Otaku Tube

JAVA SCRIPT NOTES

Presented by Sewak Dhakal

What is Java Script?

Js is a programming language.
We use it to give instructions to the computer.

Our 1st JS Code

console.log(“Hello World”);

Console.log is used to log (print) a message to the console

VARIABLES IN JS

Variables are containers for data.

Declaration keywords:

var → old, avoid using.
let → block-scoped, can change value.
const → block-scoped, cannot be reassigned.

Example:

```
let name = "Sewak";
const age = 21;
```

Data Types

Primitive types:

- String → "Hello"
- Number → 42
- Boolean → true / false
- Null → null
- Undefined → undefined
- Symbol → Symbol('id')
- BigInt → 12345678901234567890n

Non-primitive:

- Object → { name: "Sewak" }
- Array → [1, 2, 3]
- Function → function greet() {}

Operators

Arithmetic: + - * / % **

Assignment: = += -= *= /=

Comparison: == != === !== > < >= <=

Logical: && || !

Ternary: condition ? value1 : value2

Conditional Statements

Used to implement some condition in the code.

if Statement

```
let color;
if(mode === "dark-mode") {
    color = "black";
}
```

if-else Statement

```
let color;
if(mode === "dark-mode") {
    color = "black";
} else {
   color = "white";
}
```

else-if Statement

```
if(age < 18) {
    console.log("junior");
} else if (age > 60) {
   console.log("senior");
} else {
    console.log("middle");
}
```

LOOPS IN JS

Loops are used to execute a piece of code again & again

for loop:

```
for (let i = 0; i < 5; i++) console.log(i);
```

while loop:

```
let i = 0;
while (i < 5) { console.log(i); i++; }
```

for...of (arrays):

```
for (let item of ["a", "b"]) console.log(item);
```

for...in (objects):

```
for (let key in {name:"Sewak"}) console.log(key);
```

STRING IN JS

A string is a sequence of characters used to represent text.

Creating Strings

```
let str1 = "Hello";      // double quotes
let str2 = 'World';      // single quotes
let str3 = `Hi, Sewak`;  // backticks
```

String Properties

Length: string.length → number of characters (spaces count too).

COMMON & IMPORTANT STRING METHODS

- **Accessing Characters:**
  - **charAt(index)**

```
let word = "Hello";
console.log(word.charAt(1)); // "e"
```

  - **Bracket notation:**

```
console.log(word[1]); // "e"
```

- **Changing Case:**

toUpperCase() → Converts to uppercase.
toLowerCase() → Converts to lowercase.

```
let name = "Sewak";
console.log(name.toUpperCase()); // "SEWAK"
console.log(name.toLowerCase()); // "sewak"
```

Searching Inside Strings

- indexOf(searchValue) → First occurrence position.
- lastIndexOf(searchValue) → Last occurrence position.
- includes(searchValue) → Boolean if found.
- startsWith(searchValue) → Checks start.
- endsWith(searchValue) → Checks end.

```
let text = "JavaScript is awesome";
console.log(text.indexOf("is"));     // 11
console.log(text.includes("awesome")); // true
console.log(text.startsWith("Java"));  // true
console.log(text.endsWith("me"));      // true
```

Extracting Parts of Strings

- slice(start, end) → Extracts portion (end not included).

```
let str = "Frontend Developer";
console.log(str.slice(0, 8)); // "Frontend"
console.log(str.slice(-9));   // "Developer"
```

- substring(start, end) → Similar to slice, but can't use negative indexes.

```
console.log(str.substring(0, 8)); // "Frontend"
```

- substr(start, length) → (Old, avoid in new code)

```
console.log(str.substr(0, 8)); // "Frontend"
```

Removing Extra Spaces

- trim() → Removes spaces from start & end.
- trimStart() / trimEnd() → Removes from start or end only

```
let messy = "   hello world   ";
console.log(messy.trim()); // "hello world"
```

Splitting and Joining

- split(separator) → Turns string into array.

```
let fruits = "apple,banana,orange";
console.log(fruits.split(","));
// ["apple", "banana", "orange"]
```

TEMPLATE LITERALS (BACKTICKS)

Allows embedding variables and expressions.

```
let user = "Sewak";
let greet = `Hello, ${user}! Today is ${new Date().toDateString()}`;
console.log(greet);
```

ARRAY IN JS

Arrays are the collections of items of same datatypes.

- **Create: let arr = [1, 2, 3];**
- **Common methods:**
  - **push(), pop()**
  - **shift(), unshift()**
  - **forEach()**
  - **map(), filter(), reduce()**

```
let nums = [1, 2, 3];
let doubled = nums.map(n => n * 2);
```

- **push()**
  - **What it does: Adds one or more elements to the end of an array.**
  - **Changes the original array? ✅ Yes**
  - **Returns: The new length of the array.**

```
let fruits = ["apple", "banana"];
let length = fruits.push("orange");
console.log(fruits); // ["apple", "banana", "orange"]
console.log(length); // 3
```

- **pop()**
  - **What it does: Removes the last element from an array.**
  - **Changes the original array? ✅ Yes**
  - **Returns: The removed element.**

```
let fruits = ["apple", "banana", "orange"];
let last = fruits.pop();
console.log(fruits); // ["apple", "banana"]
console.log(last);   // "orange"
```

- **shift()**
  - **What it does: Removes the first element from an array.**
  - **Changes the original array? ✅ Yes**
  - **Returns: The removed element.**

```
let fruits = ["apple", "banana", "orange"];
let first = fruits.shift();
console.log(fruits); // ["banana", "orange"]
console.log(first);  // "apple"
```

