|  |  |  |
| --- | --- | --- |
| JAVASCRIPT Documentation | | |
|  | | |
| TOPIC | EXPLAINATION |
| 1. String concatenation using “+” operator | With the help of “+” operator two strings can be combined.  Example:  let firstName = "Arijit"  let lastName = "Deb"  let fullName = firstName +" "+ lastName  console.log(fullName) |
| 1. String concatenation using concat() function | The concat() method concatenates the string arguments to the calling string and returns a new string. Takes the items to be concatenated as argument separated by comas.  Example:  let firstName = "Arijit"  let lastName = "Deb"  let fullName = firstName.concate(“ ”, lastName)  console.log(fullName) |
|  |  |
| 1. Rules regarding variable names | 1. Can’t define a variable more than once. So, there will be error if:  let name = “Auritra”  let name = “Arijit”  2. Numbers can’t be variable name. So, it is not possible to have:  let 2 = 3  3. Variable names can only be a word or “$” sign before or after the word, or “-” sign before or after the word or word with a number without spacing.  4. Variable names cannot be reserved keywords. |
| 1. === | Equality operator.  \*Can be used for all data types |
| 1. !== | Not equality operator  \*Can be used for all data types |
| 1. < | Less than operator.  \*Can be used for numeric data types |
| 1. <= | Less than equal to operator.  \*Can be used for numeric data types |
| 1. > | Greater than operator.  \*Can be used for numeric data types |
| 1. >= | Greater than equal to operator.  \*Can be used for numeric data types |
| 1. If (true) {} | Code will always execute |
| 1. If (false) {} | Code will never execute |
| 1. Scoping in Javascript | 1. Lexical scoping (Static scoping)  2. There are two types of scope – global scope a scope.  3. Global scope contains all the things outside of all the code blocks.  4. Local scope contains all the things inside a code block. |
| 1. Scoping Rules | 1. In a scope, you can access variables defined within the scope or in any parent/ancestor scope.  Example 2:  let name1 = “Arijit”  if (true) {  let name2 = “Auritra”  if (true) {  let name3 = “Abhi”  console.log(name1)  console.log(name2)  console.log(name3)  }  }  It is possible to access all the variables name1(defined in parent scope which is here the global scope) and name2 (defined inside the parent local scope) and name3 (defined inside the child local scope).  2. Scope of same level (two independent scope) cannot access variables of each other.  Example:  if (true) {  let name1 = “Arijit”  }  if (true) {  let name2 = “Auritra”  }  H  3. Variables with same name can be defined in different scopes.  4. The value of a variable defined in parent scope can be changed in the child scope. This is known as variable shadowing in JavaScript. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |