JAVA SCRIPT DOCUMENTATION

*console.log("This is my first java script file")*

*console.log("Lets start documenting all the basics of java script")*

**Variables**

\* Variables are containers for storing data values

\* let, const, var

\* let is used to declare a block scope local variable, optionally initializing it to a value

**/\* LET \*/**

*let name = "John Doe"   \* string*

*let age = 30          \* number*

*let isStudent = true   \* boolean*

*let hobbies = ["reading", "gaming", "coding"] \* array*

\* objects are not a primitive data type, but they are a complex data type

\* objects are collections of key-value pairs

\* objects are used to store multiple values in a single variable

\* objects are mutable, meaning they can be changed after they are created

*let address = {        \* object*

*street: "123 Main St",*

*city: "New York",*

*state: "NY"*

*}*

\* null is a special value that represents the absence of any value or object

\* null is an object type in JavaScript, but it is not a valid object

\* null is a primitive data type in JavaScript, but it is not a valid primitive value

\* null is often used to indicate that a variable has been declared but has not yet been assigned a value

*let nullValue = null  \* null*

\* undefined is a special value that represents the absence of a value or object

\* undefined is a primitive data type in JavaScript, but it is not a valid primitive value

\* undefined is often used to indicate that a variable has not yet been assigned a value

\* undefined is the default value of a variable that has been declared but not yet assigned a value

\* undefined is also the default value of a function that does not return a value

*let undefinedValue; \* undefined*

\* bigint is a special type of number that can represent integers with arbitrary precision

\* bigint is a primitive data type in JavaScript, but it is not a valid primitive value

\* bigint is often used to represent large integers that cannot be represented by the number type

\* bigint is created by appending "n" to the end of an integer literal

*let bigIntValue = 1234567890123456789012345678901234567890n \* bigint*

\* symbol is a special type of object that is used to create unique identifiers for object properties

\* symbol is a primitive data type in JavaScript, but it is not a valid primitive value

\* symbol is often used to create unique identifiers for object properties

\* symbol is created by calling the Symbol() function

\* symbol is often used to create unique identifiers for object properties

*let symbolValue = Symbol("unique") \* symbol*

\* function is a special type of object that is used to create reusable blocks of code

\* function is a first-class object in JavaScript, meaning it can be treated like any other object

\* function is a primitive data type in JavaScript, but it is not a valid primitive value

\* function is often used to create reusable blocks of code

\* function is created by calling the function keyword followed by a name and a set of parentheses

*let functionValue = function() { \* function*

*console.log("This is a function")*

*}*

\* date is a special type of object that is used to represent dates and times

\* date is a first-class object in JavaScript, meaning it can be treated like any other object

\* date is a primitive data type in JavaScript, but it is not a valid primitive value

\* date is often used to represent dates and times

*let dateValue = new Date() \* date*

\* regex is a special type of object that is used to represent regular expressions

\* regex is a first-class object in JavaScript, meaning it can be treated like any other object

\* regex is a primitive data type in JavaScript, but it is not a valid primitive value

\* example of regex is used to match a pattern in a string

\* regex is created by calling the RegExp() function or by using the /pattern/ syntax

\* example : /[a-z]/ is a regex that matches any lowercase letter from a to z

*let regexValue = /[a-z]/ \* regex*

*console.log(name, age, isStudent, hobbies, address, nullValue, undefinedValue, bigIntValue, symbolValue, functionValue, dateValue, regexValue)*

\* to check data type of a variable

*console.log(typeof name) \* string*

*console.log(typeof age) \* number*

*console.log(typeof isStudent) \* boolean*

*console.log(typeof hobbies) \* object*

*console.log(typeof address) \* object*

*console.log(typeof nullValue) \* object*

*console.log(typeof undefinedValue) \* undefined*

*console.log(typeof bigIntValue) \* bigint*

*console.log(typeof symbolValue) \* symbol*

*console.log(typeof functionValue) \* function*

*console.log(typeof dateValue) \* date*

*console.log(typeof regexValue) \* regex*

**/\* CONSTANTS \*/**

\* Constants are variables that cannot be changed after they are declared

\* Constants are declared using the const keyword

*const PI = 3.14 \* constant*

*const MAX\_VALUE = 100 \* constant*

*const MIN\_VALUE = 0 \* constant*

*console.log(PI, MAX\_VALUE, MIN\_VALUE)*

**/\* VAR \*/**

\* Var is used to declare a variable that can be changed after it is declared

\* Var is used in earlier versions of JavaScript, but it is not recommended for use in modern JavaScript

