# Summer Homework – Computer Science JavaScript Notes

## 1. Comments

• // is used for inline comments.  
Example: var pixels = 5; // inline comment  
  
• /\* ... \*/ is used for multi-line comments.  
Example:  
/\*  
 This can be used for  
 multiple lines because multi-line  
 comments work like this.  
\*/  
  
Tip: Use comments to explain what your code is doing.

## 2. Data Types

JavaScript has 7 data types (6 primitive + objects):  
• undefined – a declared variable with no value yet.  
• null – an intentional 'empty' value (nothing).  
• boolean – true or false.  
• string – text inside quotes.  
• symbol – a unique and immutable primitive value.  
• number – integers and decimals (NaN and Infinity are special number values).  
• object – collections of key–value pairs (arrays, functions, etc. are objects).  
  
Examples: var myName = "Laith"; let ourCollege = "BHASVIC"; var age = 10;

## 3. Variables: var, let, const

• var – function‑scoped; available throughout the function it is defined in.  
• let – block‑scoped; only available inside the nearest { } block.  
• const – block‑scoped and cannot be reassigned (use for constants).  
  
Example: const PI = 3.14159;

## 4. Storing Values with Assignment Operators

Semicolons are optional in JavaScript, but using them is good practice.  
Example:  
var a;  
var b = 2;  
a = 7;  
b = a;  
console.log(a); // prints 7

## 5. Initialising Variables & Case Sensitivity

You can declare and assign in one line: var x = 5;  
Practice:  
var A = 15;  
var D = 20;  
var C = 25;  
  
JavaScript is case‑sensitive: A and a are different variable names.  
Use camelCase for variable names, e.g., totalScore, lastName.

## 6. Increment and Decrement

• Increment adds 1: myVar++; (same as myVar = myVar + 1)  
• Decrement subtracts 1: myVar--;  
Example: var myVar = 11; myVar++; myVar--;

## 7. Decimal Numbers & Remainders

• Decimal example: var decimal = 5.7;  
• Remainder operator % returns the remainder after division:  
Example: 11 % 3; // 2

## 8. Escaping Quotes & Escape Sequences

To include quotes inside a string, either escape them with a backslash or use different outer quotes.  
  
Examples:  
var s1 = "I am a "double‑quoted" string";  
var s2 = 'I am a "double‑quoted" string';  
Backticks (`) allow both single and double quotes without escaping:  
var s3 = `He said, "It's fine."`;  
  
Common escape sequences:  
• \' single quote  
• \" double quote  
• \\ backslash  
• \n new line  
• \r carriage return  
• \t tab  
• \b backspace  
• \f form feed

## 9. Strings: Length, Indexing, Immutability

• str.length returns the number of characters in a string.  
• Indexing starts at 0 (zero‑based). str[0] is the first character.  
• Strings are immutable: you cannot change a single character in place; create a new string instead.  
  
Example:  
var firstName = "Adnan";  
var firstLetter = firstName[0]; // 'A'  
var lastLetter = firstName[firstName.length - 1];

## 10. String Concatenation

• Use + to join strings:  
var myStr = "This is the start " + "and this is the end.";  
  
• Use += to append:  
var greeting = "Hello ";  
greeting += "Adnan!";  
console.log(greeting); // Hello Adnan!

## 11. Constructing Strings with Variables & Bracket Notation

Example:  
var myName = "Devu";  
var sentence = "My name is " + myName + "!";  
  
Bracket notation recap:  
• Position 0 is always the first character.  
• Example: var firstInitial = myName[0];

## 12. Function Practice – Word Blanks

Set‑up a function that builds a sentence from words passed in:  
function wordBlanks(myNoun, myAdj, myVerb, myAdverb) {  
 var result = "";  
 result += "The " + myAdj + " " + myNoun + " " + myVerb + " " + myAdverb + ".";  
 return result;  
}  
console.log(wordBlanks("dog", "big", "ran", "quickly"));

## 13. Arrays

• Arrays always start with a square bracket [ ]. Each element is separated by a comma.  
• Every element in the array is accessed by its index (starting from 0).  
  
