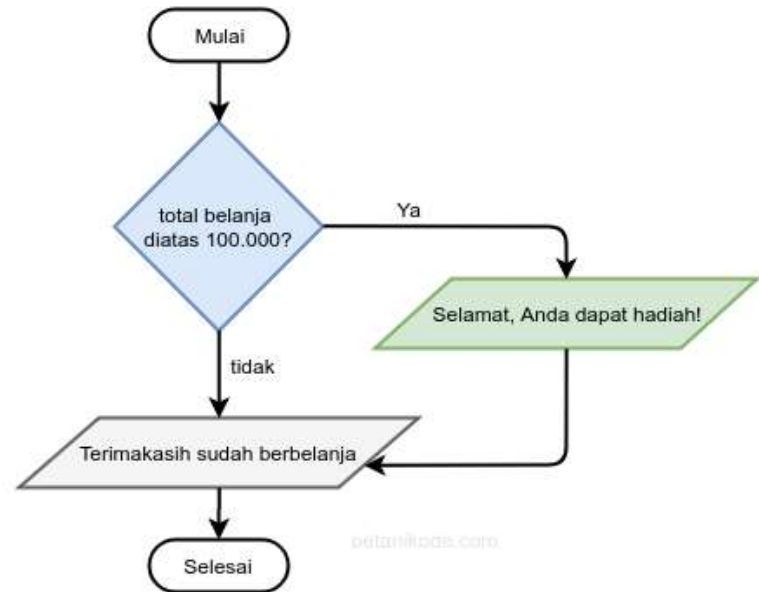


# 1. Percabangan *if*

Percabangan *if* merupakan percabangan yang hanya memiliki satu blok pilihan saat kondisi bernilai benar.

Coba perhatikan *flowchart* berikut ini:



*Flowchart* tersebut dapat dibaca seperti ini:

“Jika total belanja lebih besar dari Rp 100.000, Maka tampilkan pesan Selamat, Anda dapat hadiah“

Bila dibawah Rp 100.000 pesannya tidak ditampilkan.

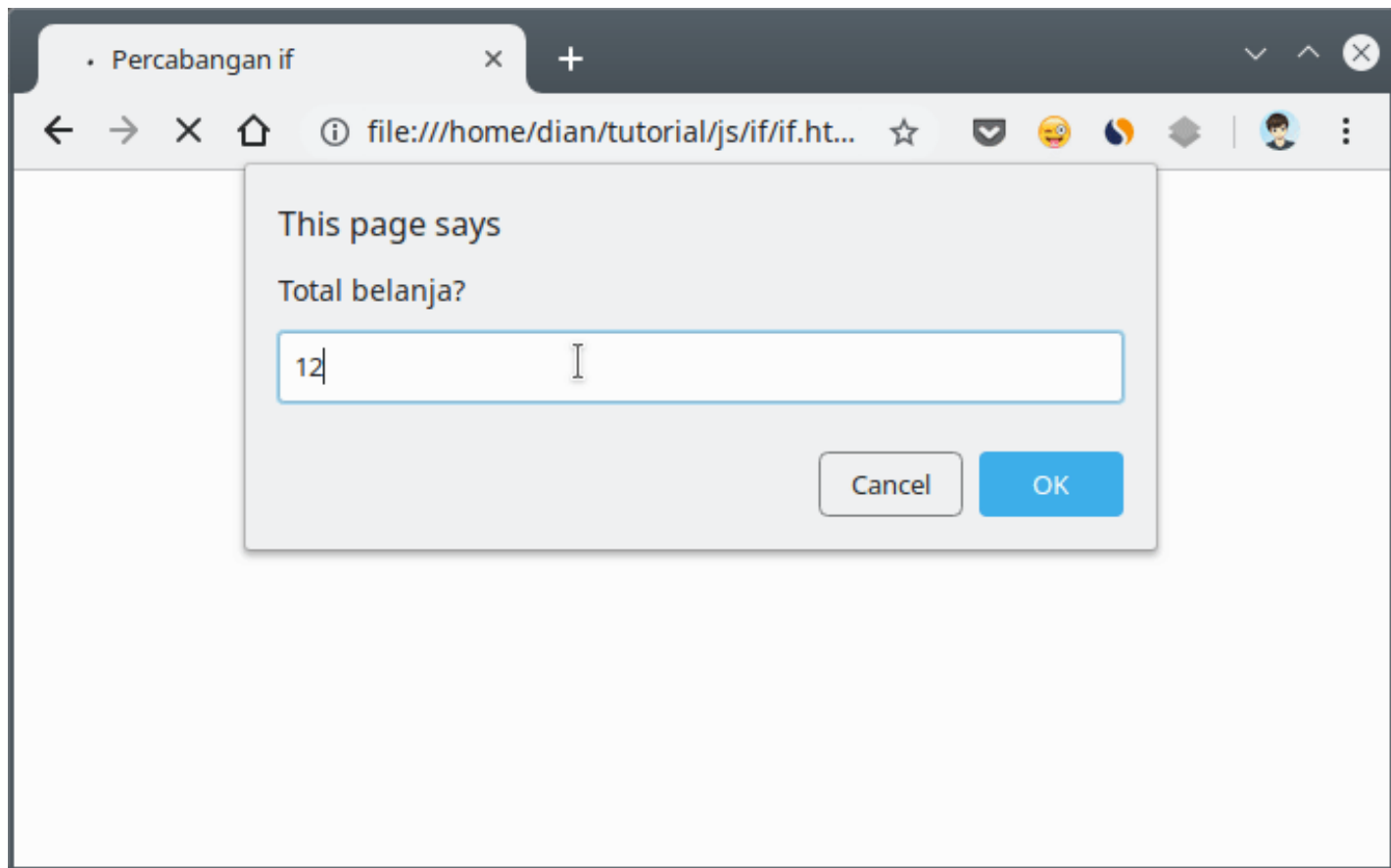
# Contoh:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Percabangan if</title>
</head>
<body>
  <script>
    var totalBelanja = prompt("Total belanja?", 0);

    if(totalBelanja > 100000){
      document.write("<h2>Selamat Anda dapat hadiah</h2>");
    }

    document.write("<p>Terimakasih sudah berbelanja di toko kami</p>")
  </script>
</body>
</html>
```

# Hasil:



# Percabangan If

```
if(totalBelanja > 100000){  
    document.write("<h2>Selamat Anda dapat hadiah</h2>");  
}
```

Bagian ini disebut dengan blok. Blok program pada Javascript, diawali dengan tanda buka kurung kurawal { dan diakhiri dengan tutup kurung kurawal }. Apabila di dalam blok hanya terdapat satu baris ekspresi atau statement, maka boleh tidak ditulis tanda kurungnya.

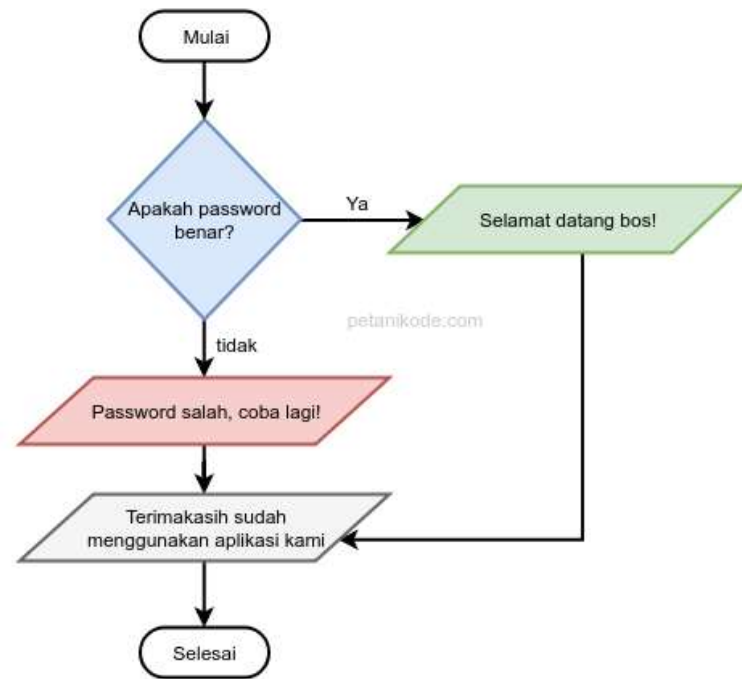
```
if(totalBelanja > 100000)  
    document.write("<h2>Selamat Anda dapat hadiah</h2>");
```

## 2. Percabangan *if/else*

Percabangan *if/else* merupakan percabangan yang memiliki dua blok pilihan.

Pilihan pertama untuk kondisi benar, dan pilihan kedua untuk kondisi salah (*else*).

Coba perhatikan flowchart untuk mengecek password ini:



Apabila password benar, pesan yang ada pada blok hijau akan ditampilkan: "Selamat datang bos!"

Tapi kalau salah, maka pesan yang ada di blok merah yang akan ditampilkan: "Password salah, coba lagi!".

Kemudian, pesan yang berada di blok abu-abu akan tetap ditampilkan, karena dia bukan bagian dari blok percabangan *if/else*.

# Contoh:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Percabangan if/else</title>
</head>
<body>
  <script>
    var password = prompt("Password:");

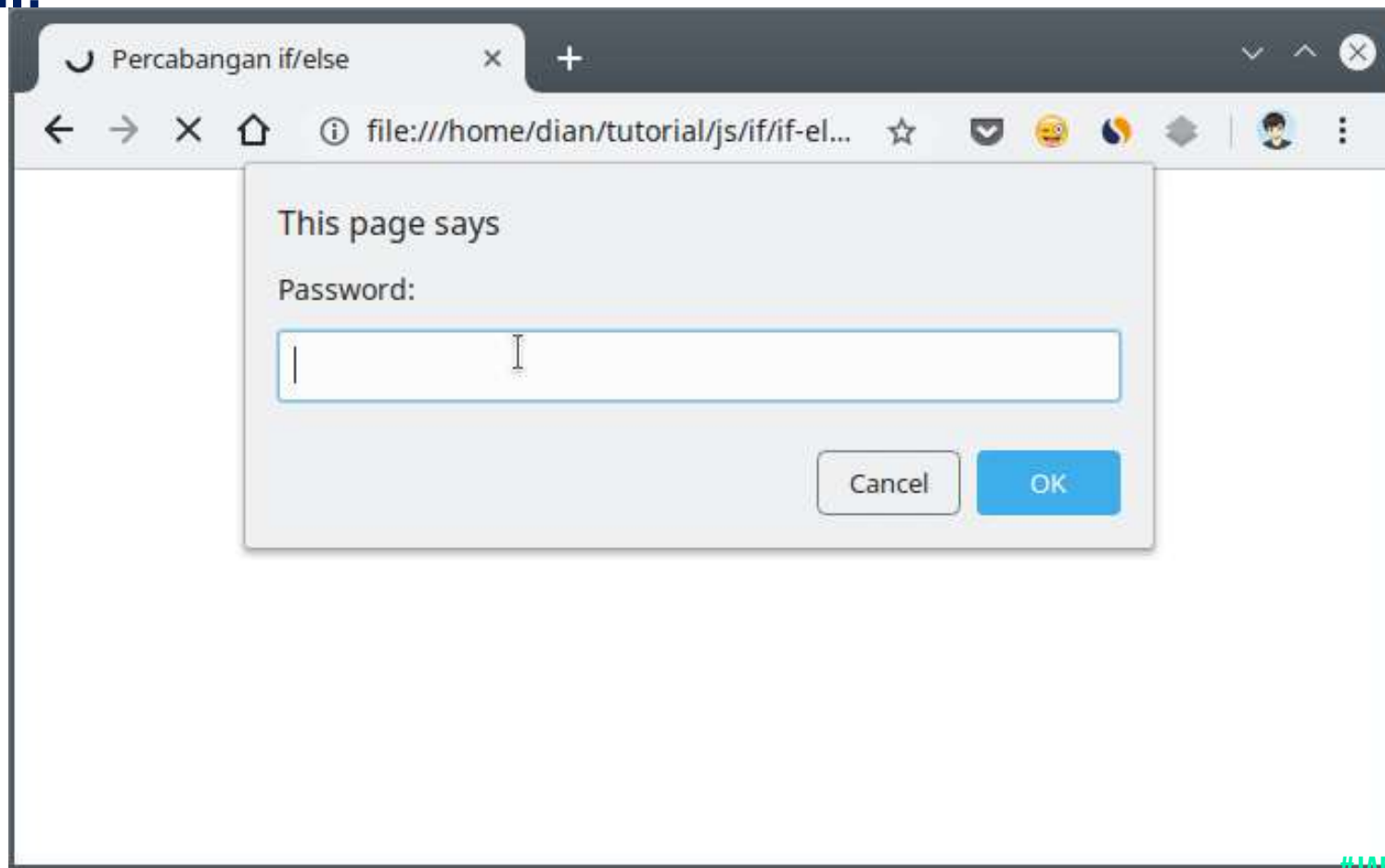
    if(password == "kopi"){
      document.write("<h2>Selamat datang bos!</h2>");
    } else {
      document.write("<p>Password salah, coba lagi!</p>");
    }

    document.write("<p>Terima kasih sudah menggunakan aplikasi ini!</p>");

  </script>
</body>
</html>
```



Hasil:

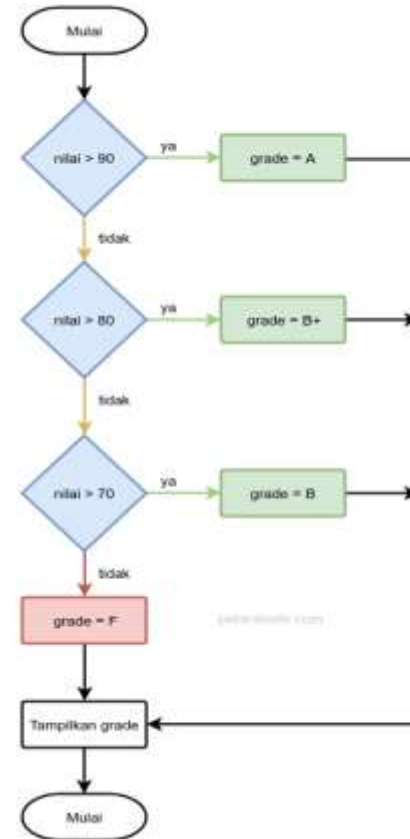


### 3. Percabangan *if/else/if*

Percabangan if/else/if merupakan percabangan yang memiliki lebih dari dua blok pilihan.

Coba perhatikan flowchart berikut:

Perhatikan blok yang diberi warna, Ini adalah blok untuk percabangan if/else/if. Kita bisa menambahkan berapapun blok yang kita inginkan.



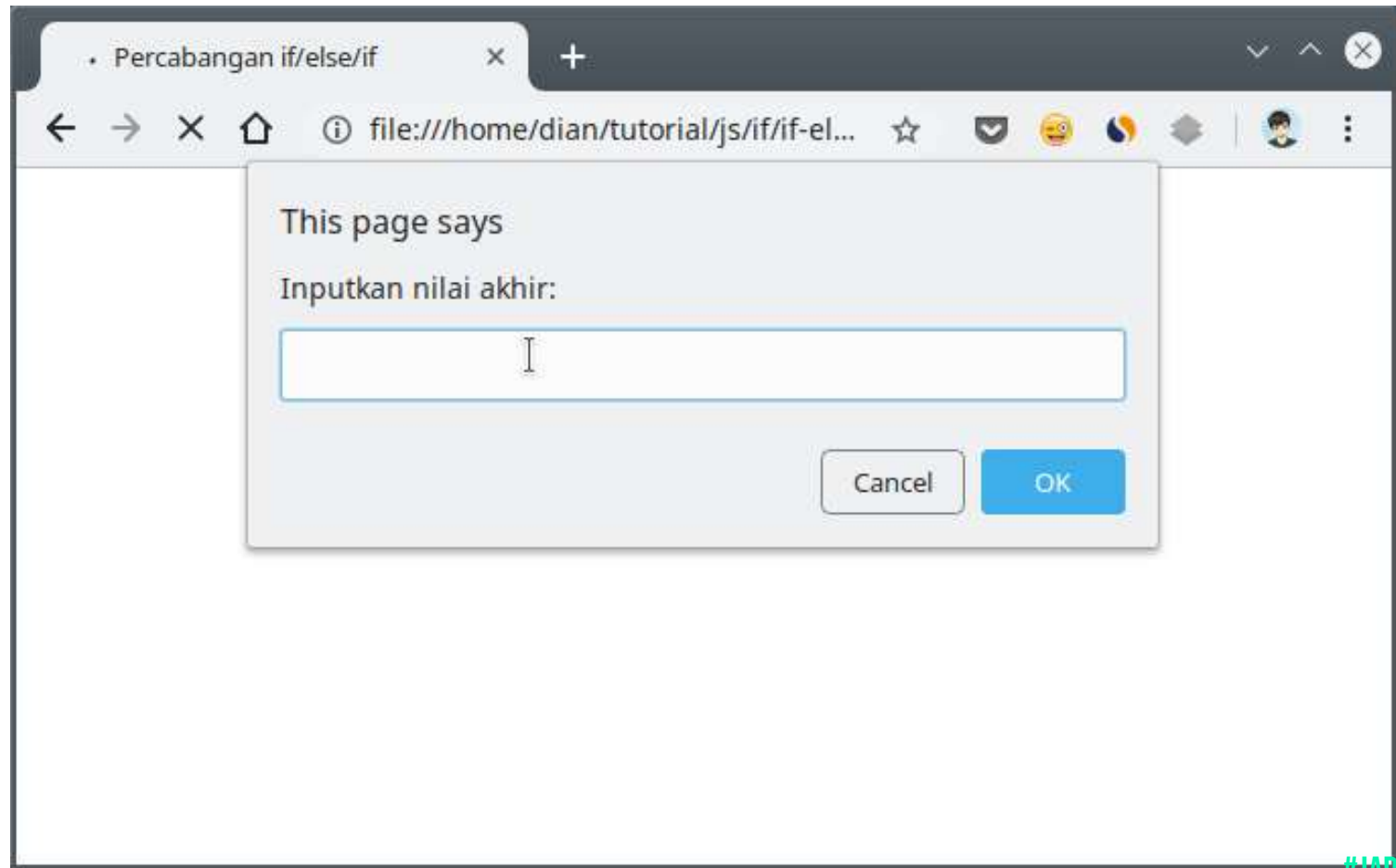
# Contoh:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Percabangan if/else/if</title>
</head>
<body>
  <script>
    var nilai = prompt("Inputkan nilai akhir:");
    var grade = "";