*var nameVar = "John Doe" \* string*

*var ageVar = 30 \* number*

*var isStudentVar = true \* boolean*

*var hobbiesVar = ["reading", "gaming", "coding"] \* array*

*var addressVar = { \* object*

*street: "123 Main St",*

*city: "New York",*

*state: "NY"*

*}*

*console.log(nameVar, ageVar, isStudentVar, hobbiesVar, addressVar)*

*console.log(typeof nameVar) \* string*

*console.log(typeof ageVar) \* number*

*console.log(typeof isStudentVar) \* boolean*

*console.log(typeof hobbiesVar) \* object*

*console.log(typeof addressVar) \* object*

**\* Example of objects**

*let strawhat = {*

*captain : "luffy",*

*crew : "strawhat",*

*rightHandMan : "zoro",*

*leftHandMan : "sanji",*

*otherCrew : ["nami", "robin", "franky", "brook", "chopper", "usopp"],*

*ship : "thousand sunny",*

*}*

\* There are two ways to access the properties of an object

\* 1. Dot notation

\* 2. Bracket notation

*console.log (strawhat.captain)*

*console.log (strawhat["crew"]) \* strawhat*

*console.log (strawhat["rightHandMan"]) \* zoro*

\* Note : To execute java script code in vs code console, we can use node and run the file using the

\* command "node filename.js"

\* If we need to make it live, we can use nodemon and run the file using the

\* command "nodemon filename.js"

\* To install nodemon, we can use the command "npm install -g nodemon"

\* adding values to the object

\* There are two ways to add values to the object

\* 1. Dot notation

\* 2. Bracket notation

*strawhat.friends = ["traffy", "bon clay", "ace", "shanks", "buggy", "vivi"] \* adding new value to the object*

*strawhat["ship"] = "thousand sunny"*

*console.log(strawhat.friends) \* new value*

*console.log(strawhat["ship"]) \* thousand sunny*

*console.log (strawhat)*

**Operators and Conditional Statements**

*console.log("lets start learning about operators and conditional statements");*

/\* COMMENTS IN JS \*/

\* Comments are used to explain the code and make it more readable

\* comments are ignored by the compiler and are not executed

\* There are two types of comments in JS

\* Single line comments are used to comment a single line of code

\* Multi line comments are used to comment multiple lines of code

\*

\* Single line comments are declared using "\* any message "

\* Multi line comments are declared using "/\* any message \*/"

\* Example of single line comments

\* console.log("Hello World") \* this is a single line comment

\* Example of multi line comments

/\*

 console.log("Hello World") \* this is a multi line comment

\*/

/\* OPERATORS IN JS \*/

\* Operators are used to perform operations on variables and values

\* There are different types of operators in JS

\* 1. Arithmetic operators

\* 2. Assignment operators

\* 3. Comparison operators

\* 4. Logical operators

\* 5. Bitwise operators

\* 6. Ternary operators

\* 7. Type operators

\* 8. Unary operators

\* 9. Relational operators

\* 10. Conditional operators

\* 11. Nullish coalescing operators

\* 12. Spread operators

\* 13. Rest operators

/\* ARITHMETIC OPERATORS \*/

\* Arithmetic operators are used to perform arithmetic operations on variables and values

\* There are several types of arithmetic operators in JS

\* 1. Addition operator (+) : This operator is used to add two or more numbers together. For example, 2 + 3 = 5

\* 2. Subtraction operator (-) : This operator is used to subtract one number from another. For example, 5 - 2 = 3

\* 3. Multiplication operator (\*) : This operator is used to multiply two or more numbers together. For example, 2 \* 3 = 6

\* 4. Division operator (/) : This operator is used to divide one number by another. For example, 6 / 2 = 3

\* 5. Modulus operator (%) : This operator is used to find the remainder of a division operation. For example, 5 % 2 = 1

\* 6. Exponentiation operator (\*\*): This operator is used to raise a number to the power of another number. For example, 2 \*\* 3 = 8

\* 7. Increment operator (++): This operator is used to increase the value of a variable by 1. For example, x++

\* 8. Decrement operator (--): This operator is used to decrease the value of a variable by 1. For example, x--

\* Example of arithmetic operators

*console.log("Arithmetic Operators")*

*console.log(2 + 3)*

*console.log(5 - 2)*

*console.log(2 \* 3)*

*console.log(6 / 2)*

*console.log(5 % 2)*

*console.log(2 \*\* 3)*

*let x = 5*

*console.log(x++)*

*console.log(x)*

*console.log(x--)*

*console.log(x)*

*console.log(++x)*

*console.log(x)*

*console.log(--x)*

*console.log(x)*

*console.log(2 + 3 \* 4)*

*console.log((2 + 3) \* 4)*

*console.log(2 + 3 - 4)*

/\* Difference between ++a and a++ \*/

\* The difference between ++a and a++ is that ++a increments the value of a before it is used in an expression, while a++ increments the value of a after it is used in an expression. For example:

