JavaScript

Who we are

Komal Jain

Ravi S. Tiwari aka RT
Socially here
Professionally here

What will we cover

- History
- Language Details
- Runtimes and Execution Environment
- Types
- Variable Declarations
- Operators
- Conditionals and Loops
- Jump Statements
- Objects
- Array
- Function
- Utilities

JavaScript - How, When, Why

- Marc Andreessen, Netscape founder, believed HTML needed a “glue language”.
- Brendan Eich was hired with the goal of embedding Scheme in Netscape Navigator - April 1995.
- Instead, Eich was commissioned to create a language that resembled Java.
- Eich wrote a prototype in **10 days** in May 1995.
- Enters Mocha, *ahem*, LiveScript, *ahem*, JavaScript.
- Standardised by ECMA(**European Computer Manufacturers Association**): ECMA-262 by committee TC39.
- Officially called ECMAScript.
- Various implementations of the standard(ECMAScript) viz., JavaScript(most popular), ActionScript, JScript.

JavaScript - The journey so far

- Netscape submitted JavaScript to Ecma International - 1996.
- First Edition - emergence as ECMA-262 standard - 1997.
- Second Edition - ECMAScript 2 - 1998.
- ECMAScript 3 - 1999.
- ECMAScript 4 - Work started in 2000, never saw the sun - discord among players - guess what - Microsoft.
- ECMAScript 3.1 - ECMAScript 4 renamed to 3.1 - 2007.
- ECMAScript 5 - ECMAScript 3.1 renamed to 5 - 2009.
- Harmony - Agenda to drive language development further.
- ECMAScript 5.1 - 2011 - aka **ES5** - ECMAScript 5 with minor corrections.
- ECMAScript 2015 - 2015 - aka **ES6.**
- ECMAScript 2016 - 2016 - probably ES7.
- ECMAScript 2017 - WIP.

JavaScript - Language Details

- Started as a scripting language.
- Now used as a general purpose programming language. It supports multiple paradigms of programming - Functional, Object Oriented.
- From browser to mobiles, servers, embedded systems, TV, you name it.
- Atwood Law: "Any application that *can* be written in JavaScript, *will* eventually be written in JavaScript".
- Any programming language offers capabilities based on runtime environment.
- JavaScript is the same everywhere, but things that can be achieved with it depends on the runtime, e.g. you can access process, stream in Node, but not in browser.
- And for the sake of sanity, please read out aloud - "JavaScript is not threaded at all - neither single nor multiple". Thread, Process fork etc are concerns of runtime not the language.

JavaScript - The runtime aka Engine

- JavaScript has come a long way - 20 years.
- Big internet giants - Google, Mozilla, Microsoft, Apple - have created respective runtimes aka engines.
- Notable among them are - V8 by Google, [Spider, Trace]Monkey by Mozilla, Chakra by Microsoft, Nitro by Apple.
- These engines vary in their execution speed and performance - V8 is a winner here, used to power Node, and many more.
- Almost all engines compile JavaScript in-place - on browser or on node.
- Compilation not in traditional sense - no intermediate bytecode available.

ES5.1

JavaScript - Browser as our execution environment

- Browsers have built in JS engines.
- We will use browser capabilities to run and learn JavaScript.
- Create a directory containing two files - index.html and app.js
- Refer notes for the content of these files.
- Browse the index.html file in chrome
- Open Console Tab in Developer Tools

JavaScript - Data Types

- JavaScript is loosely-typed, dynamic, case-sensitive language.

  ```
  var seven = 7;
  seven = "See, I am assigned a string value";
  ```

- The language defines following data types:
  1. Undefined
  2. Null
  3. Boolean
  4. Number
  5. String
  6. Symbol - **ES6 onwards**
  7. Object

- All but **Object** are primitive types - they are immutable, their values cannot be changed.

