**JavaScript Programming**

1. **JS Introduction**
2. *JavaScript is one of the 3 languages all web developers must learn*
3. *JavaScript was invented by Brenan Eich in 1995.*
4. *ECMAScript is the official name of the JavaScript.*
5. *JavaScript is designed by c++ language.*
6. **JS Console - Debugging, Testing, Logging**
7. *console.log(loggedCode)*
8. *console.error(errorCode)*
9. *console.dir(loggedCodeDetails)*
10. **JS Statements**
11. *A JavaScript statement is a line or block of code that instructs the JavaScript interpreter or browser engine to perform a specific action or operation.*
12. *JavaScript statements are separerated(not closed) by semicolon (;).*
13. *JavaScript statements are composed of: values, operators, expressions, keywords,comments, variables.*
14. **JS Expressions**
15. *An expression is a combination of values, variables, and operators, which computes to value.*
16. *Js evalutes expressions from left to right.*[*See here*](#_toc162)
17. **JS Keywords**
18. *JavaScript keywords are used to indentify actions to be performed.*
19. *Keywords are called reserved words*
20. **JS Comments**
21. */\*comments\*/*
22. *//comments*
23. **JS Case Sensitive**
24. *all javascript identifiers are case sensitive.*
25. **JS Values**
26. *number, string, boolean, undefined, null, symbol, object, array, function, NaN, Infinity, -Infinity*
27. *let uname = “Sobuj”*
28. **JS Errors**
29. *js main errors:*
30. *syntax error: syntax errors occur when the code violates the language’s grammatical rules.*
31. *runtime error: runtime errors occur during the execution of a program when something unexpected happens.*
32. *this error handling using* [*try..catch*](#_toc140) *block.*
33. *logical(bug) error: logical errors are the most challenging to identify and fix because they don’t cause the program to crash or throw errors. Instead, they lead to incorrect program behavior or unexpected result.*
34. *js specific errors:*
35. *reference error: reference error is a sepecific type of runtime error that occurs when you try to access a variable of function that has not been declared or is not in scope.*
36. *type error: type error is a specific type of runtime error that occurs when an operations is performed on a value that is of an inappropriate or unexpected data type.*
37. **JS Camel Case**
38. *javascript programmers tend to use camel case that starts with lowercase letter.*
39. **JS Identifiers/Names**
40. *idenfifiers are javascript names*
41. *identifiers rules:*
42. *identifiers must begin with letter(A-Za-z), dollar sign($) or underscore(\_).*
43. *Numbers are not allowed as the first character in identifiers.*
44. **JS Variables**
45. *javascript variables can be declared in 4 ways:*
46. *automatically: uname= “Sobuj”*
47. *var: var uname= “Sobuj”*
48. *let: let unme= “Sobuj”*
49. *const: const uname= “Sobuj”*
50. *the var keyword was used in all JavaScript code from 1995 to 2015.*
51. *the let and const keywords were added to javascript in 2015.*
52. *automatically threats as var.*
53. *var,let,const charactersistics:*
54. *var: redefined,reassigned,global-scope*
55. *let: not-redefined, reassigned, block-scope*
56. *const: not-redefined, not-reassigned, local-scope*
57. *var, let, const usecases:*
58. *always use const if the value should not be changed.*
59. *always use const if the type should not be changed(arrays and objects).*
60. *only use let if you can’t use const.*
61. *only use var if you must support old browser.*
62. **JS Operators**
63. *operators are symbols or special keywords that are used to perform various operations on values and variables.*
64. *Javascript operators:*
65. *arithmetic operators: addition(+),subtraction(-),multiplication(\*),*[*exponentiation(\*\*)*](#_toc148)*,diviision(/),modulus(%),increment(++),decrement(--)*
66. *assignment operators: assignment(=),addition assignment(+=),subtraction assignment(-=),multiplication assignment(\*=),division assignment(/=),modulus assignment(%=),exponentiation assignment(\*\*=);*
67. *comparison operators: equal to(==),equal value and type(===),not equal(!=),not equal and type(!==),greater than(>),less than(<),greater than or equal to(>=),less than or equal to(<=);*
