JS_1

Introduction to JavaScript

Agenda

- Introduction to JS
- How JS runs on the Browser
- How to learn JS with 10X speed
- variable and and Dynamic Typing of JS
- DataTypes in JS (Primitive and Non primitive)
- conditionals, loops
- Functions in JS
- strings

JavaScript is a language that is used to add interactivity to your websites, web apps etc..

History of JavaScript

- Before JavaScript was developed, the information displayed was static which means nothing can be done by the user. **Brendan Eich** developed JavaScript in just 10 days. LiveScript was the name given before JavaScript. Due to the popularity of Java programming language, Brendan gave the name JavaScript instead of LiveScript.
- JavaScript was introduced for browsers. Browsers like Chrome, Mozilla Firefox, and Safari support JavaScript.

fun facts

$\boxed{1995}$

In 10 days, Brendan Eich wrote the initial version of JavaScript for Netscape 2.0 Beta.

$\boxed{\text{ECMAScript}}$

→ Bad parts

→ Copy few of the features from other language

Running JS in browser

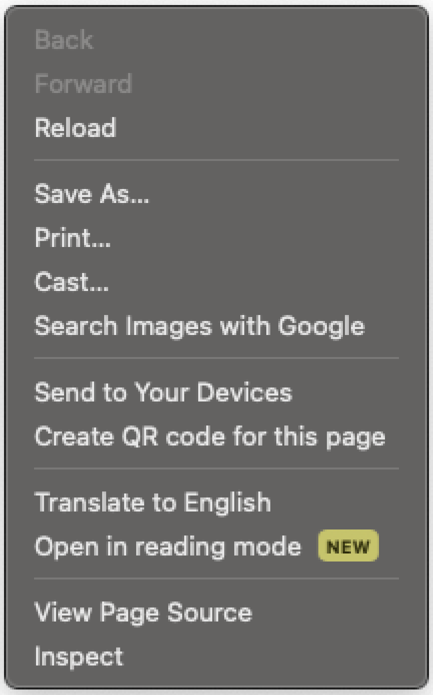
Browser loads HTML file and everything that is added to it is exceuted . If you add script tag that will be executed as well now to run code you need to do the same

```
<body>
    <h1>JS intro</h1>
```

```
      <script>
       // print
        console.log("Hello DevPrep :) ");
     </script>
</body>
```

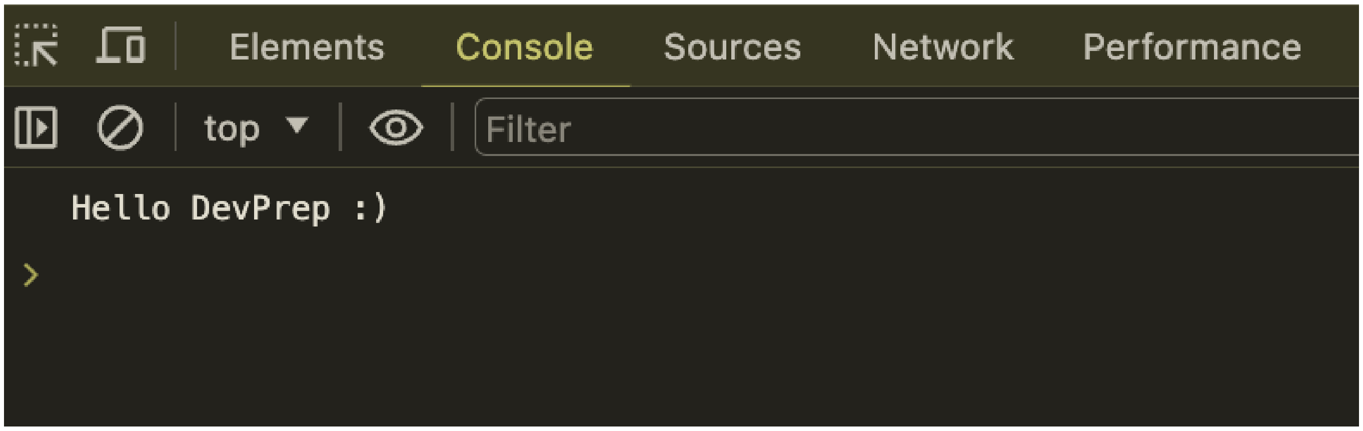
Now to see the output

- open the html file in browser using live server
- right click on the page and select inspect . It will open a dev tool



Back
Forward
Reload
Save As...
Print...
Cast...
Search Images with Google
Send to Your Devices
Create QR code for this page
Translate to English
Open in reading mode NEW
View Page Source
Inspect

- click on console option it will show the output



```
Hello DevPrep :)
>
```

Quick Revision Notes for Beginners

1. `let` Declaration and `undefined`

- **`let` Declaration**:
  - Used to declare variables.
  - Variables declared with `let` are not initialized until their definition is evaluated.
- **`undefined`**:
  - Default value of a declared variable that has not been assigned a value.
  - Indicates the absence of a value.
- Assigning a value to a variable after declaration.
- In js we have only `numbers` -> that behave like maths numbers an there is no concept of integers

```
let varName;
console.log(varName); // Output: undefined

varName = 10;
console.log(varName); // Output: 10

varName = 10.1;
console.log(varName); // Output: 10.1
console.log(5/2) // Output: 2.5
```

2. Strings and Character

- **Strings**:
  - Sequence of characters enclosed in single (`'`) or double (`"`) quotes.
  - Example: `"I am a string"`.

```
let newVar;
newVar = "I am a string";
console.log(newVar); // Output: I am a string
```

3. Boolean Values

- **Boolean**:
  - Represents logical values: `true` or `false`.

```
newVar = true;
console.log(newVar); // Output: true
```

4. Null

- `null`:
  - Represents the intentional absence of any object value.
  - Used to explicitly indicate "no value".

```
newVar = null;
console.log(newVar); // Output: null
```

typeof

What is the Problem?