JavaScript - Data types - Undefined

- The Undefined type has only one value - ***undefined*** - note the lowercase.
- Whenever a variable has not been assigned a value, the runtime assigns it the **undefined** value.

```
var primitive_undefined;
console.log('Every declared uninitialized variable holds the value : ', primitive_undefined);


console.log('If we try access a variable that is not declared, we get : ');
console.log(undeclared_variable);
```

- Note: The variable must have been declared, otherwise we get a ReferenceError

JavaScript - Data types - Null

- The Null type has only one value - ***null*** - note the lowercase.
- **null** represents intentional absence of an object value .

```
        var null_value = null;
console.log(null_value, 'represents intentional absence of Object value');
        null_value = new Date();
```

- **null** is meant for objects, **undefined** can be used for both primitive values and objects. However, this is not a strict enforcement.

JavaScript - Data types - Boolean

- The Boolean data type has two values - **true** and **false.**
- There is a wrapper object for boolean types - **Boolean**.
- Wrapper objects encapsulate primitive values.

JavaScript - Data types - Number

- Number in JavaScript is [double-precision 64-bit binary format IEEE 754 value] .
- Can represent numbers from **$-(2^{53} - 1)$** to **$2^{53} - 1$**.
- In addition, there are three symbolic values **-Infinity**, **+Infinity**, **NaN**(Not a Number)
- There are no integers, floats, decimal, long - only numbers. **0**(Zero) has two representations **-0** and **+0**. **0** is **+0**
- **Number** is the wrapper object for Number data type. It has various properties like **Number.MAX_VALUE**, **Number.MIN_VALUE**.
- Numbers can be represented in decimal, octal, hexadecimal and scientific notations

  Decimal = 12345 - any valid decimal digit - 0 through 9
  Octal = **0**1234 - any valid octal digit prefixed with 0 - 0 through 7
  Hex = **0x**DEADCAFE9 - any valid hex digit - 0 through 9, A through F - case insensitive
  Sci = 2.99**E**8 - either E or e

JavaScript - Data types - String

- The String data type represents textual data that are UCS-2 (UTF-16) code points.*
- The default encoding for JavaScript is UTF-8.
- These are immutable sequence of 16 bits, considered as primitive values.
- Operations that seem to modify strings actually return a new string with the modifications.
- **String** is the wrapper object for String type.
- There is no character type in JavaScript. Characters are represented as string of length 1.

* UCS-2 represents the Basic Multilingual Plane. It contains all code points from \u0000 to \uFFFF. ES6 supports characters from other planes as well. UCS-2 is deprecated now.

JavaScript - Data types - Object

- Object represents a container type whose value is accessed via reference in memory.
- Except primitives, everything in JavaScript is an Object
- **Date**, **Array**, **Math**, **RegExp**, **Error**, **Function**, **Promise** (ES6 onwards) etc all are objects.
- **Function** is an special object with additional capability of being *callable*.

JavaScript - Data Types In Action

```
var audienceStrength;
var speaker = null;
var isCommonSession = true;
var sessionDurationInDays = 2;
var sessionTitle = 'JavaScript';
var sessionDetails = {
  audienceStrength: undefined,
  speaker: null,
  isCommonSession: true,
  sessionDurationInDays: 2,
  sessionTitle: 'JavaScript'
};
```

```
var undefinedValue = undefined;
var anotherWayForUndefined = void 0;

var singleQuoteString = 'SQS',
    doubleQuoteString = "DQS";

var wrapperString = new String("WS");
console.log(wrapperString.valueOf());
var wrapperNumber = new Number(10);
console.log(wrapperNumber.valueOf());
```

JavaScript - Building Blocks In A Glance

Variable	Explanation	Example
String	A string of text. To signify that the variable is a string, you should enclose it in quote marks.	`var myVariable = 'Bob';`
Number	A number. Numbers don't have quotes around them.	`var myVariable = 10;`
Boolean	A True/False value. The words `true`and `false` are special keywords in JS, and don't need quotes.	`var myVariable = true;`
Array	A structure that allows you to store multiple values in one single reference.	`var myVariable = [1,'Bob','Steve',10];` Refer to each member of the array like this: `myVariable[0]`, `myVariable[1]`, etc.
Object	Basically, anything. Everything in JavaScript is an object, and can be stored in a variable. Keep this in mind as you learn.	`var myVariable = document.querySelector('h1');` All of the above examples too.