    if(nilai >= 90) grade = "A"
    else if(nilai >= 80) grade = "B+"
    else if(nilai >= 70) grade = "B"
    else if(nilai >= 60) grade = "C+"
    else if(nilai >= 50) grade = "C"
    else if(nilai >= 40) grade = "D"
    else if(nilai >= 30) grade = "E"
    else grade = "F";

    document.write(`<p>Grade anda: ${grade}</p>`);
  </script>
</body>
</html>
```

## Hasil:



## 4. Percabangan *switch/case*

Percabangan *switch/case* adalah bentuk lain dari percabangan *if/else/if*. Strukturnya seperti ini:

```
switch(variabel){
    case <value>:
        // blok kode
        break;
    case <value>:
        // blok kode
        break;
    default:
        // blok kode
}
```

- blok kode (case) dapat dibuat sebanyak yang dibutuhkan blok switch.
- <value>, diisi dengan nilai yang dibandingkan variabel.
- Setiap case harus diakhiri dengan break. Default tidak memerlukan break.
- Pemberian break bertujuan agar program berhenti mengecek case berikutnya saat sebuah case terpenuhi.

# Contoh:

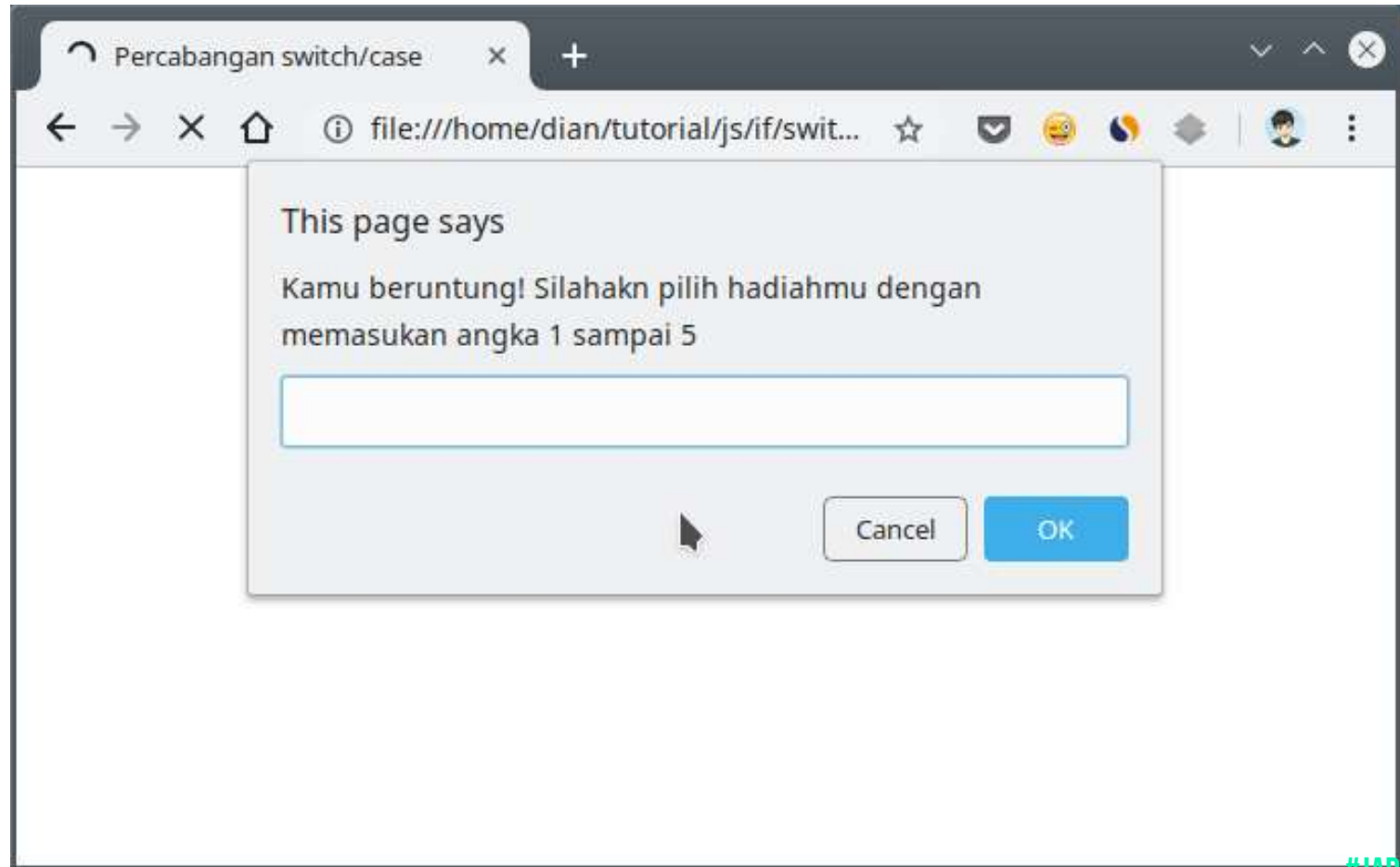
```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Percabangan switch/case</title>
</head>
<body>
  <script>

    var jawab = prompt("Kamu beruntung! Silahkn pilih hadiahmu dengan memasukan angka 1 sampai 5");
    var hadiah = "";

    switch(jawab){
      case "1":
        hadiah = "Tisu";
        break;
      case "2":
        hadiah = "1 Kotak Kopi";
        break;
      case "3":
        hadiah = "Sticker";
        break;
```

