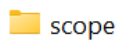


# Exercise Lecture 9

Rembet, Freshy Lestari

## I. Block Scope *with* let & const

- Pertama-tama buat satu folder disini saya menamakan foldernya **scope** (opsional) yang akan di isi dengan 5 file di dalamnya.



15/09/2024 12:08

File folder

- Kedua saya membuat file pertama dengan nama **demo1**



15/09/2024 12:21

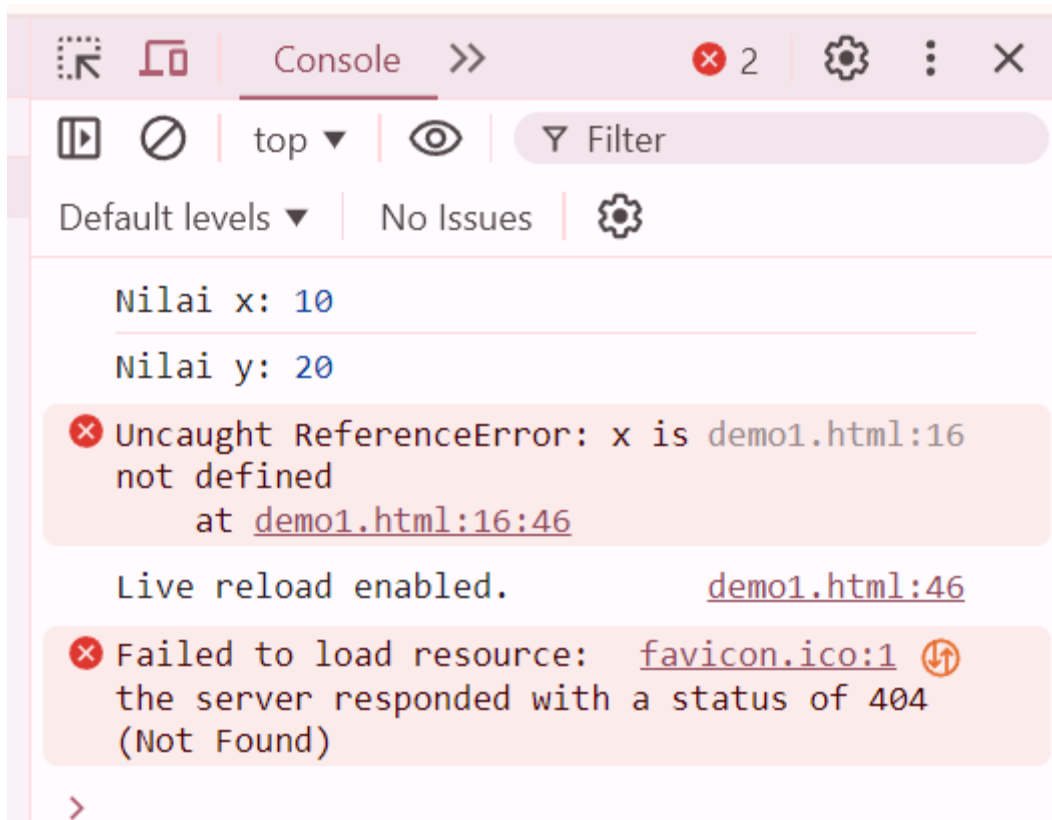
Chrome HTML Docu...

1 KB

Dalam demo1 di isi dengan code html yang telah sir berikan di gcr seperti berikut:

```
<> demo1.html > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>Block Scope with let & const</title>
5  </head>
6  <body>
7  |   <script>
8  |       if (true) {
9  |           let x = 10; // x hanya dapat di akses dalam blok ini
10 |           const y = 20; // y juga hanya dapat di akses dalam blok ini
11 |           console.log("Nilai x:", x); //Output: 10
12 |           console.log("Nilai y:", y); //Output: 20
13 |       }
14 |
15 |       //x dan y tidak dapat di akses di luar blok:
16 |       console.log("Nilai x di luar blok:", x); //error
17 |   </script>
18 </body>
19 </html>
```

Kode yang di atas akan menghasilkan output:



- Kesimpulan: Di dalam blok if, x dan y ditampilkan dengan nilai 10 dan 20. Setelah blok if, akan muncul error karena mencoba mengakses variabel x yang tidak ada di luar cakupan blok tersebut.