When working with variables in JavaScript, it can be challenging to determine the type of value a variable holds, especially in a dynamically typed language like JavaScript. Knowing the type of a variable is important for debugging and ensuring the correct operations are performed.

How `typeof` Solves the Problem

The `typeof` operator helps determine the type of a variable, returning a string that indicates the type. This is useful for checking the type of a variable before performing operations on it.

Summary

- **`typeof` Operator**:
  - Used to determine the type of a variable.
  - Returns a string indicating the type.
- **Common Types**:
  - `number`: Represents numeric values.

> - `string`: Represents sequence of characters.
> - `undefined`: Represents unassigned variables.
> - `boolean`: Represents logical values (`true` or `false`).

Code Example

```
// Number
let num = 10;
console.log(typeof num); // Output: "number"

// String
let str = "Hello, World!";
console.log(typeof str); // Output: "string"

// Undefined
let unassigned;
console.log(typeof unassigned); // Output: "undefined"

// Boolean
let isTrue = true;
console.log(typeof isTrue); // Output: "boolean"
```

Quick Revision Notes: `if-else` and `switch` Case

1. `if-else` Statements

The `if-else` statement is used to perform different actions based on different conditions.

Syntax:

```
if (condition) {
    // code to be executed if the condition is true
} else {
    // code to be executed if the condition is false
}
```

Example:

```
let number = 5;
if (number > 0) {
    console.log("Positive");
} else {
    console.log("Non-positive");
}
```

- If `number` is greater than 0, it prints "Positive".
- Otherwise, it prints "Non-positive".

Another Example:

```
let number = 10;
if (number % 2 === 0) {
    console.log("I am even");
} else {
    console.log("I am odd");
}
```

- If `number` is even, it prints "I am even".
- If `number` is odd, it prints "I am odd".

2. `switch` Case

The `switch` statement is used to perform different actions based on different conditions, especially when there are many possible conditions.

Syntax:

```
switch (expression) {
    case value1:
        // code to be executed if expression === value1
        break;
    case value2:
        // code to be executed if expression === value2
        break;
    // more cases ...
    default:
        // code to be executed if none of the cases match
}
```

Example:

```
let fruit = "Apple";

switch (fruit) {
    case "Apple":
        console.log("I am an apple");
        break;
    case "Banana":
        console.log("I am a banana");
        break;
    case "Orange":
        console.log("I am an orange");
        break;
    default:
        console.log("Unknown fruit");
}
```

- If `fruit` is "Apple", it prints "I am an apple".
- If `fruit` is "Banana", it prints "I am a banana".
- If `fruit` is "Orange", it prints "I am an orange".
- If `fruit` does not match any of the cases, it prints "Unknown fruit".

Another Example:

```
let day = "Thursday";

switch (day) {
    case "Monday":
        console.log("Working");
        break;
    case "Tuesday":
    case "Wednesday":
    case "Thursday":
    case "Friday":
        console.log("Today is an off");
        break;
    case "Saturday":
    case "Sunday":
        console.log("Weekend");
        break;
    default:
        console.log("Invalid day");
}
```

- If `day` is "Monday", it prints "Working".
- If `day` is "Tuesday", "Wednesday", "Thursday", or "Friday", it prints "Today is an off".
- If `day` is "Saturday" or "Sunday", it prints "Weekend".
- If `day` does not match any of the cases, it prints "Invalid day".

Summary

- `if-else` **Statements**:
  - Used for basic conditional logic.
  - Executes different code blocks based on a condition.
  - Useful for simple conditions.

Example:

```
let number = 5;
if (number > 0) {
    console.log("Positive");
} else {
    console.log("Non-positive");
}
```

- `switch` **Case**:
  - Used for more complex conditional logic with multiple possible conditions.
  - Executes different code blocks based on the value of an expression.
  - Useful when you have many conditions to check.

Example:

```
let fruit = "Apple";

switch (fruit) {
    case "Apple":
        console.log("I am an apple");
        break;
    case "Banana":
        console.log("I am a banana");
        break;
    case "Orange":
        console.log("I am an orange");
        break;
    default:
        console.log("Unknown fruit");
}
```

- **Key Points**:
  - `if-else` for simple, straightforward conditions.
  - `switch` for handling multiple potential conditions efficiently.

Introduction to Functions

What is a Function?

A function in JavaScript is a block of code designed to perform a particular task. Functions allow you to reuse code, make your program more modular, and improve readability.

Basic Syntax

```
function functionName(parameters) {
    // code to be executed
}
```

Example:

```
function greet() {
    console.log("Hello, World!");
}
```

```javascript
greet(); // Output: Hello, World!
```

Functions with Parameters and Return Values

Functions can take parameters (inputs) and return values (outputs).

Example:

```javascript
function add(a, b) {
    return a + b;
}

let sum = add(5, 3);
console.log(sum); // Output: 8
```

String Handling in JavaScript

Strings in JavaScript can be enclosed in single quotes (`'`) or double quotes (`"`).

Example:

```javascript
let singleQuoteString = 'Hello, World!';
let doubleQuoteString = "Hello, World!";

console.log(singleQuoteString); // Output: Hello, World!
console.log(doubleQuoteString); // Output: Hello, World!
```

Adding Strings (Concatenation)

You can concatenate strings using the `+` operator.

Example:

```javascript
let firstName = "John";
let lastName = "Doe";
let fullName = firstName + " " + lastName;
```

```
console.log(fullName); // Output: John Doe
```