```
    case "4":  
        hadiah = "Minyak Goreng";  
        break;  
    case "5":  
        hadiah = "Uang Rp 50.000";  
        break;  
    default:  
        document.write("<p>Opps! anda salah pilih</p>");  
}  
  
if(hadiah === ""){  
    document.write("<p>Kamu gagal mendapat hadiah</p>");  
} else {  
    document.write("<h2>Selamat kamu mendapatkan " + hadiah + "</h2>");  
}  
</script>  
</body>  
</html>
```

# Hasil:





Percabangan *switch/case* juga dapat dibuat seperti ini:

```
var nilai = prompt("input nilai");
var grade = "";

switch(true){
    case nilai < 90:
        grade = "A";
        break;
    case nilai < 80:
        grade = "B+";
        break;
    case nilai < 70:
        grade = "B";
        break;
    case nilai < 60:
        grade = "C+";
        break;
```

```
    case nilai < 50:
        grade = "C";
        break;
    case nilai < 40:
        grade = "D";
        break;
    case nilai < 30:
        grade = "E";
        break;
    default:
        grade = "F";
}
```

Pertama. berikan nilai true pada switch, ini agar bisa masuk ke dalam blok switch.

Lalu di dalam blok switch, kita buat kondisi dengan menggunakan case.

Hasilnya akan sama seperti pada contoh percabangan if/else/if.

## 5. Percabangan dengan Operator Ternary

Percabangan menggunakan operator *ternary* merupakan bentuk lain dari percabangan *if/else*. Bisa dibilang bentuk singkatnya dari *if/else*.

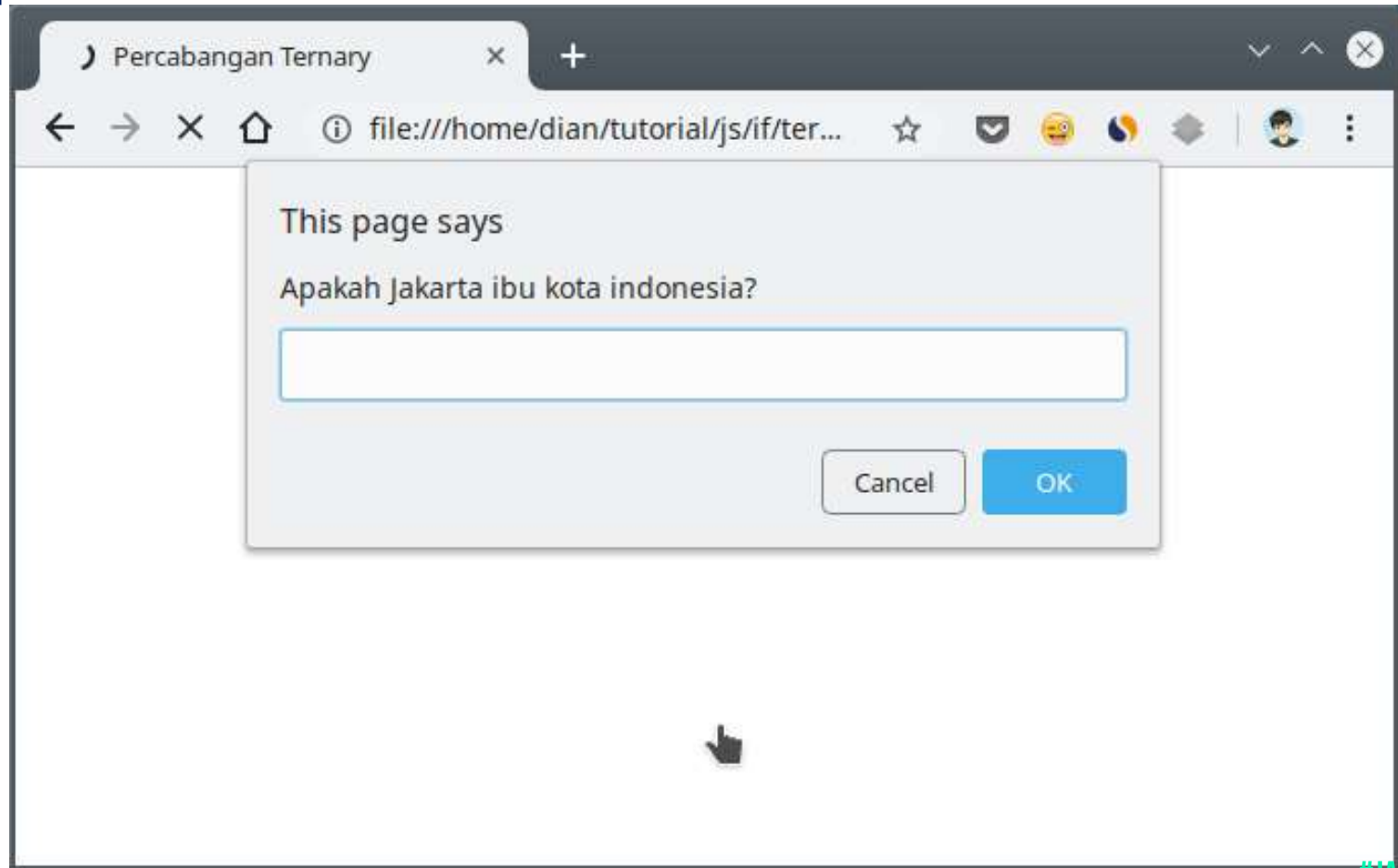
```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Percabangan Ternary</title>
</head>
<body>
  <script>
    var jwb = prompt("Apakah Jakarta ibu kota indonesia?");

    var jawaban = (jwb.toUpperCase() == "IYA") ? "Benar": "Salah";

    document.write(`Jawaban anda: <b>${jawaban}</b>`);
  </script>
</body>
</html>
```

Fungsi method  
toUpperCase()  
mengubah teks yang  
diinput menjadi capital  
semua

Hasil:



Operator ternary berperan sebagai percabangan if/else:

```
var jawaban = (jwb.toUpperCase() == "IYA") ? "Benar": "Salah";
```

Apabila kondisi yang ada di dalam kurung—`(jwb.toUpperCase() == "IYA")`— bernilai **true**, maka nanti isi dari variabel jawaban akan sama dengan "**Benar**".

Tapi kalau bernilai false, maka variabel jawaban akan berisi "**Salah**".

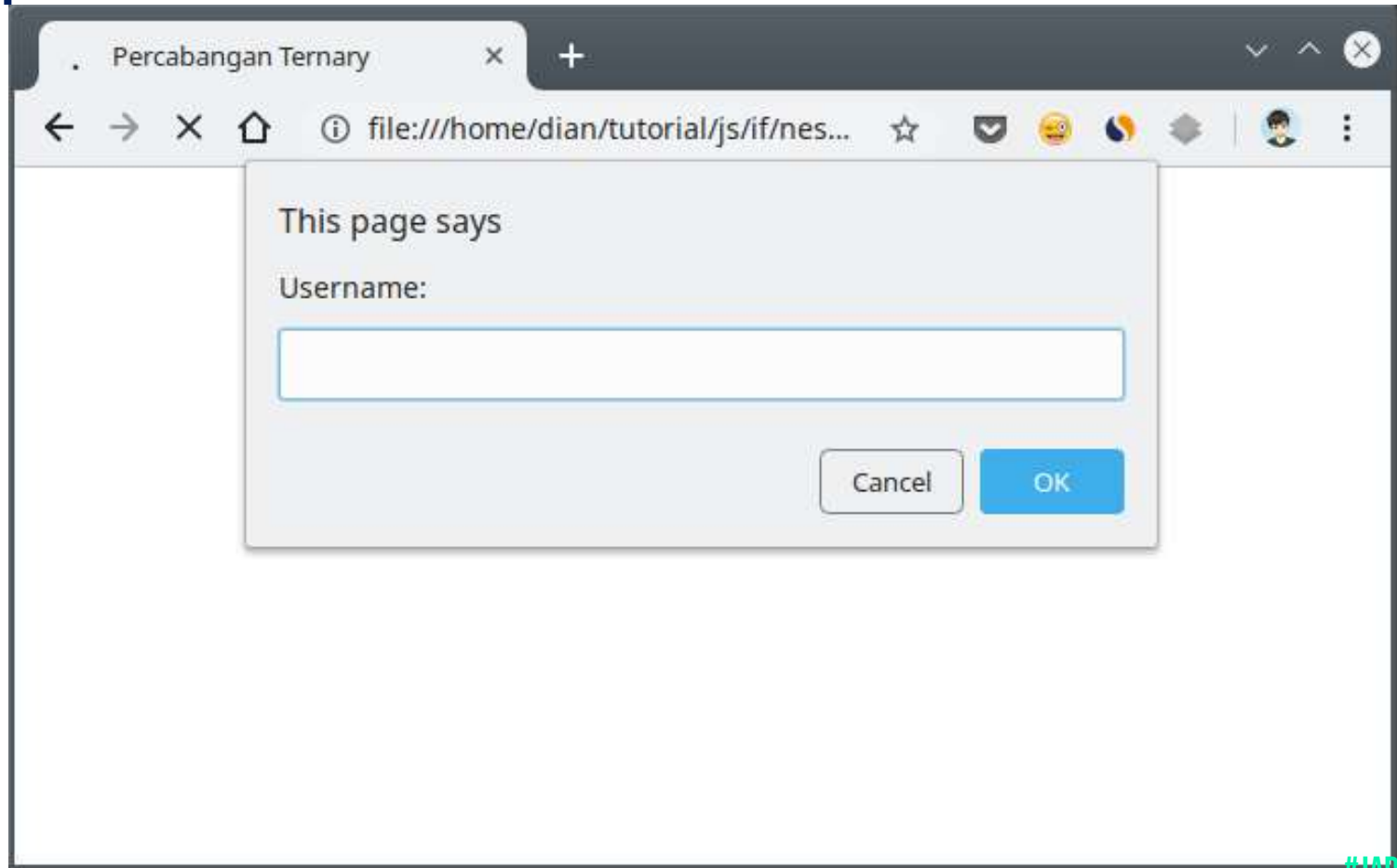
## 6. Percabangan Bersarang (Nested)

blok percabangan juga dapat dibuat di dalam percabangan.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Percabangan Ternary</title>
</head>
<body>
  <script>
    var username = prompt("Username:");
    var password = prompt("Password:");

    if(username == "petanikode"){
      if(password == "kopi"){
        document.write("<h2>Selamat datang pak bos!</h2>");
      } else {
        document.write("<p>Password salah, coba lagi!</p>");
      }
    } else {
      document.write("<p>Anda tidak terdaftar!</p>");
    }
  </script>
</body>
</html>
```

Hasil:



## Menggunakan Operator Logika pada Percabangan

Percabangan bersarang dapat buat lebih sederhana dengan operator logika. Contoh:

```
var username = prompt("Username:");
var password = prompt("Password:");

if(username == "petanikode"){
    if(password == "kopi"){
        document.write("<h2>Selamat datang pak bos!</h2>");
    } else {
        document.write("<p>Password salah, coba lagi!</p>");
    }
} else {
    document.write("<p>Anda tidak terdaftar!</p>");
}
```



# Operator Logika

Dapat di sederhanakan dengan operator logika AND (&&).

```
var username = prompt("Username:");
var password = prompt("Password:");

if(username == "petanikode" && password == "kopi"){
    document.write("<h2>Selamat datang pak bos!</h2>");
} else {
    document.write("<p>Password salah, coba lagi!</p>");
}
```

Namun, ini bukan solusi terbaik karena user tidak bisa di cek terdaftar atau tidak.