## II. Function Scope *using* var

- Masih dalam folder yang sama, selanjutnya kita buat file yang kedua dengan nama **demo2**

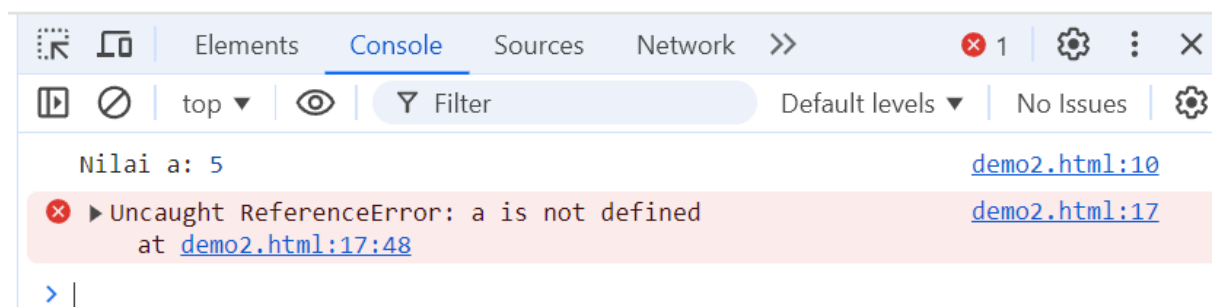
Dalam demo2 ini di isi dengan code html seperti berikut:

```

demo2.html > html
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Function Scope using var</title>
5    </head>
6    <body>
7      <script>
8        function exampleFunction() {
9          var a = 5; // a hanya dapat diakses dalam fungsi
10         console.log("Nilai a:", a); //output: 5
11        }
12
13        // call function
14        exampleFunction();
15
16        // a tidak dapat diakses di luar fungsi
17        console.log("Nilai a di luar fungsi:", a); //error
18      </script>
19    </body>
20  </html>

```

Code ini akan menghasilkan output:



- Kesimpulan: Di dalam fungsi exampleFunction, variabel a bernilai 5 dan ditampilkan dengan console.log. Di luar fungsi, variabel a tidak bisa diakses dan akan menyebabkan error ReferenceError.

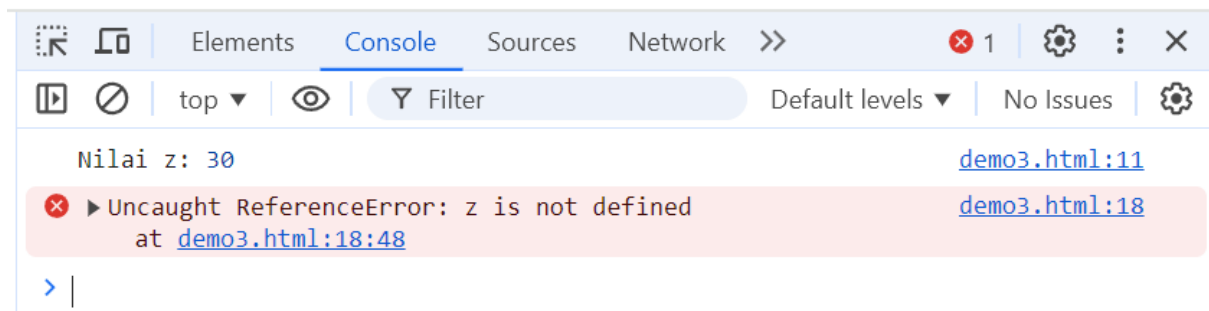
### III. Arrow Function Scope

- Folder yang ketiga kita beri nama **demo3**

Dalam demo3 kita isi dengan code html seperti berikut:

```
demo3.html > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>Arrow Function Scope</title>
5  </head>
6  <body>
7  |   <script>
8  |       // arrow function
9  |       const arrowFunction = () => {
10 |           let z = 30; //z hanya dapat di akses dalam fungsi
11 |           console.log("Nilai z:", z); //Output: 30
12 |       };
13 |
14 |       //call function
15 |       arrowFunction();
16 |
17 |       //z tidak dapat di akses di luar fungsi
18 |       console.log("Nilai z di luar fungsi:", z); //error
19 |   </script>
20 </body>
21 </html>
```


Output yang akan di hasilkan adalah:



- Kesimpulan: Saat memanggil fungsi arrowFunction, variabel z bernilai 30 dan ditampilkan dengan console.log. Ketika mencoba mengakses z di luar fungsi, akan terjadi error ReferenceError karena z tidak didefinisikan di luar fungsi tersebut.

