Introduction to JavaScript

Gordon Tian

408-668-5680
gtian@svuca.edu
gordontian@126.com

JavaScript

JavaScript is the most popular programming language in the world !!!

- JavaScript was invented by Brendan Eich in 1995, and became an ECMA standard in 1997.
- ECMA-262 is the official name. ECMAScript 6 (released in June 2015) is the latest official version of JavaScript
- **JavaScript is the default scripting language in HTML.**
- *NOTHING to do with Java*

JavaScript

JavaScript can:

1. Change HTML element content
2. Change HTML element attribute
3. Change HTML CSS
4. Validate HTML Form data

JavaScript

- JavaScript can be placed in the <body> and the <head> sections of an HTML page. It is a good idea to place scripts at the bottom of the <body> element. This can improve page load, because HTML display is not blocked by scripts loading.
- Inside HTML, JavaScript code must be inserted between <script> and </script> tags.
- External Javascript:

```
<script src="myScript.js"></script>
```

JavaScript

- A computer program is a list of "instructions" to be "executed" by the computer.
- In a programming language, these program instructions are called **statements.**
- JavaScript is a programming language.
- JavaScript **statements** are separated by semicolons (Values, Operators, Expressions, Keywords, and Comments)
- JavaScript **syntax** is the set of rules, how JavaScript programs are constructed.

JavaScript Values

- The JavaScript syntax defines two types of values: Fixed values and variable values.
- Fixed values are called literals. Variable values are called variables.
- var myName = “Gordon Tian”;

JavaScript Operators Expressions

- Operators:
  - assignment =
  - arithmetic operators + - * / % ++ -- += -= *= /= %=
  - Logical == === != !== > < >= <=

- An expression is a combination of values, variables, and operators, which computes to a value.

Javascript Identifier

Identifiers are names.

- In JavaScript, identifiers are used to name variables (and keywords, and functions, and labels).
- The rules for legal names are much the same in most programming languages.
- In JavaScript, the first character must be a letter, an underscore (_), or a dollar sign ($). NOT A NUMBER!!!
- Subsequent characters may be letters, digits, underscores, or dollar signs.
- Case sensetive
- Unicode character set

Camel Case

Historically, programmers have used three ways of joining multiple words into one variable name:

- **Hyphens:**

  first-name, last-name, master-card, inter-city.
- **Underscore:**

  first_name, last_name, master_card, inter_city.
- **Camel Case:**

  FirstName, LastName, MasterCard, InterCity.

JavaScript Code Block

- JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.
- The purpose of code blocks is to define statements to be executed together.
- One place you will find statements grouped together in blocks, are in JavaScript functions:

```
function sayHello() {
  document.getElementById(”hello").innerHTML = "Hello World!";
}
```

JavaScript Keywords

JavaScript keywords are reserved words. Reserved words cannot be used as names for variables.

break, continue, debugger, do … while, for, function, if … else, return, switch, try … catch, var, *eval, arguments, private, protected, public…*

JavaScript Data Types

Six primitive data types:

- Boolean
- Null
- Undefined
- Number, +Infinity, -Infinity, and NaN
- String
- Symbol (new in ECMAsacript 6)one

One non-primitive: Object

JavaScript Function

```
function name(parameter1, parameter2,
    parameter3) {
     code to be executed
}
```

Functions are Objects!!!

JavaScript functions have a built-in object called the arguments object.

Function Invocation

JavaScript functions can be invoked in 4 different ways.

1. All global functions belong to window object
2. Function as object methods

Function Invocation

```
var myObject = {
    firstName:"John",
    lastName: "Doe",
    fullName: function () {
        return this.firstName + " " + this.lastName;
    }
}
myObject.fullName();         // Will return "John Doe"
```

Function Invocation

3. Invoking a Function with a Function Constructor

```
// This is a function constructor:
function myFunction(arg1, arg2) {
    this.firstName = arg1;
    this.lastName  = arg2;
}
// This creates a new object
var x = new myFunction("John","Doe");
x.firstName;
```

Function Invocation

4. Invoking a Function with a Function Method call/apply

```
function myFunction(a, b) {
    return a * b;
}
myObject = myFunction.call(myObject, 10, 2);     // Will return 20
myArray = [10, 2];
myObject = myFunction.apply(myObject, myArray);  // Will also return 20
```

In JavaScript strict mode, the first argument becomes the value of this in the invoked function, even if the argument is not an object.

In "non-strict" mode, if the value of the first argument is null or undefined, it is replaced with the global object.

JavaScript Object

```
var car = {
                type:"Fiat",
                model:500,
                color:"white”,
                price:”$30,000”
};
car.price
```

JavaScript Object Prototype

- Every JavaScript object has a prototype. The prototype is also an object.
- All JavaScript objects inherit their properties and methods from their prototype.
- The standard way to create an object prototype is to use an object constructor function:

Prototype

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

Prototype

```
var myFather = new person("John", "Doe",
   50, "blue");

var myMother = new person("Sally",
   "Rally", 48, "green");
```

Prototype

```
function person(first, last, age, eyecolor) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eyecolor;
    this.nationality = "English”;
    this.name = function() {
        return        this.firstName + " " +  this.lastName;};
}
```

Prototype

```
person.prototype.nationality = "English";
person.prototype.name = function() {
    return this.firstName + " " +
  this.lastName;
};
```

JavaScript Scope

Scope is the set of variables, objects, and functions you have access to.

- Local Scope: declared inside a function
- A variable declared outside a function, becomes GLOBAL

The lifetime of a JavaScript variable starts when it is declared.

- Local variables are deleted when the function is completed.
- Global variables are deleted when you close the page.
- *In HTML, the global scope is the window object: All global variables belong to the window object.*

JavaScript Scope

JavaScript does not create a new scope for each code block.

It is true in many programming languages, but not true in JavaScript.

It is a common mistake, among new JavaScript developers, to believe that this code returns undefined:

```
for (var i = 0; i < 10; i++) {
// some code
}
return i;
```

JavaScript Arrays

var array-name = [item1, item2, ...];

var cars = ["Saab", "Volvo", "BMW"];

var cars = new Array("Saab", "Volvo", "BMW");

Arrays are objects in JavaScript!!!

There is no associative array in JavaScript!!!

In JavaScript, arrays use numbered indexes [ ]

In JavaScript, objects use named indexes { }

JavaScript Events

onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an element
onmouseout	The user moves the mouse away from an element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

JavaScript Hoisting

Hoisting is JavaScript's default behavior of moving declarations to the top.

JavaScript only hoists declarations, not initializations.

JavaScript Use Strict

"use strict"; Defines that JavaScript code should be executed in "strict mode".

Strict mode is declared by adding "use strict"; to the beginning of a JavaScript file, or a JavaScript function.

JSON

What is JSON?

- JSON stands for JavaScript Object Notation
- JSON is lightweight data interchange format
- JSON is language independent *
- JSON is "self-describing" and easy to understand

JSON

```
{
   "employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna",        "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
   ]
}
```

JSON names require double quotes. JavaScript names don't.

JavaScript JSON Object

```
var jsObject = JSON.parse(jasonstring)
```