# Blok Perulangan Javascript

# Pengenalan

Perulangan akan membantu kita mengeksekusi kode yang berulang-ulang, berapapun yang kita mau. Ada 5 macam bentuk perulangan di Javascript. Secara umum, perulangan dibagi 2. *Counted loop* dan *uncounted loop*. Perbedaannya:

- Counted Loop merupakan perulangan yang jelas dan sudah tentu banyak perulangannya.
- Sedangkan Uncounted Loop, merupakan perulangan yang tidak jelas berapa kali ia harus mengulang.

Perulangan yang termasuk dalam *Counted Loop*:

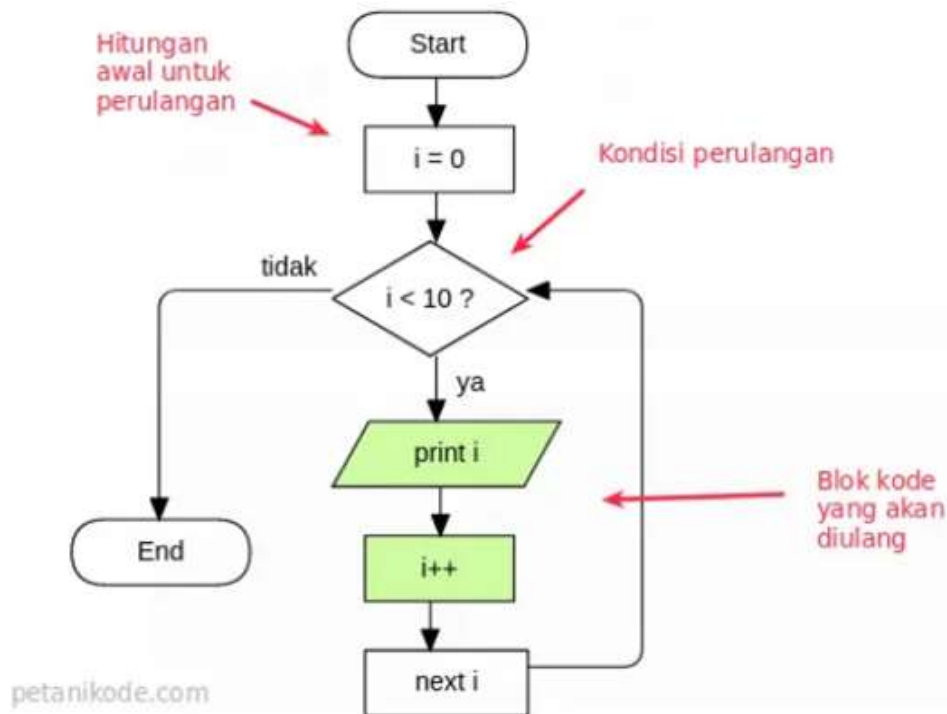
- Perulangan For
- Perulangan Foreach
- Perulangan Repeat

Perulangan yang termasuk dalam *Uncounted Loop*:

- Perulangan While
- Perulangan Do/While

# 1. Perulangan For di Javascript

Perulangan for merupakan perulangan yang termasuk dalam counted loop, karena sudah jelas berapa kali ia akan mengulang.



Contoh:

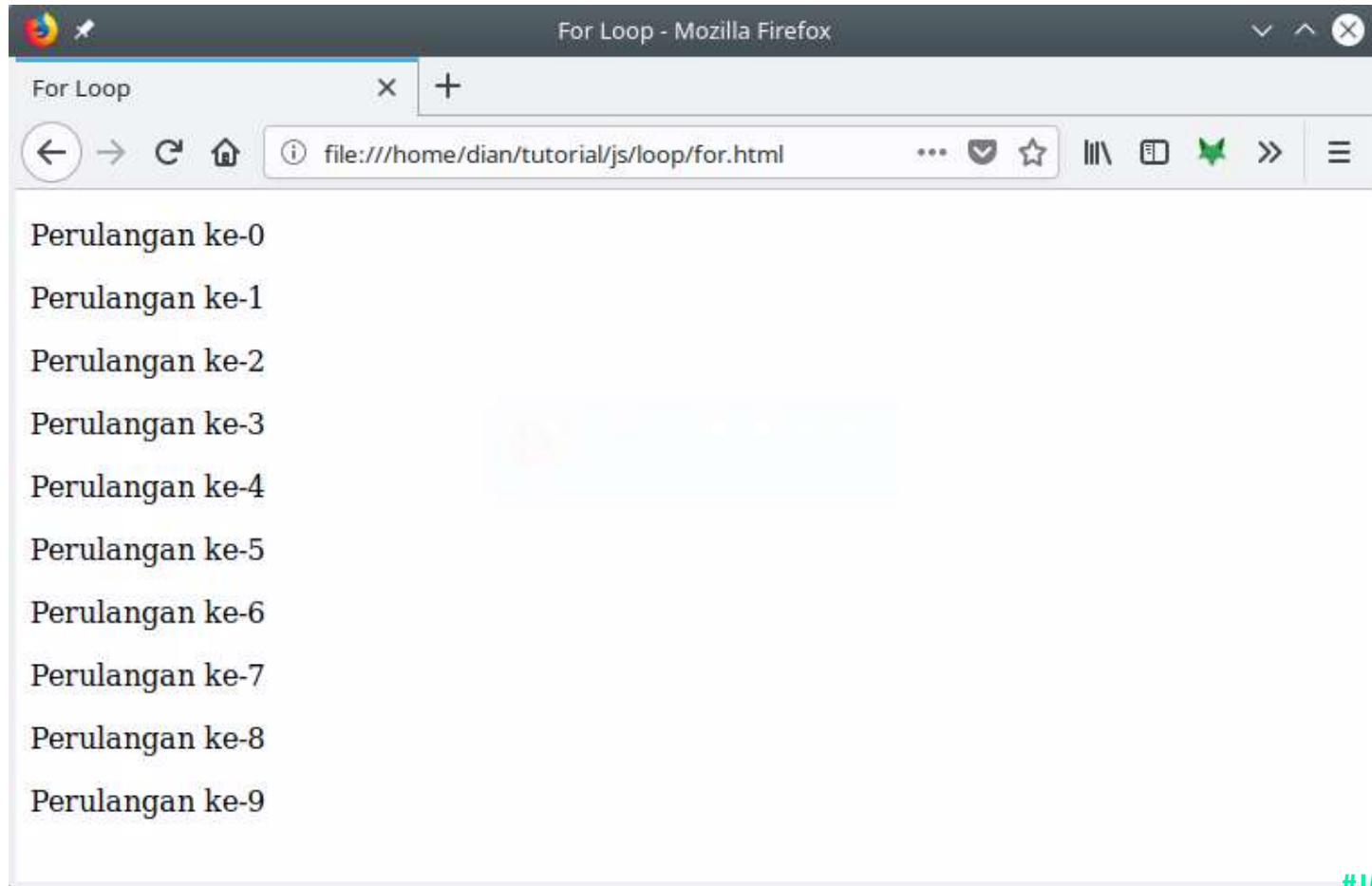
```
for(let i = 0; i < 10; i++){  
    document.write("<p>Perulangan ke-" + i + "</p>")  
}
```

