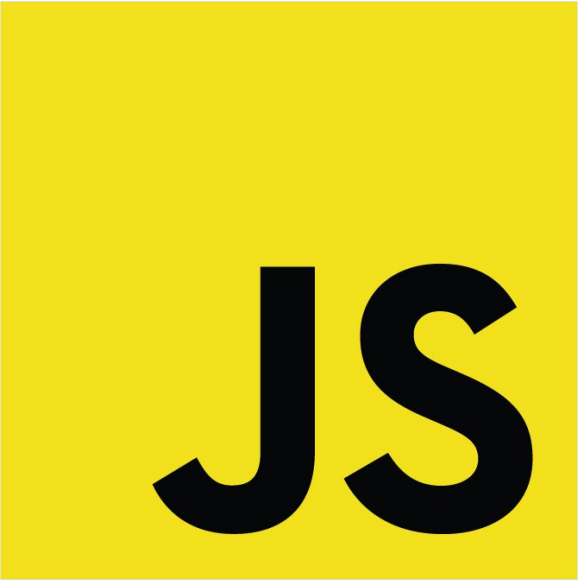
Intro to JavaScript



Who is me?

Muhammad Ikhsan Effendy

- Senior IS student
- Research Assistant at Media-Tech Lab
- Software Engineer at Ikonsultan Inovatama
- Currently working on my final project on big data mining
- Loves cats & dogs

*picture of me eating outside my habitat

Before we go

- Make sure you have code editor (e.g.) VSCode installed on your machine
- You can run your JavaScript code on your machine (without HTML and browsers) using Node.js
- You can also use online compiler such as https://www.programiz.com/javascript/online-compiler/ to test your JavaScript code
- You can also see the output from console.log in browser by Right Click > Inspect > Console

What is JavaScript?

- Dynamic programming language for web development
- Client-side scripting language (runs in browser)
- Adds interactivity and dynamic behavior to websites
- One of the three core web technologies (HTML, CSS, JS)

Why JavaScript?

- Makes websites interactive
- Handles user input and events
- Modifies web page content dynamically
- Creates animations and visual effects
- Communicates with servers
- Powers modern web applications

Variables & Data Types

Basic Data Types:

- String
- Number
- Boolean
- Array
- Object

Variables:

- let - can be reassigned
- const - cannot be reassigned

```javascript
// Variables
let name = "John";          // String
const age = 20;             // Number
let isStudent = true;       // Boolean
let skills = ["HTML", "CSS"]; // Array
let person = {              // Object
    name: "John",
    role: "Developer"
};
```

Source: data_type.js

Operators

- **Arithmetic:** Mathematical operations
- **Comparison:** Compare two values
- **Logical:** Logical operation that returns boolean value
  - AND: return true if both value is true, otherwise false
  - OR: return true if one of two value is true
  - NOT: reverse the value to the opposite value

```
// Arithmetic
let sum = 5 + 3;     // Addition
let diff = 10 - 5;   // Subtraction
let prod = 4 * 2;    // Multiplication
let quot = 15 / 3;   // Division
let mod = 17 % 2;    // Modulus

// Comparison
let isEqual = 5 === 5;     // true
let isGreater = 10 > 5;    // true
let isGreatOrEql = 13 >= 13;    // true
let isLessOrEql = 7 <= 7;       // true

// Logical
let and = true && false;   // false
let or = true || false;    // true
let not = !true;           // false
```

Source: operators.js

Conditional Statements

If Statements:

- Can have multiple conditions (else if)
- Can be nested
- Condition must be boolean or evaluate to boolean

Switch Statement:

- Good for exact value matching
- Default case handles all other values

Ternary Operator:

- Shorthand for simple if/else
- Good for quick value assignment

```javascript
// If Statement
if (age >= 18) {
    console.log("Adult");
} else if (age >= 13) {
    console.log("Teenager");
} else {
    console.log("Child");
}

// Switch Statement
switch (dayOfWeek) {
    case "Monday":
        console.log("Start of week");
        break;
    case "Friday":
        console.log("Weekend soon!");
        break;
    default:
        console.log("Regular day");
}

// Ternary Operator (Short If)
const ageStatus = age >= 18 ? "Adult" : "Minor";
```