- **unshift()**
  - **What it does: Adds one or more elements to the start of an array.**
  - **Changes the original array? ✅ Yes**
  - **Returns: The new length of the array.**

```
let fruits = ["banana", "orange"];
let length = fruits.unshift("apple");
console.log(fruits); // ["apple", "banana", "orange"]
console.log(length); // 3
```

- **forEach()**
  - **What it does: Loops through each array element and runs a function.**
  - **Changes the original array? ❌ No (unless you modify it manually inside the loop).**
  - **Returns: Nothing (undefined).**

```
let fruits = ["apple", "banana", "orange"];
fruits.forEach((item, index) => {
  console.log(index, item);
});
// Output:
// 0 apple
// 1 banana
// 2 orange
```

- **map()**
  - **What it does: Creates a new array by applying a function to each element.**
  - **Changes the original array? ❌ No**
  - **Returns: A new array.**

```
let numbers = [1, 2, 3];
let doubled = numbers.map(num => num * 2);
console.log(doubled); // [2, 4, 6]
console.log(numbers); // [1, 2, 3]
```

- **filter()**
  - **What it does: Creates a new array with elements that pass a condition.**
  - **Changes the original array? ❌ No**
  - **Returns: A new filtered array.**

```
let numbers = [1, 2, 3, 4, 5];
let evens = numbers.filter(num => num % 2 === 0);
console.log(evens);   // [2, 4]
console.log(numbers); // [1, 2, 3, 4, 5]
```

- **reduce()**
  - **What it does: Reduces an array to a single value by running a function for each element.**
  - **Changes the original array? ❌ No**
  - **Returns: A single value.**

```
let numbers = [1, 2, 3, 4];
let sum = numbers.reduce((accumulator, current) => accumulator + current, 0);
console.log(sum); // 10
```

FUNCTIONS IN JS

Block of code that performs a specific task, can be invoked whenever needed.

- **Declaration:**

```
function greet(name) {
  return `Hello, ${name}`;
}
```

- **Expression:**

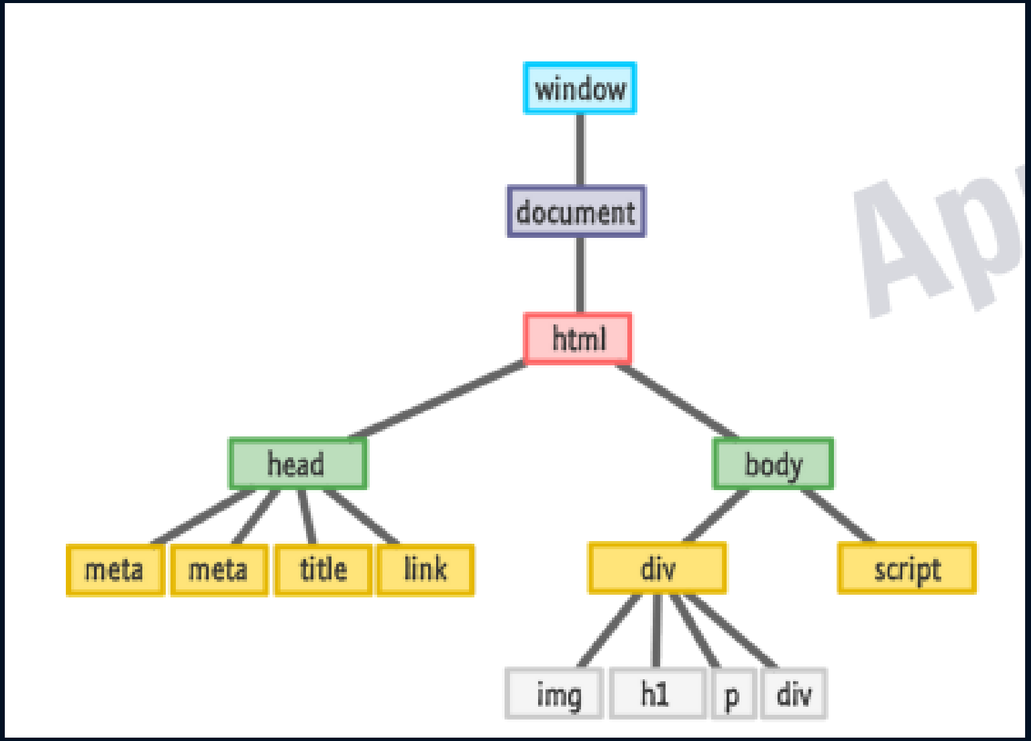
```
const greet = function(name) { return `Hello, ${name}`; };
```

- **Arrow function:**

```
const greet = name => `Hello, ${name}`;
```

DOM (DOCUMENT OBJECT MODEL)

When a web page is loaded, the browser creates a Document Object Model (DOM) of the page



- **Select elements:**

```
document.getElementById("id");
document.querySelector(".class");
document.querySelectorAll("p");
```

- **Change content:**

```
element.textContent = "Hello";
element.innerHTML = "<b>Bold</b>";
```

- **Change style:**

```
element.style.color = "red";
```

- **Create & append elements:**

```
let div = document.createElement("div");
div.textContent = "New Element";
document.body.appendChild(div);
```

EVENTS

The change in the state of an object is known as an Event.

Events are fired to notify code of "interesting changes" that may affect code execution.

```
button.addEventListener("click", function() {
  alert("Button clicked");
});
```

Common events: click, mouseover, keydown, submit

ES6+ FEATURES

- **Template literals: `Hello ${name}`**

- **Destructuring:**

```
let {name, age} = person;
```

- **Spread/rest:**

```
let nums2 = [...nums, 4, 5];
```

- **Default parameters:**

```
function greet(name = "Guest") { ... }
```