JavaScript Study Notes

fwang2@ornl.gov

Table of Contents:

---

1. Overview

1.1. Use external JavaScript

```
<html>
    <head>
        <script type="text/javascript" src="xxx.js"> </script>
    </head>
</html>
```

1.2. Variable and Operators

- Comments are //
- Code blocks { … }
- Variables

```
x=5;
var x=5;
```

- Arithmetic Operators:

```
+, -, *, /, % (modulus), ++, --
```

- Assignment Operators:

```
=, +=, -=, *=, /=, %=
```

- String addition

```
x = "5" + "5" // result is "55"
x = 5 + "5"   // result is still "55"
```

- Comparison operators

```
==      // equal to
===     // exactly equal to (value and type)
!=      // not equal to
>       // greater
<       // less
>=      // greater than or equal to
<=      // less than or equal to
```

- Logical operators && // and || // or ! // not
- Conditional operator

```
varname = (condition)? value1: value2
```

1.3. Controll Structure

- if

```
if (condition) {
...
}
```

- if else

```
if (condition) {
...
} else {
...
}
```

- if … else if … else

```
if (condition)
{ ... }
else if (condition)
{ ... }
else
{ ... }
```

- switch statement

```
switch(n)
{
    case 1:
                code block
                break;
    case 2:
                code block
                break;
    default:
                code default block
}
```

- for loop

```
for (var=startvalue; var <= endvalue; var+var+increment)
{
    code;
}
```

- for … in loop

```
for (var in object) {
    code;
}
```

- while loop while (var < endvalue) { code; }
- do … while loop

```
do {
    code;
} while (var <= endval);
```

1.4. Functions

Function can be defined as:

```
function func_name(var1, var2, ... )
{
    some code
}
```

The variables declared within a function is local to the function.

1.5. Event

Every element on a web page has certain events which can trigger a JavaScript. Examples such as:

- onLoad and onUnload: triggered when user enters or leaves the page
- onFocus, onBlur, and onChange: often used in combination with validation of form field.

```
<input type="text" size="30" id="email" onchange="checkEmail()">
```

- onMouseOver and onMouseOut - often used to create animated button

2. JavaScript Object

Things to know:

- A JavaScript object is an unordered collection of properties.
- Properties consists of a name and a value
- Objects can be declared as object literals
- Top-level ‘variables’ are properties of `window`.

2.1. Properties

Properties are values associated with an object. For example of using length property of String object:

```
<script type="text/javascript">
    var txt = "Hello world!";
    document.write(txt.length);
</script>
```

2.2. Methods

Methods are actions that ban be performed on objects.

For example:

```
<script type="text/javascript">
    var str="Hello worlds!";
    document.write(str.toUpperCase();
</script>
```

2.3. Object literal

```
var ride = {
    make: "Honda',
    model: "Civic',
    purchased: new Date(2008, 3, 12),
    owner: {
        name: "Feiyi Wang",
        occupation: "Bum"
```

```
    }
};
```

This notation, come to be known as **JSON** (JavaScript Object Notation) Also notice how nested objects are declared.

2.4. Built-in Objects

- String.
- Date.

```
// return today's date
var d = new Date();

// turn time as milliseconds since 1970
document.write(d.getTime());

// set full year
d.setFullYear(1992, 10, 3)

var weekday = new Array(7);
weekday[0] = "Sunday";
weekday[1] = "Monday";
...
weekday[6] = "Saturday";
document.write("Today is" + weekday[d.getDay()]);
```

- Array

```
// create an array
var myCars = new Array();

// or
var myCars = new Array("Sabb", "Volvo", "BMW");

// or
var myCars = ["Sabb", "Volvo", "BMW");

// for loop
for (x in myCars)
{
    document.write(myCars[x] + "<br />");
}

// merge array
var parents = ["mom", "dad"]
var children = ["mike", "jenny"]
var family = parents.concat(children);

// join array elements in to string
var fstr = family.join();

// remove the last element of the array
family.pop()        // jenney
```

```
// add to last
family.push("jenny")

// add to beginning
family.unshift("grandma");
```

- Boolean

```
var myB = new Boolean();
```

If Boolean object has no initial value, or if it is 0, -0, null, ””, false, undefined, or NaN, the object is set to False.

