

Nama : Naufal Aflakh Wijayanto

NIM : 2211104073

Kelas : SE063

Index.html:

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>TP Modul 12</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Program Tanda Bilangan</h1>
    <input type="number" id="inputNumber" placeholder="Masukkan angka">
    <button onclick="showResult()">Cek Tanda Bilangan</button>
    <label id="resultLabel">Hasil akan ditampilkan di sini</label>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

Kode HTML di atas merupakan struktur dasar dari sebuah halaman web sederhana berjudul "**Program Tanda Bilangan**". Halaman ini memungkinkan pengguna untuk memasukkan sebuah angka melalui elemen input bertipe number, lalu menekan tombol untuk mengecek tanda bilangan tersebut (positif, negatif, atau nol). Hasil dari pengecekan akan ditampilkan pada elemen label dengan ID resultLabel. Desain halaman dapat disesuaikan melalui file CSS eksternal styles.css, dan logika fungsionalitas tombol dikendalikan oleh file JavaScript eksternal script.js yang berisi fungsi showResult(). Halaman ini juga sudah responsif karena menggunakan meta tag viewport.

Script.js

```
// Fungsi untuk menentukan tanda bilangan
function CariTandaBilangan(a) {
  if (a < 0) {
    return "Negatif";
  } else if (a > 0) {
    return "Positif";
  } else {
    return "Nol";
  }
}

module.exports = { CariTandaBilangan };

// Fungsi untuk menampilkan hasil berdasarkan input
function showResult() {
  let input = document.getElementById("inputNumber").value;
  let resultLabel = document.getElementById("resultLabel");

  if (input === "") {
    resultLabel.textContent = "Masukkan angka terlebih dahulu!";
  } else {
    let result = CariTandaBilangan(parseInt(input));
    resultLabel.textContent = "Hasil: " + result;
  }
}
```

Kode JavaScript di atas berisi dua fungsi utama. Fungsi pertama, `CariTandaBilangan(a)`, digunakan untuk menentukan tanda dari suatu bilangan yang diberikan sebagai parameter: jika kurang dari nol akan mengembalikan string "Negatif", jika lebih dari nol mengembalikan "Positif", dan jika sama dengan nol mengembalikan "Nol". Fungsi ini juga diekspor menggunakan `module.exports` agar dapat digunakan dalam pengujian atau file lain (terutama dalam lingkungan Node.js). Fungsi kedua, `showResult()`, digunakan untuk menangani interaksi di halaman web. Fungsi ini mengambil nilai dari input pengguna, memvalidasi apakah input kosong, lalu menampilkan hasil pengecekan tanda bilangan ke elemen dengan ID `resultLabel`. Jika input kosong, pengguna diberi pesan peringatan agar mengisi angka terlebih dahulu.

#### CariTandaBilangan.test.js

```
const { CariTandaBilangan } = require('./script.js');

test('returns "Negatif" for negative numbers', () => {
  expect(CariTandaBilangan(-5)).toBe("Negatif");
});

test('returns "Positif" for positive numbers', () => {
  expect(CariTandaBilangan(5)).toBe("Positif");
});

test('returns "Nol" for zero', () => {
  expect(CariTandaBilangan(0)).toBe("Nol");
});
```

Kode di atas merupakan skrip unit testing menggunakan framework **Jest** untuk menguji fungsi CariTandaBilangan yang diimpor dari file script.js. Terdapat tiga pengujian utama: pertama, memastikan bahwa fungsi mengembalikan "Negatif" jika input berupa bilangan negatif; kedua, memastikan fungsi mengembalikan "Positif" jika input adalah bilangan positif; dan ketiga, memastikan fungsi mengembalikan "Nol" jika input adalah angka nol. Dengan pengujian ini, dapat dipastikan bahwa fungsi CariTandaBilangan berperilaku sesuai dengan logika yang diharapkan dalam berbagai kondisi input.

#### Styles.css

```
body {
  font-family: Arial, sans-serif;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
  background-color: #f4f4f9;
}
```

```
.container {  
  text-align: center;  
  padding: 20px;  
  background-color: white;  
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
  border-radius: 8px;  
}
```