JavaScript - Comments

Comments can be single line or multi-line.

```
// This is a single line comment. It spans over a line

/* This can span single or multiple lines*/        // We can mix another comment here

/*
*   This is
*   a
*   multi line comment
*/
```

JavaScript - Literals

Literals are data values that appear ‘as is’ in a program.

```
3.14159             // A floating point literal
108                 // An integer literal
2.99e8              // A floating point literal represented in scientific notation
“A string literal”  // A string literal
‘Another string literal’
true                // A boolean literal
false               // Another boolean literal
/javascript/gi      // A regular expression literal
{key: value}        // An object literal
[1,2,3,4,5]         // An array literal
```

JavaScript - Identifiers and Keywords

An identifier is a just a name.

Identifiers must begin with **letter(a-z A-Z)** or **underscore**(_) or **the dollar sign**(**$**)**.** Subsequent characters can be any among these in addition to **digits(0-9)**.

Certain identifiers are reserved by the language. These are called **keywords**.

Some keywords are:

ES5			**ES6 Onwards**
var	new	true	class
do	null	false	super
while	try		const
for	catch		let
if	finally		yield
else	return		

JavaScript - Expressions and Operators

- Expression is a phrase that when evaluated produces a value.
- Simple expressions are formed by literals or constants. Complex expressions uses operators and operands.
- Operators can be numeric, logical, relational or language specific.

JavaScript - Expressions and Operators

- Expression is a phrase that when evaluated produces a value.
- Simple expressions are formed by literals or constants. Complex expressions uses operators and operands.
- Operators can be numeric, logical, relational or language specific.

JavaScript - Operators

Unary operators : Unary Plus, Minus e.g. +a, -b

Binary operators :
Increment and decrement operators with pre and post variants: ++, --
Arithmetic Operators: Addition + , Subtraction - , Multiplication *, Division /
Logical Operators: And &&, Or ||, Not !
Bitwise Operators: Bitwise And &, Or |, Not ~, Shift Left <<, Shift Right >>
Relational Operators: <, >, <=, >=, ==, !=, ===, !==

Ternary operator : ? : a ? b : c

Language operators :
typeof : Determine type of operand
delete : Remove a property from an object
instanceof : Tests class of an object
in : Test property membership in object
void : Returns undefined value

JavaScript - Operators

Operator	Description	Example	Result	
+	Addition	x=y+2	x=7	y=5
-	Subtraction	x=y-2	x=3	y=5
*	Multiplication	x=y*2	x=10	y=5
/	Division	x=y/2	x=2.5	y=5
%	Modulus (division remainder)	x=y%2	x=1	y=5
++	Increment	x=++y	x=6	y=6
		x=y++	x=5	y=6
--	Decrement	x=--y	x=4	y=4
		x=y--	x=5	y=4

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

JavaScript - Comparators

Unary operators : Unary Plus, Minus e.g. +a, -b

Binary operators :
Increment and decrement operators with pre and post variants: ++, --
Arithmetic Operators: Addition + , Subtraction - , Multiplication *, Division /
Logical Operators: And &&, Or ||, Not !
Bitwise Operators: Bitwise And &, Or |, Not ~, Shift Left <<, Shift Right >>
Relational Operators: <, >, <=, >=, ==, !=, ===, !==

Ternary operator : ? : a ? b : c

Language operators :
typeof : Determine type of operand
delete : Remove a property from an object
instanceof : Tests class of an object
in : Test property membership in object
void : Returns undefined value