#### IV. Global Scope

- Berikutnya buat file dengan nama **demo4**

 demo4

15/09/2024 12:45

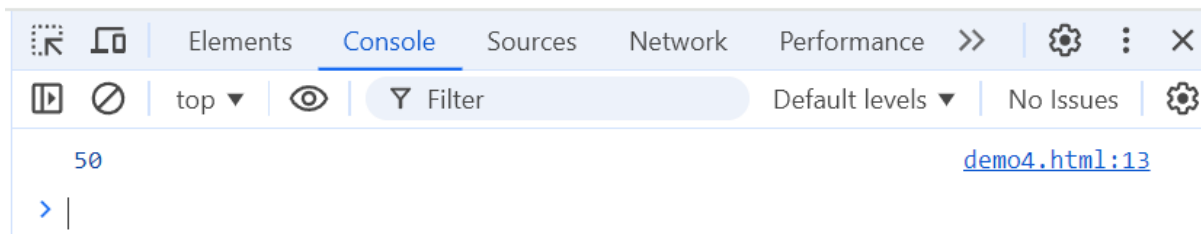
Chrome HTML Docu...

1 KB

Disini kita isi juga dengan code seperti yang sir berikan:

```
<> demo4.html > html > body
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Global Scope</title>
5    </head>
6  <body>
7    <script>
8      // variable ini dapat diakses dari mana saja
9      const globalVar = 50;
10
11     // function
12     function GlobalFunction() {
13       console.log(globalVar);
14     }
15
16     // call function
17     GlobalFunction(); //output: 50
18   </script>
19 </body>
20 </html>
```

Output:



The screenshot shows the Chrome DevTools console with the 'Console' tab selected. The output is the number '50', which is the value of the global variable 'globalVar'. The source is listed as 'demo4.html:13', corresponding to the line where 'GlobalFunction()' is called in the code block above.

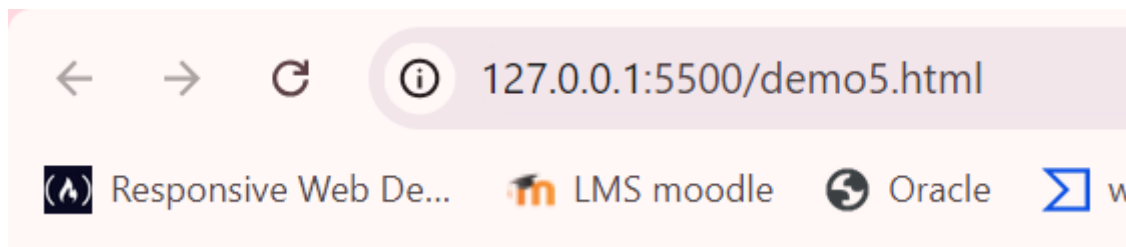
- Kesimpulan: Variabel `globalVar` memiliki **global scope**, artinya bisa diakses di mana saja dalam skrip, baik di dalam fungsi maupun di luar. Fungsi `GlobalFunction` bisa mengakses `globalVar` tanpa masalah, dan ketika dipanggil, akan mencetak nilai 50 ke console.

## V. Template Literal ES6

- Terahir kita buat file dengan nama **demo5** dan di isi dengan code html seperti berikut:

```
<?demo5.html> <?html>
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title> Template Literal ES6</title>
5    </head>
6    <body>
7      <div id="output"></div>
8
9      <script>
10         //Menggunakan Template Literal
11         const nama = 'Freshy Rembet';
12         const usia = 19;
13         const pekerjaan = 'Mahasiswa';
14
15         const profil = `
16             Name: ${nama} <br>
17             Age: ${usia} tahun <br>
18             Expertise/job: ${pekerjaan}
19         `;
20
21         // Menampilkan output ke dalam elemen HTML dengan ID "output"
22         document.getElementById('output').innerHTML = profil;
23     </script>
24 </body>
25 </html>
```

Outputnya:



Name: Freshy Rembet  
Age: 19 tahun  
Expertise/job: Mahasiswa

## VI. CALCULATOR




- Disini kita buat folder baru dengan nama:

calculator

11/09/2024 10:36

File folder

- Dalam folder ini kita buat 3 file di dalamnya:

Name	Date modified	Type
 app	11/09/2024 11:23	JavaScript Source File
 index	11/09/2024 11:23	Chrome HTML Docu...
 style	11/09/2024 11:23	CSS Source File


Di setiap file kita isi dengan code yang telah sir berikan di ger

- HTML, di bagian ini kita hanya membuat kerangka dari kalkulator.

```
index.html x # style.css JS app.js
index.html > html > body > div.calculator
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Calculator Demo</title>
7   <link rel="stylesheet" href="style.css">
8 </head>
9 <body>
10   <div class="calculator">
11     <input type="text" class="calculator-screen" value="" disabled />
12     <div class="calculator-keys">
13       <button type="button" class="operator" value="+">+</button>
14       <button type="button" class="operator" value="-">-</button>
15       <button type="button" class="operator" value="*">*</button>
16       <button type="button" class="operator" value="/">/</button>
17
18       <button type="button" value="7">7</button>
19       <button type="button" value="8">8</button>
20       <button type="button" value="9">9</button>
21       <button type="button" value="4">4</button>
22       <button type="button" value="5">5</button>
23       <button type="button" value="6">6</button>
24       <button type="button" value="1">1</button>
25       <button type="button" value="2">2</button>
26       <button type="button" value="3">3</button>
27       <button type="button" value="0">0</button>
28       <button type="button" class="decimal" value=".">.</button>
29       <button type="button" class="all-clear" value="all-clear">AC</button>
30
31       <button type="button" class="equal-sign" value="=">=</button>
32     </div>
33   </div>
34   <script src="app.js"></script>
35 </body>
36 </html>
```

