JS

{ JavaScript }

CRAFT KNOWLEDGE

Objectives

Introduction to JavaScript

- JAVASCRIPT STATEMENTS
- JAVASCRIPT COMMENTS
- JAVASCRIPT VARIABLES
- LET,CONST,VAR KEY WORD
- JAVASCRIPT DATA TYPES
- JAVASCRIPT OPERATORS
    - JAVASCRIPT ARITHMETIC OPERATORS
    - JAVASCRIPT ASSIGNMENT OPERATORS
    - JAVASCRIPT LOGICAL OPERATORS
    - JAVASCRIPT COMPARISON OPERATORS

JAVASCRIPT STATEMENTS

- A computer program is a list of "instructions" to be "executed" by a computer.
- In a programming language, these programming instructions are called statements.
- A JavaScript program is a list of programming statements.
- JavaScript statements are composed of:

  Values, Operators, Expressions, Keywords, and Comments.

- **Semicolons**
  - Semicolons separate JavaScript statements.
  - Add a semicolon at the end of each executable statement:
- **JavaScript White Space**
  - JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.
  - A good practice is to put spaces around operators (= + - * /):

```
let x = y + z;
```

- **JavaScript Code Blocks**
  - JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.
  - The purpose of code blocks is to define statements to be executed together.
  - One place you will find statements grouped together in blocks, is in JavaScript functions:

JavaScript syntax:- is the set of rules, how JavaScript programs are constructed

JAVASCRIPT COMMENTS

- JavaScript comments can be used to explain JavaScript code, and to make it more readable.

- **Single Line Comments**
  - Single line comments start with //.
  - Any text between // and the end of the line will be ignored by JavaScript (will not be executed).

    example:- // This is single line comment

- **Multi-line Comments**
  - Multi-line comments start with /* and end with */.
  - Any text between /* and */ will be ignored by JavaScript.

    Example:- /* This is multi line comment
    we can write multi line comment */

JAVASCRIPT VARIABLES

- Variables are containers for storing data (storing data values).

  Example let name = "Jhon";

  age = 24;
- **Declare a variable**
  - 4 Ways to Declare a JavaScript Variable:
    - Using var
    - Using let
    - Using const
    - Using nothing

- All JavaScript variables must be identified with unique names.
- These unique names are called **identifiers.**
- Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).
- The general rules for constructing names for variables (unique identifiers) are:
  - Names can contain letters, digits, underscores, and dollar signs.
  - Names must begin with a letter.
  - Names can also begin with $ and _
  - Names are case sensitive (y and Y are different variables).
  - Reserved words (like JavaScript keywords) cannot be used as names.
  - By convention, variable names use camelcase like message, yourAge, and myName.

- **The Assignment Operator**
    - In JavaScript, the equal sign (=) is an "assignment" operator, not an "equal to" operator use for assign value to variable

- **Initialize a variable**
- Once you have declared a variable, you can initialize it with a value. To initialize a variable, you specify the variable name, followed by an equals sign (=) and a value.

- **Value = undefined**
    - In computer programs, variables are often declared without a value. The value can be something that has to be calculated, or something that will be provided later, like user input.
    - A variable declared without a value will have the value undefined.

LET,CONST,VAR KEY WORD

- **The let keyword**
  - 1. Variables defined with let can not be redeclared.

    Example:
    ```
    let x = "John Doe";
    let x = 0;
    ```
    But we can by using var keyword

    Example:
    ```
    var x = "John Doe";
    var x = 0;
    ```
  - 2. Variables defined with let have block scope.
  - Variables declared inside a { } block cannot be accessed from outside the block:

    Example
    ```
    {
      let x = 2;
    }
    ```
  - variables declared with the var keyword can NOT have block scope.
  - Variables declared inside a { } block can be accessed from outside the block.

- **The const keyword**
  - 1. Variables defined with const can not be redeclared.

    ```
    Example: const x = "John Doe";
                     const x = 0;     // error
    ```

  - 2. Variables defined with const have block scope.
  - Variables declared inside a { } block cannot be accessed from outside the block:

    ```
    Example
       {
          const x = 2;
       }
       console.log(x)      // error
    ```

  - 3. Variables defined with const cannot be Reassigned.

    ```
    Example
          const x =10;
                    x=20 ;      // error
    ```

  - 4. JavaScript const variables must be assigned a value when they are declared:

    ```
    Example
         const x ;  // error
    ```

JAVASCRIPT DATA TYPES

- JavaScript has the following primitive data types:
    - **number**: JavaScript uses the number type to represent both integer and floating-point numbers.
      Example
      ```
      let num1 = 100;   // integer number
      let mark = 90.8;   // float number
      ```
    - **string**: is a sequence of zero or more characters. A string literal begins and ends with either a single quote(') or a double quote (").
        - A string that begins with a double quote must end with a double quote. Likewise, a string that begins with a single quote must also end with a single quote:
          Example
          ```
          let greeting = 'Hello world';
          let message = "hi every one";
          ```
    - **boolean**: The boolean type has two literal values: true and false in lowercase.
      Example
      ```
      let inProgress = true;
      let completed = false;
      ```

- **undefined**: is a primitive type that has only one value undefined. By default, when a variable is declared but not initialized, it is assigned the value of undefined.

  Example

  ```
  let counter;
  console.log(counter);        // undefined
  ```

- **null**: is a primitive data type that also has only one value null.

  Example:

  ```
  let obj = null;
  ```

  - JavaScript defines that null is equal to undefined as follows:

- **NaN:** stands for Not a Number. It is a special numeric value that indicates an invalid number. For example, the division of a string by a number returns NaN.

  Example: `console.log('a'/2);   // NaN;`

- JavaScript is a **dynamically** typed language. It means that a variable doesn’t associate with a type. In other words, a variable can hold a value of different types.

  Example:

  ```
        let counter = 120; // counter is a number
            counter = false; // counter is now a boolean
             counter = "foo"; // counter is now a string
  ```
- To get the current type of the value that the variable stores, you use the **typeof**() operator:

  Example:

  ```
      typeof(counter ) // number
  ```
- JavaScript allows values of other types to be converted into Other type
  - Convert a number to a string: by using toString(); method

    Exapmle

    ```
        let num = 10
        let num_string = num.toString()
    ```
  - Convert a String to a number: by using Number(), parseFloat() and parseInt()

    Exapmle

    ```
        let num_string = "10"  // string data b/c in  quote (").
        let num = Number(num)
        let num_Float = parseFloat(num)
        let num_Int = parseInt(num)
    ```

JAVASCRIPT OPERATORS

- There are different types of JavaScript operators:
  - **1. Arithmetic Operators:** are used to perform arithmetic on numbers:
    - Addition (+)

      Example

      ```
      let num1 = 10;
      let num2 = 2;
      let num3 = num1 + num2 ;    // 12
      ```
    - Subtraction (-)

      ```
      let num3 = num1 - num2 ; // 8
      ```
    - Multiplication(*)

      ```
      let num3 = num1 * num2 ; // 16
      ```
    - Exponentiation (**)

      ```
      let num3 = num1 ** num2 ; // 100
      ```
    - Division(/)

      ```
      let num3 = num1 / num2 ; // 5
      ```
    - Modulus (Division Remainder) (%)

      ```
      let num3 = num1 % num2 ; // 0   no remainder
      ```

- **2. Assignment Operators:** Assignment operators assign values to variables.
  - x = 10;   // normal assignment
  - x += 5;   // The Addition Assignment Operator (+=) adds a value to a variable.
  - x -=5;   //The Subtraction Assignment Operator (+=) adds a value to a variable.
  - x *= 5;   // The Multiplication Assignment Operator (+=) adds a value to a variable.
  - x **=5;   //The Exponentiation Assignment Operator (+=) adds a value to a variable.
  - x /= 5;   // The Division Assignment Operator (+=) adds a value to a variable.
  - x %=5;   //The Modulus Assignment Operator (+=) adds a value to a variable.

- **3. Comparison Operators:**
  - equal to (==)   compare only value .
  - equal value and equal type (===) compare both value and data types.
  - not equal (!=)   compare only value in negative case.
  - not equal value or not equal type (!==) compare both value and data types in negative case.
  - greater than (>)
  - less than (>=)
  - greater than or equal to(<=)

- **4. Logical Operators**:
  - logical and (&&)

    Example

    ```
    true && true      // true
    true && false     // false
    false && true     // false
    false && false    // false
    ```

  - logical or (||)

    Example

    ```
    true || true // true
    true || false // true
    false  || true // true
    false || false // false
    ```

  - logical not (!)

    Example

    ```
    ! true    //  false
    ! false  //   true
    ```