*let a = 5*

*console.log("Difference between ++a and a++")*

*console.log(++a)*

*console.log(a)*

*let b = 5*

*console.log(b++)*

*console.log(b)*

\* console.log(a--)

/\* ASSIGNMENT OPERATORS \*/

\* Assignment operators are used to assign values to variables

\* There are several types of assignment operators in JS

\* 1. Assignment operator (=) : This operator is used to assign a value to a variable. For example, x = 5

\* 2. Addition assignment operator (+=) : This operator is used to add a value to a variable and assign the result to the variable. For example, x += 5 is equivalent to x = x + 5

\* 3. Subtraction assignment operator (-=) : This operator is used to subtract a value from a variable and assign the result to the variable. For example, x -= 5 is equivalent to x = x - 5

\* 4. Multiplication assignment operator (\*=) : This operator is used to multiply a variable by a value and assign the result to the variable. For example, x \*= 5 is equivalent to x = x \* 5

\* 5. Division assignment operator (/=) : This operator is used to divide a variable by a value and assign the result to the variable. For example, x /= 5 is equivalent to x = x / 5

\* 6. Modulus assignment operator (%=) : This operator is used to find the remainder of a division operation and assign the result to the variable. For example, x %= 5 is equivalent to x = x % 5

\* 7. Exponentiation assignment operator (\*\*=) : This operator is used to raise a variable to the power of a value and assign the result to the variable. For example, x \*\*= 5 is equivalent to x = x \*\* 5

\* 8. Bitwise AND assignment operator (&=) : This operator is used to perform a bitwise AND operation on a variable and assign the result to the variable. For example, x &= 5 is equivalent to x = x & 5

\* 9. Bitwise OR assignment operator (|=) : This operator is used to perform a bitwise OR operation on a variable and assign the result to the variable. For example, x |= 5 is equivalent to x = x | 5

\* 10. Bitwise XOR assignment operator (^=) : This operator is used to perform a bitwise XOR operation on a variable and assign the result to the variable. For example, x ^= 5 is equivalent to x = x ^ 5

\* 11. Left shift assignment operator (<<=) : This operator is used to perform a left shift operation on a variable and assign the result to the variable. For example, x <<= 5 is equivalent to x = x << 5

\* 12. Right shift assignment operator (>>=) : This operator is used to perform a right shift operation on a variable and assign the result to the variable. For example, x >>= 5 is equivalent to x = x >> 5

\* 13. Unsigned right shift assignment operator (>>>=) : This operator is used to perform an unsigned right shift operation on a variable and assign the result to the variable. For example, x >>>= 5 is equivalent to x = x >>> 5

\* 14. Nullish coalescing assignment operator (??=) : This operator is used to assign a value to a variable if the variable is null or undefined. For example, x ??= 5 is equivalent to x = x ?? 5

\* 15. Logical AND assignment operator (&&=) : This operator is used to perform a logical AND operation on a variable and assign the result to the variable. For example, x &&= 5 is equivalent to x = x && 5

\* 16. Logical OR assignment operator (||=) : This operator is used to perform a logical OR operation on a variable and assign the result to the variable. For example, x ||= 5 is equivalent to x = x || 5

\* 17. Optional chaining assignment operator (?.=) : This operator is used to assign a value to a variable if the variable is not null or undefined. For example, x ?.= 5 is equivalent to x = x ?. 5

\* 18. Optional chaining nullish coalescing assignment operator (???.=) : This operator is used to assign a value to a variable if the variable is not null or undefined. For example, x ???= 5 is equivalent to x = x ??? 5

\* Example of assignment operators

*console.log("Assignment Operators");*

*let c = 5 \* assignment operator*

*c += 4 \* addition assignment operator*

*console.log(c) \* 9*

/\* COMPARISON OPERATORS \*/

\* Comparison operators are used to compare two values and return a boolean value (true or false)

\* There are several types of comparison operators in JS

\* 1. Equal to operator (==) : This operator is used to compare two values for equality. For example, 2 == 2 is true

\* 2. Strict equal to operator (===) : This operator is used to compare two values for equality and type. For example, 2 === "2" is false

\* 3. Not equal to operator (!=) : This operator is used to compare two values for inequality. For example, 2 != 3 is true

\* 4. Strict not equal to operator (!==) : This operator is used to compare two values for inequality and type. For example, 2 !== "2" is true

\* 5. Greater than operator (>) : This operator is used to compare two values and return true if the left value is greater than the right value. For example, 2 > 1 is true

\* 6. Less than operator (<) : This operator is used to compare two values and return true if the left value is less than the right value. For example, 2 < 3 is true