- CSS, di bagian css ini kita mulai mengrangkai/membuat tampilan di kalkulator menjadi menarik dengan cara membagi dan mengedit setiap bagian-bagian class yang telah di buat di html tadi.

<> index.html × # style.css × JS app.js

# style.css >  .calculator-screen

```
1  * {
2    box-sizing: border-box;
3  }
4
5  body{
6    font-family: 'Roboto', sans-serif;
7    display: flex;
8    justify-content: center;
9    align-items: center;
10   height: 100vh;
11   margin: 0;
12   background-color: #f4f4f4;
13 }
14
15 .calculator{
16   border: 1px solid #999;
17   padding: 20px;
18   border-radius: 10px;
19   background-color: #fff;
20 }
21
22 .calculator-screen{
23   width: 100%;
24   height: 40px;
25   text-align: right;
26   margin-bottom: 10px;
27   padding-right: 10px;
28   font-size: 1.5em;
29 }
30
31 .calculator-keys{
32   display: grid;
33   grid-template-columns: repeat(4, 1fr);
34   gap: 10px;
35 }
```

```
36
37 button{
38   padding: 20px;
39   font-size: 1.2em;
40   cursor: pointer;
41 }
42
43 .operator{
44   background-color: #f0ad4e;
45   color: white;
46 }
47
48 .equal-sign{
49   background-color: #5cb85c;
50   color: white;
51   grid-column: span 4;
52 }
```



- JAVA, di sini adalah titik dimana kalkulator yang kita buat agar biasa berjalan/ berfungsi dengan baik.

```
index.html # style.css JS app.js ×
JS app.js > forEach() callback > button.addEventListener('click') callback
1 // create class
2 class Calculator {
3   // init process
4   constructor() {
5     this.screenValue = '';
6     this.firstOperand = null;
7     this.secondOperand = null;
8     this.operator = null;
9   }
10
11   appendNumber = (number) => {
12     this.screenValue += number;
13     this.updateScreen();
14   };
15
16   chooseOperator = (operator) => {
17     if (this.screenValue === '') return;
18     this.firstOperand = parseFloat(this.screenValue);
19     this.operator = operator;
20     this.screenValue = '';
21   };
22
23   calculate = () => {
24     if (this.operator === null || this.screenValue === '') return;
25     this.secondOperand = parseFloat(this.screenValue);
26     const result = this.operate();
27     this.screenValue = result;
28     this.updateScreen();
29     this.firstOperand = null;
30     this.secondOperand = null;
31     this.operator = null;
32   };
33
34   operate = () => {
35     switch (this.operator) {
36       case '+':
```

```

33
34     operate = () => {
35         switch (this.operator) {
36             case '+':
37                 return this.firstOperand + this.secondOperand;
38             case '-':
39                 return this.firstOperand - this.secondOperand;
40             case '*':
41                 return this.firstOperand * this.secondOperand;
42             case '/':
43                 return this.firstOperand / this.secondOperand;
44             default:
45                 return '';
46         }
47     };
48
49     clear = () => {
50         this.screenValue = '';
51         this.firstOperand = null;
52         this.secondOperand = null;
53         this.operator = null;
54         this.updateScreen();
55     };
56
57     updateScreen = () => {
58         const screen = document.querySelector('.calculator-screen');
59         screen.value = this.screenValue;
60     };
61 }
62
63 // create object
64 const calculator = new Calculator();
65
66 // add to html elements
67 document.querySelectorAll('button').forEach((button) => {

```

```

63 // create object
64 const calculator = new Calculator();
65
66 // add to html elements
67 document.querySelectorAll('button').forEach((button) => {
68     button.addEventListener('click', () => {
69         const value = button.value;
70         if (!isNaN(value) || value === '.') {
71             calculator.appendNumber(value);
72         } else if (value === 'all-clear') {
73             calculator.clear();
74         } else if (value === '=') {
75             calculator.calculate();
76         } else {
77             calculator.chooseOperator(value);
78         }
79     });
80 });

```

Output dari keseluruhan code html, css, dan java:

