**JavaScript Notes**

**Overview**

* **JavaScript** uses the **ECMAScript (ES)** standard as its language specification.
* JavaScript is a **case-sensitive** language.

**Syntax for Math Operations**

* Basic operators: +, -, \*, /
* **Order of operations**: Follows **BODMAS** rule:
  + **Brackets** () have the highest priority.
  + **Division** / and **Multiplication** \* have the same precedence and are evaluated left to right.
  + **Addition** + and **Subtraction** - have the same precedence and are evaluated left to right.

**Numbers**

* **Integers**: 2, 3, 4
* **Floating-point numbers**: 0.2, 0.4
* **Math Functions**:
  + Math.round(): Rounds a number to the nearest integer.

**Strings**

* Strings can be enclosed in **single quotes** '' or **double quotes** "".
  + Example: 'hello' or "hello"
* **Concatenation**:
  + Example: 'dsds' + 'sdsd' + 'dsd' = 'dsdssdsddsd'
* **Type Coercion**:
  + Example: 'ksldk' + 2 = 'ksldk2'
* **Checking Type**:
  + typeof 'hello' returns 'string'
  + typeof 22 returns 'number'
* **Escape Characters**:
  + \': Single quote
  + \": Double quote
  + \n: New line
* **Template Strings** (using backticks `):
  + Example: `ITEMS(${1 + 1})` → ITEMS(2)
  + **Interpolation**: Embed expressions inside ${}.
  + Example: `ITEMS(${1 + 1}): $${(2095 + 799) / 100}`
* **Multi-line Strings**:
  + Using backticks:

Copy

`some

text`

* + Using escape characters: 'some \ntext'
* **Best Practice**: Use **single quotes** as the default for strings.

**Variables**

* **Rules for Naming Variables**:
  + Cannot use **reserved keywords** (e.g., let, const, var).
  + Cannot start with a **number**.
  + Can use $ and \_ as special characters.
    - Example: let $name = 3; or let \_name = 3;
* **Reassignment Shortcuts**:
  + +=: variable += 2 → variable = variable + 2
  + -=: variable -= 2 → variable = variable - 2
  + /=: variable /= 2 → variable = variable / 2
  + \*=: variable \*= 2 → variable = variable \* 2
  + ++: variable++ → variable = variable + 1
  + --: variable-- → variable = variable - 1
* **Naming Conventions**:
  + **camelCase**: nikhilVyas (default in JavaScript)
  + **PascalCase**: NikhilVyas
  + **snake\_case**: nikhil\_vyas
  + **kebab-case**: Not supported in JavaScript but used in HTML and CSS.
* **Variable Declaration**:
  + let variable1;: Use when the value may change.
  + const variable2;: Use for constant values (cannot be reassigned).
  + var variable3 = 3;: Legacy way (avoid due to scope issues).
* **typeof Operator**: Used to check the type of a value.
  + Example: typeof 'hello' → 'string'

**Booleans**

* **Values**: true or false
* **Purpose**: Represent conditions or states.
* **Syntax**:

javascript

Copy

let flag = true;

**Comparison Operators**

* >: Greater than
* <: Less than
* >=: Greater than or equal to
* <=: Less than or equal to
* ===: Strict equality (no type coercion)
* !==: Strict inequality (no type coercion)
* **Order of Evaluation**:
  1. ()
  2. \*, /
  3. +, -
  4. Comparison operators (evaluated last).
  5. Example: (3 > 5 - 5) → 5 - 5 is evaluated first.

**Conditional Statements**

* **if and else**:

javascript

Copy

if (condition) {

// code

} else if (anotherCondition) {

// code

} else {

// code

}

* **Logical Operators**:
  + && (AND): Short-circuits if the first condition is false.
  + || (OR): Short-circuits if the first condition is true.
  + ! (NOT): Inverts the boolean value.
* **Truthy and Falsy Values**:
  + **Falsy Values**: false, 0, '', NaN, undefined, null
  + **Truthy Values**: Any value not in the falsy list.
  + Example:

javascript

Copy

if (5) {

console.log('truthy'); // Executes because 5 is truthy.

}

**Shortcuts for Conditional Statements**

* **Ternary Operator**:

javascript

Copy

condition ? expressionIfTrue : expressionIfFalse;

* **Guard Operator** (&&):

javascript

Copy

false && console.log('hello'); // Does not execute.

* **Default Operator** (||):

javascript

Copy

let value = someValue || 'default';

**Functions**

* **Purpose**: Reuse code.
* **Syntax**:

javascript

Copy

function functionName() {

// function body

}

* **Best Practice**: Use **camelCase** for function names.
* **Scope**: Variables inside a function are **local** to that function.
* **Parameters**:
  + Cannot use reserved keywords.
  + Cannot start with a number.
  + Can use $ and \_.
  + Example:

javascript

Copy

function calculate(price) {

console.log(price);

}

calculate(500);

**Objects**

* **Definition**: Groups multiple values together.
* **Syntax**:

javascript

Copy

let object = {

property: value,

method: function() {

// code

}

};

* **Accessing Properties**:
  + Dot notation: object.property
  + Bracket notation: object['property'] (useful for special cases like object['delivery-time'])
* **Nested Objects**: Objects can contain other objects.
* **Methods**: Functions inside objects are called methods.

**JSON (JavaScript Object Notation)**

* **Purpose**: A lightweight data interchange format.
* **Syntax**:
  + All keys and strings must use **double quotes** "".
* **Conversion**:
  + **JavaScript Object to JSON**:

javascript

Copy

const jsonString = JSON.stringify(object);

* + **JSON to JavaScript Object**:

javascript

Copy

const parsedObject = JSON.parse(jsonString);

**null and undefined**

* **null**: Represents an intentional absence of value.
* **undefined**: Represents an uninitialized or missing value.
* **Default Parameters**:

javascript

Copy

function func(parameter = 'default') {

console.log(parameter);

}

func(); // 'default'

func(undefined); // 'default'

func(null); // null

**Local Storage**

* **Usage**:

javascript

Copy

localStorage.setItem('key', 'value');