\* 7. Greater than or equal to operator (>=) : This operator is used to compare two values and return true if the left value is greater than or equal to the right value. For example, 2 >= 2 is true

\* 8. Less than or equal to operator (<=) : This operator is used to compare two values and return true if the left value is less than or equal to the right value. For example, 2 <= 3 is true

\* Example of comparison operators

*console.log("Comparison Operators") ;*

*console.log(2 == 2) \* true*

*console.log(2 == "2") \* false*

*console.log(2 === "2") \* false*

*console.log(2 != 3) \* true*

*console.log(2 !== "2") \* true*

*console.log(2 > 1) \* true*

*console.log(2 < 3) \* true*

*console.log(2 >= 2) \* true*

*console.log(2 <= 3) \* true*

/\* LOGICAL OPERATORS \*/

\* Logical operators are used to perform logical operations on boolean values

\* There are several types of logical operators in JS

\* 1. Logical AND operator (&&) : This operator is used to perform a logical AND operation on two boolean values. For example, true && false is false

\* 2. Logical OR operator (||) : This operator is used to perform a logical OR operation on two boolean values. For example, true || false is true

\* 3. Logical NOT operator (!) : This operator is used to perform a logical NOT operation on a boolean value. For example, !true is false

\* Example of logical operators

*console.log("Logical Operators");*

*console.log(true && false) \* false*

*console.log(true || false) \* true*

*console.log(!true) \* false*

/\* CONDITIONAL STATEMENTS \*/

\* Conditional statements are used to perform different actions based on different conditions

\* There are several types of conditional statements in JS

\* 1. if statement : This statement is used to execute a block of code if a specified condition is true. For example, if (x > 5) { console.log("x is greater than 5") }

\* 2. if...else statement : This statement is used to execute a block of code if a specified condition is true, and another block of code if the condition is false. For example, if (x > 5) { console.log("x is greater than 5") } else { console.log("x is less than or equal to 5") }

\* 3. else if statement : This statement is used to specify a new condition if the first condition is false. For example, if (x > 5) { console.log("x is greater than 5") } else if (x < 5) { console.log("x is less than 5") } else { console.log("x is equal to 5") }

\* 4. switch statement : This statement is used to execute a block of code based on different cases. For example, switch (x) { case 1: console.log("x is 1") break; case 2: console.log("x is 2") break; default: console.log("x is not 1 or 2") }

\* 1. if statement

*console.log("if statement")*

*let d = 5*

*if (d > 1 ){*

*console.log ("d is greater thann 1")*

*}*

\* example 2:

*let age = 18 ;*

*if (age >= 18){*

*console.log ("can vote")*

*}*

\* 2. if...else statement

*console.log("if...else statement")*

*let e = 5*

*if (e > 1){*

*console.log("e is greater than 1")*

*} else {*

*console.log("e is less than or equal to 1")*

*}*

\* 3. else if statement

*console.log("else if statement")*

*let f = 5*

*if (f > 1){*

*console.log("f is greater than 1")*

*}*

*else if (f < 1){*

*console.log("f is less than 1")*

*} else {*

*console.log("f is equal to 1")*

*}*

\* 4. switch statement

*console.log("switch statement") ;*

*let g = 2;*

*switch (g){*

*case 1:*

*console.log("g is 1")*

*break*

*case 2:*

*console.log("g is 2")*

*break*

*case 3:*

*console.log("g is 3")*

*break*

*default:*

*console.log("g is not 1, 2 or 3")*

*break*

*}*

/\* 5. Ternary operator \*/

\* Ternary operator is a shorthand if statement

\* It is used to assign a value to a variable based on a condition

\* !Syntax: condition ? value1 : value2

\* Example: let x = 5; let y = x > 5 ? "x is greater than 5" : "x is less than or equal to 5";

*console.log("Ternary operator");*

*let h = 7;*

*let i = h > 5 ? "h is greater than 5" : "h is less than or equal to 5";*

*console.log(i);*

**User inputs (prompt and alert)**

\* let username = prompt("Please enter your name: "); \* prompt is used to take input from the user

\* alert("Hello " + username + "! Welcome to the world of JavaScript!"); \* alert is used to display a message to the user

*let score = prompt("Please enter your score:*

*score = parseInt(score);*

*switch (true){*

*case (score >= 90):*

*alert("You got an A!");*

*break;*

*case (score >= 80 && score < 90):*

*alert("You got a B!");*

*break;*

*case (score >= 70 && score < 80):*

*alert("You got a C!");*

*break;*

*case (score >= 60 && score <70):*

*alert("You got a D!");*

*break;*

*case (score >= 50 && score < 60):*

*alert("You got an E!");*

*break;*

*}*