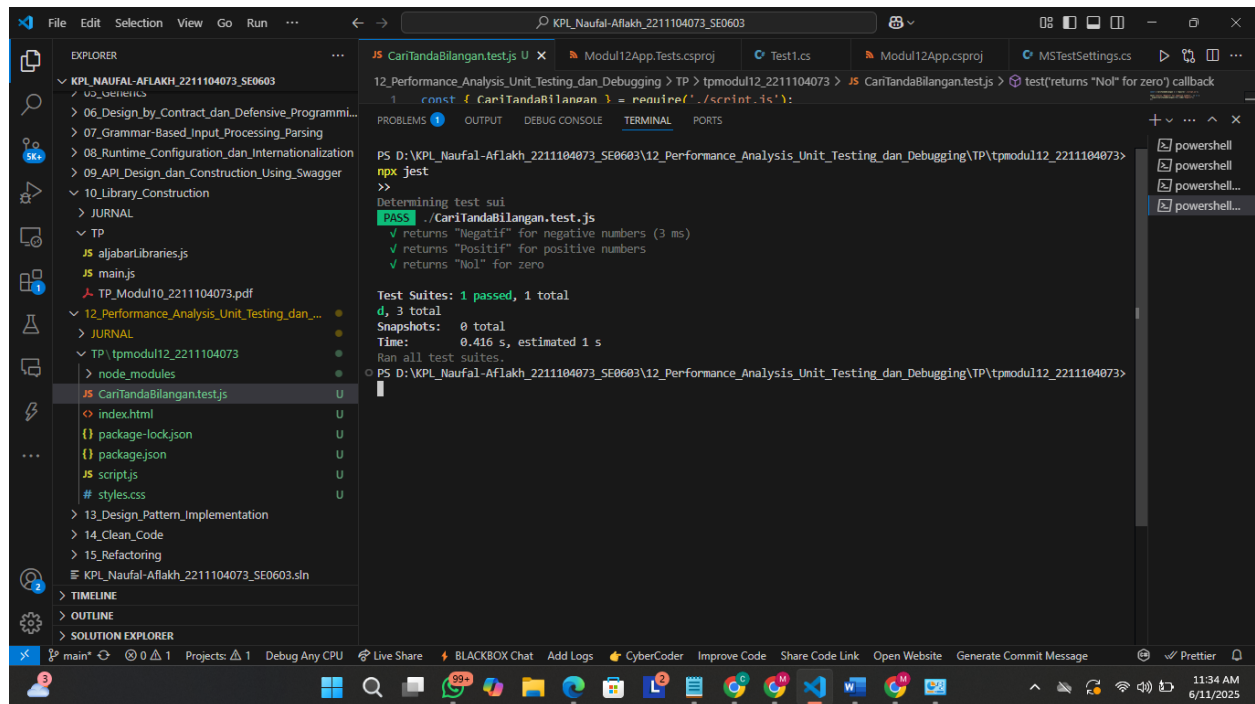
```
input {  
  padding: 8px;  
  margin: 10px 0;  
  width: 200px;  
  border: 1px solid #ccc;  
  border-radius: 4px;  
}
```

```
button {  
  padding: 10px 20px;  
  border: none;  
  background-color: #007bff;  
  color: white;  
  border-radius: 4px;  
  cursor: pointer;  
}
```

```
button:hover {  
  background-color: #0056b3;  
}
```

```
label {  
  display: block;  
  margin-top: 20px;  
  font-size: 1.2em;  
  font-weight: bold;  
}
```

## MENAMBAHKAN UNIT TESTING



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer:** The file tree on the left shows the project structure. The file `CariTandaBilangan.test.js` is selected under the `TP` folder.
- Code Editor:** The main editor shows the content of `CariTandaBilangan.test.js`. The code includes a `const CariTandaBilangan = require('../script.js');` and a `test('returns "No!" for zero') callback` function.
- Terminal:** The terminal at the bottom displays the output of the `npx jest` command. It shows that the test suite passed, with 1 passed test and 1 total test. The execution time was 0.416 seconds.

```
PS D:\KPL_Naufal-Aflakh_2211104073_SE0603\12_Performance_Analysis_Unit_Testing_dan_Debugging\TP\tpmodul12_2211104073> npx jest
Determining test suites to run...
PASS ./CariTandaBilangan.test.js
  ✓ returns "Negatif" for negative numbers (3 ms)
  ✓ returns "Positif" for positive numbers
  ✓ returns "No!" for zero
Test Suites: 1 passed, 1 total
d, 3 total
Snapshots: 0 total
Time: 0.416 s, estimated 1 s
Ran all test suites.
```

Output di atas menunjukkan bahwa perintah `npx jest` berhasil menjalankan seluruh *unit test* yang terdapat pada file `CariTandaBilangan.test.js`. Hasilnya, seluruh pengujian lulus (PASS), yang berarti fungsi `CariTandaBilangan` bekerja sesuai dengan yang diharapkan untuk semua skenario pengujian: bilangan negatif, positif, dan nol. Total terdapat 1 berkas test suite dengan 3 test case yang semuanya sukses dijalankan tanpa error, dengan waktu eksekusi sekitar 0,4 detik. Ini menunjukkan bahwa implementasi fungsi telah terverifikasi dengan baik melalui pengujian otomatis.