JavaScript - Comparators

Operator	Description	Example
==	is equal to	x==8 is false x==5 is true
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

JavaScript - Statement

- Statements are JavaScript commands that when executed has some side effect on the environment.
- Statements are terminated by semicolon.
- Statements can be expression statements or declarative statements
- Statements can be categorized as

  Conditional: if switch

  Loop: while for

  Jumps: break return throw

- Statements can be blocked using curly brackets aka curlies, braces - { }

```
{
        var a = 10;
                console.log('This is a statement block.');
}
```

JavaScript - Declaration Statements

- The **var** and **function** are declaration statements.
- They declare or define variables and functions.
- These statements define identifiers (variables and function names) for use in program.

JavaScript - Declaration Statement - var

var declares a variable or many variables.

To declare a variable

var *<identifier>*;

To declare and initialize a variable

var *<id>* = *<value>*;

To assign a variable

<id> = *<value>*;

- We can declare one variable at a time

  var greeting = “We are learning JavaScript”;

- Or many variables in single statement

  var a = 10, statement = ‘Ignorance is sin; cognizance divine’, b = a * 2 ;

- We can also use var to declare functions

  var square = function(number) { return number * number } ;

JavaScript - Declaration Statement - function

- **function** is used to define functions.
- Function declaration has the form

```
function functionName(arg1,  arg2, arg3, […arg4]) {

}
```

- Function declaration statement can be used in expression form to be initialized to a variable. Such an expression is called function expression

```
var f = function functionName(arg1,  arg2, arg3, […arg4]) {

}
```

- A function expression can define a nameless function. Such a definition is called anonymous function.

```
var f = function (arg1,  arg2, arg3, […arg4]) {

}
```

JavaScript - Conditional Statement

- Conditional statements execute or skip statements depending on value of specified statement.
- These define decision points known to create branches in code.
- The more the branches in code, the more the cyclomatic complexity.
- We should strive for less cyclomatic complexity of code.

JavaScript - Conditional - if

- IF Clause

```
if (expression)
    statement
```

- IF ELSE Clause

```
if (expression)
    statement
else
    statement
```

- IF ELSE IF ELSE Clause

```
if (expression)
    statement
else if (expression)
    statement
else
    statement
```

JavaScript - Conditional - switch

- Switch is another conditional statement used when all branches depend on value of same expression.

```
switch(expression) {

  case expression:
            statement;
            break;

  case expression:
            statement;
            break;

  default:
            statement;
            break;
}
```

JavaScript - Loops - while and do while

- While statement is basic looping statement.
- There are two variants - **while** and **do while**

```
while (expression)
  statement
```

- Do-while

```
do
    statement
while (expression);
```

JavaScript - Loops - for

- **for** is more convenient looping statement.

```
for(initialize; test; increment)
   statement
```

- For in terms of while

```
initialize
while(test){
        statement;
        increment;
}
```

JavaScript - Jumps - break and continue

- Jump statements cause the runtime to jump to new location in source code.
- Break statement causes innermost loop or switch statement to exit immediately.

```
for (i = 0; i< data.length; i++) {
        if(data[i] == “key”){
                isFound = true;
        break;
    }
}
```

- Continue restarts the loop at next iteration

```
for (i = 0; i< data.length; i++) {
        if(!data[i]) continue;
  total += data[i];
}
```

JavaScript - Jumps - return

- Return causes the runtime to jump to the point from where it was called.
- Inside functions, it is used to specify the value of function invocation.

```
function isEven(candidate) {
        var isEven;
        if(candidate % 2 == 0)
                isEven = true;
        else
                isEven = false;
        return isEven;
}

function total(marks) {
        if(!marks)
                return;  // can return null, or 0 or any value
        for(var total = 0, i = 0, j= marks.length, i < j, i++)
                total += marks[i];
        return total
}
```

