [ "facebook", "Twitter",
        "Instagram" ], "online": true,
        "phone_numbers": { "home": "916-123-4567",
        "work": "916-321-9876", "mobile": "530-212-
        1111" }, "status": "active" }';

    </script>

</body>
</html>
```

Accessing Values

You can see that the console.log calls on the previous slide showed you the JSON string and the user object, respectively.

Now that you have an object, you can easily access values:

```
console.log(user.last_name);
console.log(user.social_media[0]);
console.log(user.phone_numbers.mobile);
```

You can change the values in the object:

```
user.first_name = "Bob";
console.log(user);
```

You can delete values in the object

```
delete user.phone_numbers.home;
console.log(user);
```

You can add values to the object

```
user.social_media[3]= "WordPress.com";
console.log(user);
```

Converting back to JSON

Once you have made all the changes to the user object, you can convert it back to a JSON string, if you want to send it out to some other service, or maybe a database.

```
var data = JSON.stringify(user);
console.log(data);
```

This is just the beginning of working with objects and JSON. There is tons more to do, and to learn, but this gives you a solid foundation that you can build upon.