Otherwise, it is True.

- Math
- RegExp: for regular expression
- `navigator` for browser detection.

```
// Broswer code name
navigator.appCodeName

// Browser name:
navigator.appName

// Browser version
navigator.appVersion

// Cookie enabled
navigator.cookieEnabled
```

3. JavaScript, Create Your Own Object

3.1. Create a direct instance of an object

```
// with 3 properties, you can just assign it.
p = new Object();
p.firstName = "John"
p.lastName = "Oliver"
p.age = 50;
```

3.2. Create object template

```
function person(firstname, lastname, age)
{
    this.firstname = firstname;
    this.lastname = lastname;
    this.age = age;
```

```
    this.workhours = workhours;
}
```

Notice that:

- An object tempalte is just plain function.
- You can attach method `workhours` to it, by then you have to define it somewhere.
- Once you have object template, you can create many instance of it now.

4. JavaScript Function as First-class object

A function can:

- Assigned to variables
- Assigned to a property of an object
- Passed as an paramter
- Returned as function result
- Created using literals

Function is *not* a named entity: `function` keyword creates a `Function` instance and assigned it to a `window` property if you delcare such at top-level. For example, the following two forms are the same:

```
function myFunc() {
    alert('I am working');
}

<==>

myFunc = function() {
    alert('I am working');
}
```

And the second form is also known as *function literal*, just as *object literal* we talked above.

4.1. Function as callbacks

Consider the following:

```
function hello() { alert('Hi there!'); }

setTimeout(hello, 5000)
```

or a more elegant version:

```
setTimeout( function() { alert('Hi there!'); }, 5000 )
```

In the later case, we express the functional literal directly in the parameter lsit, and no name is generated.

4.2. Function context

A function *f* acts as a method of object *o* when *o* serves as the function context of the invocation of *f*.

Example:

```
var o1 = { handle: 'o1' };
var o2 = { handle: 'o2' };
var o3 = { handle: 'o3' };

function whoAmI() {
    return this.handle;
}

o1.identifyMe = whoAmI;

alert(whoAmI()); // window

alert(o1.identifyMe()); // o1

alert(whoAmI.call(o2)); // o2

alert(whoAmI.apply(o3)); // o3
```

As you can see, you can change function context (`this`) by using function method `call()` or `apply()`.

```
alert(o1.identifyMe.call(o3)); // still o3
```

This example further shows that even we reference the function as a peroperty of `o1`, but the function context for this invocation is `o3`.

The important thing to know is not how function is declared, but how it is invoked.

4.3. Scope

- In JavaScript, scope is kept within function, but not within blocks (such as while, if and for statement).
- If you don't use `var`, then it is implictly in global scope.

  var foo = 'test';

  // within a block if (true) { // this is still within global scope var foo = 'new test'; }

  alert(foo == 'new test') // true

  function test() { var foo = 'old test'; }

  // even called, 'foo' remains within the scope of the function test();

  alert (foo == 'new test') // still true

4.4. Closure

A *closure* is a `Function` instance coupled with the local variables from its environement that are ncessary for its exeuction.

4.5. Argument checking

```
function sendMessage( msg, obj ) {
    // if both a message and object are provided
    if ( arguments.length == 2 )
        // send message to the object.
        obj.handleMesg( msg );
    else
        // otherwise, assume only a message is provided
        // so just display default msg
        alert( msg );
}
```

4.6. Type checking

```
function displayError( msg ) {
    // check and make sure that msg is not undefined

    if ( typeof msg == 'undefined' ) {
        // set default
        msg = "An error occured.';
    }

    // display the message
    alert( msg );
 }
```

5. JavaScript Development Tools

5.1. Firebug (error console, debugger, DOM inspector)

5.2. Venkman (debugging)

5.3. JSUnit (unit testing)