Source: conditionals.js

Loops

For Loop:

- Three parts: initialization, condition, increment
- Great for known number of iterations
- Can control increment value

While Loop:

- Runs while condition is true
- Good when number of iterations unknown
- Must update condition to avoid infinite loops

```javascript
// For Loop
for (let i = 0; i < 5; i++) {
    console.log(`Iteration ${i}`);
}

// While Loop
let count = 0;
while (count < 5) {
    console.log(`Count: ${count}`);
    count++;
}
```

Source: loops.js

Loops (cont'd)

For...of Loop:

- Cannot access index directly
- Works with any iterable (arrays, strings)

For...in Loop:

- Designed for objects
- Also works with arrays (but not recommended)

forEach Method:

- Provides value, index, and array
- Cannot break out early

```
// For...of Loop (Arrays)
const fruits = ["apple", "banana", "orange"];
for (const fruit of fruits) {
    console.log(fruit);
}

// For...in Loop (Objects)
const person = {name: "John", age: 30};
for (const key in person) {
    console.log(`${key}: ${person[key]}`);
}

// forEach Method (Arrays)
fruits.forEach((fruit, index) => {
    console.log(`${index}: ${fruit}`);
});
```

Source: loops.js

Functions

Components:

- Name: What we call the function
- Parameters: Input values (optional)
- Body: Code to execute
- Return value: Output (optional)

When to Use Each Type:

- Traditional functions: When you need this binding or constructors
- Arrow functions: For shorter syntax and lexical this
- One-liners: For simple operations

```javascript
// Function Declaration
function greet(name) {
    return `Hello, ${name}!`;
}

// Arrow Function (Modern Syntax)
const greet = (name) => {
    return `Hello, ${name}!`;
};

// One-liner arrow function
const greet = (name) => `Hello, ${name}!`;

// Function Usage
greet("John"); // Returns: "Hello, John!"
```

Source: functions.js

Document Object Model (DOM)

Tree-like structure of HTML elements

Allows JavaScript to:

- Access HTML elements
- Modify content
- Change styles
- React to events

Selection Tips:

- IDs must be unique
- Classes can be reused
- querySelector is most flexible
- Always check if element exists before using

```
// DOM Selection
// By ID (returns single element)
document.getElementById('myId');

// By Class (returns collection)
document.getElementsByClassName('myClass');

// By CSS selector (returns first match)
document.querySelector('.class');

// By CSS selector (returns all matches)
document.querySelectorAll('.class');
```

Source: dom.js

DOM Manipulation

```
// Changing Content
element.textContent = "New text";
element.innerHTML = "<span>HTML content</span>";

// Modifying Styles
element.style.backgroundColor = "blue";
element.style.display = "none";

// Adding/Removing Classes
element.classList.add("highlight");
element.classList.remove("hidden");
element.classList.toggle("active");
```

Source: dom.js

Best Practice:

- Prefer classList over direct styles
- Use textContent for plain text
- Be cautious with innerHTML (security)

Events Handling

Common Events:

- click: Mouse clicks
- mouseover/mouseout: Hover states
- keydown/keyup: Keyboard input
- submit: Form submission
- load: Page/resource loading
- resize: Window resizing
- scroll: Page scrolling

```
// Event Listeners
button.addEventListener('click', function() {
    console.log("Button clicked!");
});

// Multiple Events
const handleHover = () => {
    console.log("Mouse over!");
};

element.addEventListener('mouseover', handleHover);
element.addEventListener('mouseout', () => {
    console.log("Mouse out!");
});
```

Source: events.js

#That’s all for the material!

Do you have any questions?

Further questions? Email me anytime at muhammad.effendy@my.sampoernauniversity.ac.id or through Microsoft Teams