JavaScript - Jumps - throw

- Throw is similar to return, except it is used to signal an exception or error in the code.
- It also causes the runtime to jump to the point from where it was called.
- Exceptions thrown are taken care of by handlers defined using try/catch/finally statements.

```
function factorial(x) {
    // If the input argument is invalid, throw an exception!
    if (x < 0) throw new Error("x must not be negative");
    // Otherwise, compute a value and return normally
    for(var f = 1; x > 1; x--) f *= x;
    return f;
}
```

JavaScript - try catch finally

- Try Catch Finally is the exception handling mechanism of JavaScript.
- Try is used to define a block of code whose exceptions are to be handled.
- Try is followed by catch which contains code to be executed in case of exceptions.
- Catch is followed by finally which contains clean up code - executed irrespective of what happens in try.

```
try {

}
catch (e) {

}
finally {

}
```

JavaScript - Objects

- Objects are fundamental data type of the language.
- It is unordered collection of properties - a name value pair.
- Objects are composite type container - can contain primitives as well as other objects.
- Objects provide access for storage and retrieval via named string identifiers.
- Objects are dynamic in nature - properties can be added or deleted at run time.

JavaScript - Creating Objects

- Easiest way is to create via object literals.

```
var empty = {};
var point = {x: 10, y: -4};
var sessionDetails = {
        “topic”: “JavaScript Language”,
        instructors: [‘Komal Jain’, ‘RT’]
}
```

- Objects can be constructed via constructors with **new**

```
var o = new Object();
var a = new Array(4);
var d = new Date();
```

- Yet another way is via **Object.create**

```
var o = Object.create(null);
```

JavaScript - Objects - Querying Properties

- To query a property, use the dot(.) or square brackets([]) operator.

```
var intern = { name : 'Abc', college: 'Xyz'};
var internName = intern.name;
var internCollege = intern['college']
```

- We can create or set property values in same way as variable assignment

```
intern.knownLanguages = ['English', 'Hindi', 'Punjabi'];
intern['has-knowledge-of-programming-languages'] = ['C', 'Java'];
```

- We can query all the properties of an object using the for-in loop

```
for(var key in object) {
  console.log("Key = "+ key +", Value = "+ object[key]);
}
```

- If a missing property is accessed, we get **undefined**

...contd

JavaScript - Objects - Querying Properties

- We can delete properties of an object using the **delete** operator.

```
delete intern['has-knowledge-of-programming-languages'];
```

- We can test for the presence of properties using the **in** operator

```
"name" in intern
intern.hasOwnProperty('college')
```

```
function copyProperties(target, source) {
  for(prop in source) {
    target[prop] = source[prop];
  }
  return target;
}
```

- Common object methods are **toString**, **hasOwnProperty**, **valueOf** etc.

JavaScript - Arrays

- An array is an ordered collection of values which are index accessible.
- Arrays are untyped in JavaScript just as objects.
- These are dynamic as well in the sense they can grow and shrink at runtime.

- ***Creating Arrays***
- **Literal form**

```
var empty = [];
var primes = [2,3,5,7,11,13]
var misc = ['string', 123, true, {}, [], null]
```

- **Using Array constructor**

```
var a = new Array(); // creates empty Array
var b = new Array(5); // array with length of 5
var c = new Array(3,4,5) // Array with 3 elements - 3,4,5
```

JavaScript - Arrays - Common Operations

- **Querying elements**
- Array members are accessed via the square brackets notation with corresponding index.

```
misc[2] = false;
var evenPrime = primes[0];
```

- **Array length**
- The length of an array is the number of elements it can accommodate at any given time.

```
misc.length // 6
```

- **Adding elements**
- Elements can be added using index position

```
var a = [];
a[0] = ‘zero’ ;
a[1] = ‘one’ ;
```

- Elements can be stuffed in last using the push method

```
a.push(‘three’);
a.push(‘four’);
```

