1 Introduction to JavaScript

Before learning Sencha Touch, it's necessary to learn JavaScript as Sencha Touch is a JavaScript framework. So here in this chapter we will cover following:

- JavaScript - Birth.
- Introduction to JavaScript
- JavaScript Syntax
- Primitive data types of JavaScript
- Arrays in JavaScript
- Functions in JavaScript
- Objects in JavaScript
- Control Structures in JavaScript
- How to use JavaScript

JavaScript - Birth

JavaScript was originally invented at Netscape. They introduced it as a light weight scripting language to interact with web pages and providing dynamic content to web pages. Original name of the language developed by **Netscape** was **Livescript** but later it was changed to JavaScript when Netscape added support for Java in their Netscape Navigator web browser. After its introduction, JavaScript was quickly adopted as client side scripting language for web pages. Later **Microsoft** introduced JavaScript support to its web browser. Soon after releasing JavaScript as a client side script language Netscape introduced it as a server side scripting language. Today JavaScript is the most popular programming language among the web developers. JavaScript standards are defined by **ECMA** script.

Introduction to JavaScript

As the name suggest JavaScript is not a programming language but it's a scripting language. However it's **C** like syntax makes it look like a traditional procedural language. JavaScript supports functions, dynamic objects, loosely typed variables, associative arrays, regular expressions, prototype inheritance and **DOM** support. With features like this one can hardly call it "Just a Scripting Language".

JavaScript is **object oriented** language. But here objects are dynamic. You can create objects at run time and add properties and methods to it. Also you can remove properties and methods from object at run time. This makes JavaScript objects completely dynamic in nature. In traditional object oriented programming language, classes are defined and objects are created from template of the class. JavaScript allows you to create object directly without a class. Objects can be created simply by adding their components like properties and methods. JavaScript is also a prototype based language. New objects can be created from existing objects. JavaScript implements prototype inheritance in slightly different manner and this is the point where most of the programmers fail to understand object orientation in JavaScript.

JavaScript Syntax

Like other languages, JavaScript also have syntax that makes it completely structured language. JavaScript is case sensitive language.

Whitespaces

Whitespaces are generally referred to **tabs**, **spaces** and **newlines** used outside the string constant. In JavaScript whitespaces are normally used to separate tokens.

```
var foo = 'Hello World';
```

Here spaces between `var` and `foo` can't be removed. Other spaces like space between `foo` and `=` and space between `=` and `'Hello World'` can be removed. JavaScript uses semicolons as statement terminators. That is optional. If you don't insert semicolon but add new line between statements and if those statements are well formed, there will not be any error.

```
a = b + c
(d*e)
```

These statements are parsed as

```
a = b+c(d*e)
```

But only well formatted statements are parsed. So for a good practice it's recommended to use **semicolon** as statement terminators.

Comments

JavaScript support single line and multi-line comments.

```
//This is single line comment
/*This is
multi-line comment*/
```

Variables

Variables are declared with a `var` statement. In JavaScript variables does not have type attached to it so any value can be stored in variables. Variable name must start with a letter or underscore (_) followed by any number of letters and digits. As mentioned earlier, JavaScript is a case sensitive language so `var a` and `var A` both are different variables. Variables name should not be one of the reserved keywords of JavaScript. Like other languages JavaScript variables also have scope.

```
var a=1;
function test(){
      var b =2;
      var c = a+b;
}
```

Scope of b variable inside a function is limited to its body while variable a declared outside the function has a global scope and it also can be accessed in a function.

Any uninitialized variable will have undefined value.

```
var a; //undefined
```

Null is used to define a variable with empty value.

```
var a = null;
```

Here `undefined` and `null` are primitive data types of JavaScript.

JavaScript Operators

JavaScript has set of arithmetic operators, assignment operators, conditional operators. Following are arithmetic operators.

Operator	**Description**
+	Used for adding two numbers or to concat two strings
-	Used for subtraction of two numbers.
*	Used for multiplication of two numbers
/	Used for divison of two numbers
%	Used for getting reminder of divison
++	Used to increment variable value by 1
--	Used to decrement variable value by 1

Following are conditional operators.

Operator	Description
==	Equal to .Used to compare two values
===	Used to compare two values and types
!=	Not equal to.
!==	Not equal value and type.
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Primitive Data Types of JavaScript

Numbers

JavaScript as a single number type and it is represented internally as 64 bit floating point. Following are valid numbers in JavaScript.

```
var a =1; //Integer
var a=1.23; //Float
var a = 0111; //Octal, actual value is 73
var a = 0xFF; //Hexadecimal, actual value is 255
```

Number constructor can be used for numeric conversion.

```
var a = '11.11';
var b= Number(a);
```

Strings

In JavaScript string literal can be wrapped in single quote or double quote. It can contain any number of digits or characters. **Backslash** (\) is used as escape character.

```
var s = 'this is string';
var s = 'This is Tom\'s house'; //use of backslash to escape single
//quote.
```

Strings can be compared with == operator and it returns true if string content and all the character cases match.