68. *logical operators: logical and(&&),logical or(||), logical not(!);*
69. *type operators: typeof,* [*instanceof*](#_toc155) *;*
70. *ternary operator: conditon ? output : output ;*
71. **JS Data Types**
72. *js data types:*
73. *string,number,bigint,boolean,undefined,null,symbol,object*
74. *object data type can contain: object,array,date,function;*
75. *primitive vs reference data type:*
76. *primitive: primitive types are single values that can not be changed after declared.*
77. *Primitive data are store in a call stack memory.*
78. *String,number,bigint,boolean,null,undefined,symbol are primitive data type.*
79. *Call stack means every declare time create a new memory space to store primitive data.*
80. *Reference: reference types are more complex data types that represent collection of data or objects.reference types are mutable.*
81. *Object, function, array,date are reference data type.*
82. *Refernce data are store in a heap memory.*
83. *Heap memory means root declare data are always store in heap memory as single and identifiers store in call stack identifiers always refer heap memory data.*
84. **JS Strings**
85. *js strings is declared within the single or double quotes;*
86. *Escape characters: \’,\”,\\,\n,b;*
87. *string as object: new String(“”);*
88. *string methods:*
89. *charAt(index);charCodeAt(index);concat(str1,str2…);endsWith(‘character’,optional[serialNo]);startsWith(‘character’,optional[serialNo]);String.fromCharCode(97,97,97);includes(“str”,strtIndex);indexOf(‘str’,startIndex);lastIndexOf(‘str’,startIndex);localeCompare();match(reqx);padEnd(length,str);padStart(length,str);repeat();replace();replaceAll();search();slice(startIndex,endIndex);split(separeator,limit);startsWith();substr();substring();toLocaleLowerCase();toLocaleUpperCase(); toLowerCase();toUpperCase();toString();trim();trimEnd();trimStart();valueOf();*
90. *string properties:*
91. *length;constructor;prototype;*
92. **JS Numbers**
93. *js has only one type of number;*
94. *js numbers global: NaN,isNaN(),Infinity,-Infinity; ...more;*
95. *js numbers as object: new Number(123);*
96. *js numbers method:*
97. *toString(base); toLocaleString(“bn-BD”); toFixed(); toPrecision(totalDigits); toExponential(); valueOf();*
98. **JS BigInt**
99. *js bigint variables are used to store big integer values that are too big to be represented by a normal javascript number.*
100. *Let score=232333n; bounce=BigInt(123213123123213123123);*
101. *BigInt();*
102. **JS Boolean**
103. *a javascript boolean represents one of two values: true or false;*
104. *Boolean();*
105. *everything with a “value” is true:*
106. *100; -12; “something”; true;*
107. *everything without a “value” is false:*
108. *0; -0; “”; undefined; null; NaN; false;*
109. *js booleans as object: new Boolean();*
110. **JS Symbol**
111. *symbol is a built-in object whose constructor returns a symbol primitive also called a symbol value or just a symbol that’s guaranteed to be unique.*
112. *Symbol()*
113. *symbol as object property:{[Symbol()]: “good”}*
114. **JS Truths Talking**
115. *comparing javascript two reference(objects) data types always return false;*
116. *Comparing javascript two primitive data types always return true;*
117. **JS Globals Codes**
118. *isNaN();isFinite();Number.isInteger();Number.isSafeInteger();parseInt();parseFloat();*
119. *Number.MAX\_SAFE\_INTEGER;Number.MIN\_SAFE\_INTEGER;Number.MAX\_VALUE;Number.MIN\_VALUE;Number.POSITIVE\_INFINITY;Number.NEGETIVE\_INFINITY;Number.NaN;*
120. *BigInt(number);Boolean(condition);Symbol();*
121. *new String();new Number(); new Boolean();*
122. *String.prototype; Number.prototype; Boolean.prototype;*

**Example Code Blocks**

# [try….catch statemnets](#try..catch_block)

try{

}catch(error){

}

# [Exponentiation operator](#exponentiation_operator)

const n1 = 3;

const n2 = 3;

const result = n1\*\*n2;

console.log(result);

# [**Instanceof operator**](#instanceof_operator)

class Person{

}

const p1 = new Person();

console.log(p1 instanceof Person);

# [**Evaluates expression**](#evalutes_expression)

let x = 16 + 4 + "Volvo";