JavaScript - Common Array Methods

- **join**: Joins array elements using provided separator and returns string.
- **reverse**: Reverse the order of elements in the array.
- **sort**: Sorts array elements in place using default sorting for elements.
- **concat**: Creates and return new array that contains concatenation of arrays on which it is invoked
- **slice**: Returns a slice(subarray) of the array.
- **splice**: General purpose method for insertion and removal.
- **push**: Appends one or more elements at the end of the array.
- **pop**: Deletes the last element of the array, and returns the element.
- **shift**: Removes and return first element of array
- **unshift**: Adds one or more elements to the beginning of the array.
- **indexOf**: Returns the index of specified element in the array, otherwise returns -1.

JavaScript - Advanced Array Method

- Following are some advanced array methods that take into consideration a callback. We will learn about callbacks in coming sessions and thus will revisit them in detail.
- **forEach**: Provides general iteration over arrays
- **map**: Used for transformation purposes, maps each element of array to a function for desired output
- **filter**: Used to filter array based on a predicate
- **reduce**: Combine array elements to reduce to a single value.
- **every**: Checks if every array element satisfies the given predicate.
- **some**: Checks if some(one or more) elements satisfy the given predicate.

JavaScript - Functions

```
function sayHello() {
  console.log('Hello Interns');
}
```

prints “Hello interns” to the console
invoke as - sayHello();

```
function distance(x1, y1, x2, y2) {
  var dx = x2 - x1;
  var dy = y2 - y1;
  return Math.sqrt(dx*dx + dy*dy);
}
```

a normal function statement
invoke as - distance(0, 0, 5, 5);

```
function factorial(n) {
  if(n <= 1) return 1;
  return n * factorial(n - 1);
}
```

a recursive function statement
invoke as - factorial(4)

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

a named function expression
invoke as - sum(3,5) or add(6, 7);

```
var product = function(a, b) {
  return a * b;
}
```

an anonymous function expression
invoke as - product(4,6);

JavaScript - Utilities - Math

- Math is a global utility provided by the runtime for mathematical operations.
- Common operations are:
  Math.sqrt - For finding square root
  Math.pow - For raising a number to a power
  Math.round - For rounding a number
  Math.floor - For finding floor value of a decimal number

  etc

JavaScript - Utilities - Date

- JavaScript dates are time values that are number of milliseconds since January 1, 1970 UTC.
- There is no literal form for dates. To create date values, always use the Date constructor.

```
new Date();
new Date(value);
new Date(dateString);
new Date(year, month[, date[, hours[, minutes[, seconds[,
milliseconds]]]]]);
```

JavaScript - Utilities - JSON

- JSON stands for **J**ava**S**cript **O**bject **N**otation.
- It is a format for data interchange via serialization and deserialization.
- Its representation is similar to Object literal notation with some variations
- Runtime provides parsing and serializing facilities for JSON data through JSON.parse and JSON.serialize functions from string to object and vice-versa respectively.
- Sample JSON

```
{
        "id": 1,
        "name": "A green door",
        "price": 12.50,
        "tags": ["home", "green"]
}
```

JavaScript - Utilities - Dialog-Boxes

Three types of dialog boxes:

- Alert Box
  - `alert('Welcome to TTN ...');`
- Prompt Box
  - `var myName = prompt('Please enter your name.');`
- Confirm Box
  - `Confirm('Are you sure you wanna proceed ?');`

Thanks

Queries

Doubts

Discussion

Questions

- Prompt for amount, interest rate and no. of years and calculate simple interest.
- is palindrome string
- Area of circle
- Copy information of one object to another and log it to console.
- create a list of objects of Employee with info as follow :
    a. Name, age, salary ,DOB
    b. filter all employees with salary greater than 5000
    c. group employee on the basis of their age
    d. fetch employees with salary less than 1000 and age greater than 20. Then give them an increment 5 times their salary.