Example: var myArray = ["Smith", 17];  
  
Nested Arrays  
Example: var myArray = [["Bulls", 23], ["Cod", 32]];  
  
Accessing Arrays  
You can access array elements using their index number.  
Example: var ourArray = [50, 60, 70]; var ourData = ourArray[0]; // 50  
  
Modifying Array Data  
You can change array values directly by assigning new values using the index.  
Example: myArray[0] = 2;  
  
Accessing Multi-Dimensional Arrays  
You can access nested arrays using multiple index values.  
Example: var myArray = [[1,2,3], [4,5,6], [7,8,9]]; var myData = myArray[0][0]; // 1  
  
Manipulating Arrays with push()  
push() adds an element to the end of an array.  
Example: myArray.push(["Hope", 36]);  
  
Manipulating Arrays with pop()  
pop() removes the last element from an array.  
Example: var removedFromArray = myArray.pop();  
  
Manipulating Arrays with shift()  
shift() removes the first element from an array.  
Example: var removedFromArray = myArray.shift();  
  
Manipulating Arrays with unshift()  
unshift() adds an element to the beginning of an array.  
Example: myArray.unshift(["Liam", 18]);

## 14. Functions

Functions allow you to write reusable blocks of code.  
Example:  
function myReusableCode() {  
 console.log("Heyya, what are you doing");  
}  
myReusableCode();  
  
Function Arguments  
Functions can take inputs (parameters) called arguments.  
Example:  
function myCodeWithArgs(a, b) { console.log(a + b); }  
myCodeWithArgs(10, 5); // 15

## 15. Scope (Global vs Local)

• Global Scope: Variables declared outside of a function are global. If var is used outside a function, it is global.  
• Local Scope: Variables declared inside a function are local to that function only.  
• Undefined Value: A function that doesn’t return anything will return undefined by default.

## 16. Return a Value from a Function with return

function minusSeven(num) {  
 return num - 7;  
}  
  
console.log(minusSeven(10));  
// Output: 3  
  
function timesFive(num) {  
 return num \* 5;  
}  
  
console.log(timesFive(3));  
// Output: 15

## 17. Undefined Value Returned

function addSix() {  
 sum += 6;  
}  
// No return value → Answer would be undefined.

## 18. Assignment with a Returned Value

var changed = 0;  
  
function change(num) {  
 return (num + 5) / 3;  
}  
  
changed = change(10);  
// changed = (10 + 5) / 3 = 5

## 19. Stand in Line (Queue Example)

function nextInLine(arr, item) {  
 arr.push(item);  
 return arr.shift();  
}  
  
let testArr = [1, 2, 3, 4, 5];  
  
console.log("Before: " + JSON.stringify(testArr));  
console.log(nextInLine(testArr, 6));  
console.log("After: " + JSON.stringify(testArr));  
  
// Before: [1, 2, 3, 4, 5]  
// Returned: 1  
// After: [2, 3, 4, 5, 6]

## 20. Boolean Values

function welcomeToBooleans() {  
 return true;  
}

## 21. Use Conditional Logic with if Statements

function trueOrFalse(wasThatTrue) {  
 if (wasThatTrue) {  
 return "Yes, that was true";  
 }  
 return "No, that was not true";  
}  
  
console.log(trueOrFalse(true));  
// Output: "Yes, that was true"

## 22. Comparison with the Equality Operator

function testEqual(val) {  
 if (val == 12) {  
 return "Equal";  
 }  
 return "Not Equal";  
}  
  
console.log(testEqual(12));  
// Output: "Equal"

## 23. Comparison with the Strict Equality Operator

// Strict equal sign is ===  
// Strict equality does not do type conversion.  
  
12 === 12 // true  
12 === "12" // false  
12 == "12" // true

## 24. Comparison with the Inequality Operator

function testNotEqual(val) {  
 if (val != 12) {  
 return "Not Equal";  
 }  
 return "Equal";  
}

## 25. Comparison with Greater Than Operator

function testGreaterThan(val) {  
 if (val > 7) {  
 return "Greater than 7";  
 }  
 if (val > 2) {  
 return "Greater than 2";  
 }  
 return "2 or less";  
}

## 26. Comparison with Greater Than or Equal To Operator

function testGreaterOrEqual(val) {  
 if (val >= 20) {  
 return "20 or more";  
 }  
 if (val >= 10) {  
 return "10 or more";  
 }  
 return "Less than 10";  
}