Boolean

Boolean variables can be defined true or false literals.

```
var a = true;
```

Arrays in JavaScript

Array object is used to store multiple values indexed by integer keys in a single variable.

```
var arr = []; //defines an empty array

var arr = [1,2,3,4]; //defines an array with four elements.
```

Values inside an array can be accessed using index.

```
var arr = [1,2,3,4];
var b= arr[0];
```

Here variable b will have value 1. Array constructor can also be used to declare an array.

```
var arr = new Array(10); //defines array with 10 elements.
```

`length` property can be used to know number of elements in an array.

```
var arr = [1,2,3,4,5];
var b=arr.length;
```

`push` method can be used to add an element in an array. `push` method will insert element at next empty index.

```
var arr = [];
arr.push(1);
```

`sort` method can be used to sort an array. Default sort order is alphabetic and ascending.

```
var arr = [];
arr.sort();
```

`pop` method can be used to remove last element from an array.

```
var arr = [1,2,3,4,5];
arr.pop();
```

`reverse` method can be used to reverse order of elements in an array.

```
var arr = [1,2,3,4,5];
arr.reverse(); //Now arr elements will be [5,4,3,2,1]
```

JavaScript also supports an **associative** array. An **associative** array is set of key value pairs. Values are stored in association with a key and when we provide a key, array will return its **associative** value.

```
var arr = {key1: 'value1',key2: 'value2'};
var a = arr['key1']; //variable a has value 'value1'
```

Functions in JavaScript

Functions contain set of reusable statements. Every function in JavaScript is instance of Native Function object of JavaScript. Function can be defined using function **literal**.

```
function add(a,b){
	var c= a + b;
	return c;
}
```

Also we can define a variable as a function.

```
var add = function(a,b){
	var c= a + b;
	return c;
}
```

Function can be invoked by name of the function and passing necessary arguments to it.

```
add(1,2);
```

A function which returns value can be assigned to any variable.

```
var c = add(1,2);
```

Objects in JavaScript

In JavaScript **string**, **number**, **Boolean**, **null** and **undefined** are primitive data types. All other values are objects. Class definition is not required to create objects in JavaScript. JavaScript objects are **mutable** keyed collection. JavaScript object contains methods and properties. Property is key value pair and methods are instance of Function object. Generally objects are used to store and organize data. Objects can contain other objects, to represent tree structure. Object literal can be specified by pair of curly braces with zero or more than one properties or methods. Objects are always passed by reference.

```
var obj = {}; //represent an empty object.

var obj = {
      name: 'John Smith',
      phone: '123456789'
}; //represent object with two properties.
```

Specific property value can be accessed by using name of property.

```
var name = obj.name; //name will have value 'John Smith'
```

Any attempt of accessing non existing member will return value `undefined`.

```
var email = obj.email; //email will have value undefiend.
```

Member can also be accessed in associative array style.

```
var phone = obj['phone']; //phone will have value 123456789
```

Any member value can be updated by assigning new value to it.

```
obj.name = 'Mark Smith';
obj['name'] = 'Mark Smith';
```

Any member of object can be deleted using delete statement.

```
delete obj.name;
```

`typeof` operator can be used to determine type of object property.

```
alert(typeof obj.name); //will alert string.
```

JavaScript supports inheritance by **prototype** model.

```
var base = function() {
      this.baseproperty = 'Base Property';
      this.overridefunction = function(){
            alert('base function');
      }

}
var derived = function(){
      this. overridefunction = function(){
            alert('derived function');
      }
}

var b = new base();
derived.prototype =  b;

var d= new derived();

d.overridefunction(); // shows alert ' derived function '
var property = d. baseproperty; //stores value 'Base Property' in
variable
```

Also a new member can be added in object as follow.

```
Base.prototype.myMethod = function(){
}
```

Control Structure in JavaScript

if else

It is used to evaluate condition and execute statements based in evaluation.

```
if(condition){
      //statements
}else if(condition){
      //statements
}else{
      //statements
}
```

Turnery operator

Turnery operator is same as if else statement. It evaluates the expression and executes one of the statements.

```
var result = (condition)? Statements: Alternatives;

var result = (true)? alert(‘True’): alert(‘False’);
```

Here both statements and alternatives should be specified. Above code will show True in alert.

Switch Statement

Switch Statement is used to evaluate condition and execute more than one cases based on it.

```
switch(condition){
      case firstvalue: break;
      case secondvalue: break;
      case thirdvalue: break;
      default: break;
}
```

`break` statement is optional but it's necessary to stop execution of next cases. `default` is optional, but generally it is use to manage default case. String literals can be used instead of values.

For Loop

For loop is used to execute statements more than till particular condition is matched.

```
for(initialization; condition ; increment) {
      //statements
}
```

For In loop

Generally for in loop is used to iterate through all the properties of object or all the indices of an array.

```
for( var property in object){
      //statements
}
```

While loop

While loop is used to execute statements more than once based on certain condition. Unlike for loop there is no increment in while loop.

```
while(condition){
      //statements
}
```

Do While loop

Do while loop is same as while loop. Only difference is, statements inside loop will be executed at least once.

```
do{
      //statements
}while(condition)
```

How to use JavaScript

JavaScript can be inserted in any html page with Script tag in head section

```
<head>
      <script type=”text/javascript” language=”JavaScript”>
            //JavaScript statements.
</script>
</head>
```

Also JavaScript can be placed in external file with .js extension and can be referenced in head section.

```
<head>
      <script src=”myfile.js” language=”JavaScript” />
</head>
```