# **Assignment For Day 2smiling face with smiling eyes1st place medalherb**

### ****1. What is lexical structure?****

**Answer :** Lexical structure is the basic building blocks of JavaScript code. It defines **how the code is written and read** — like grammar in a language. It includes:

* **Keywords** (like if, let, function)
* **Identifiers** (variable names)
* **Operators** (+, -, =)
* **Punctuation** (;, {}, ())
* **Comments** (// or /\* ... \*/)
* **Literals** (like 10, "hello")

It helps the JavaScript engine understand the code correctly.

### ****2. What is Unicode?****

**Answer :** Unicode is a standard way of **representing characters from all languages** (like English, Hindi, Chinese, etc.) using unique codes.  
JavaScript uses Unicode to **store text** so it can support emojis, different scripts, and symbols.  
Example:

* The letter "A" has a Unicode value of U+0041.
* The emoji ****  has a Unicode value of U+1F60A.

### ****3. Explain all the keywords present in JavaScript with examples.****

**Answer :** JavaScript has reserved words called **keywords**, which have special meanings. Here are some commonly used ones:

| **Keyword** | **Meaning** | **Example** |
| --- | --- | --- |
| var | Declares a variable | var x = 10; |
| let | Declares a block-scoped variable | let y = 20; |
| const | Declares a constant variable | const pi = 3.14; |
| if | Conditional statement | if (x > 5) { ... } |
| else | Alternative for if | else { ... } |
| for | Loop | for (let i=0; i<5; i++) {} |
| while | Loop | while (i < 5) {} |
| do | Loop that runs at least once | do { ... } while (x < 5); |
| function | Defines a function | function add() { ... } |
| return | Returns value from a function | return x + y; |
| break | Exits a loop | break; |
| continue | Skips one iteration in a loop | continue; |
| switch | Selects code block to run | switch (value) { ... } |
| case | A condition in switch | case 1: break; |
| default | The fallback in switch | default: break; |
| try | Tries a block of code | try { ... } |
| catch | Catches errors from try | catch(error) { ... } |
| finally | Always runs after try/catch | finally { ... } |
| throw | Throws a custom error | throw "error message"; |
| typeof | Finds the data type | typeof 10; // "number" |
| new | Creates an object from a constructor | new Date(); |
| this | Refers to the current object | this.name |
| delete | Deletes object property | delete obj.key; |
| class | Declares a class | class Car { ... } |
| extends | Inherit from another class | class Bike extends Vehicle |
| super | Calls the parent class constructor | super(); |
| import | Imports modules (in ES6+) | import x from 'file'; |
| export | Exports modules | export default function() {} |
| await | Waits for a Promise (inside async) | await fetch(); |
| async | Declares an asynchronous function | async function getData() {} |

These are just some — there are more, but these are most common.

### ****4. What are shorthand operators? Explain with an example.****

**Answer :** Shorthand operators make assignments easier. Instead of writing full expressions, you can shorten them.

Examples:

js

CopyEdit

let x = 10;

// Instead of: x = x + 5;

x += 5; // x is now 15

// Instead of: x = x - 2;

x -= 2; // x is now 13

// Instead of: x = x \* 2;

x \*= 2; // x is now 26

// Instead of: x = x / 2;

x /= 2; // x is now 13

Common shorthand operators:

* += Add and assign
* -= Subtract and assign
* \*= Multiply and assign
* /= Divide and assign
* %= Modulus and assign

### ****5. What is**** "use strict" ****in JavaScript?****

**Answer :** "use strict" is a special directive in JavaScript that tells the browser to run the code in **strict mode**.  
Strict mode helps catch mistakes and makes your code safer by:

* Not allowing you to use undeclared variables
* Preventing you from deleting protected variables
* Avoiding duplicate parameter names in functions

- Example:

js

CopyEdit

"use strict";

x = 10; // Error: x is not declared

It’s usually written at the top of the file or function.