Yang perlu diperhatikan adalah kondisi yang ada di dalam kurung setelah kata for, kondisi ini akan menentukan:

- Hitungan dimulai dari 0 ( $i = 0$ );
- Hitungannya Sampai  $i < 10$ ;
- di setiap perulangan  $i$  akan  $+1$  ( $i++$ ).

Variabel  $i$  pada perulangan for berfungsi untuk menyimpan nilai hitungan. Jadi setiap perulangan dilakukan nilai  $i$  akan selalu bertambah satu. Karena kita menentukannya di bagian  $i++$ .

## Hasil:

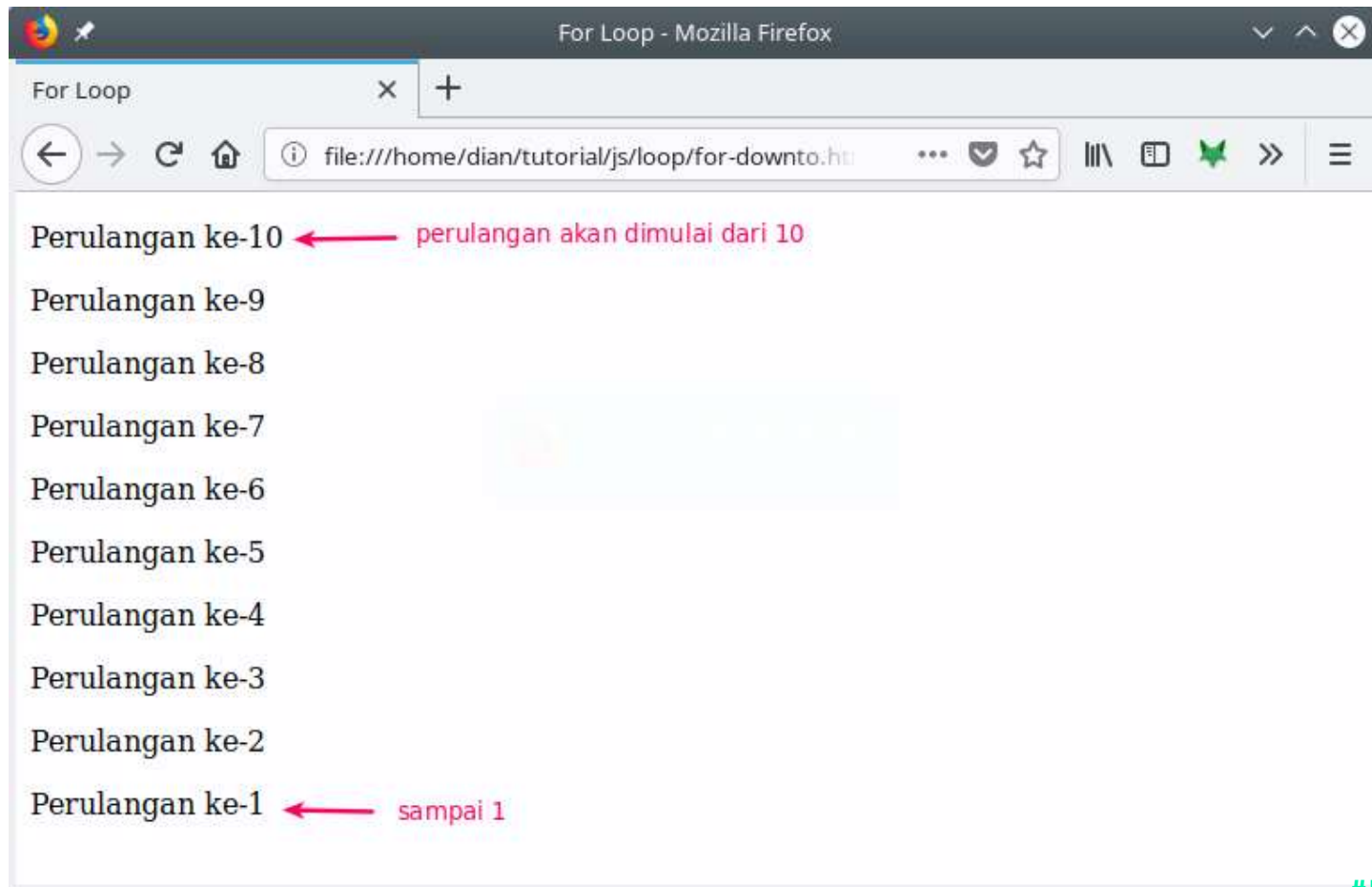


Counter juga bisa menghitung mundur, contoh:

```
for(counter = 10; counter > 0; counter--){  
    document.write("<p>Perulangan ke "+counter+"</p>");  
}
```



Hasil:



## 2. Perulangan While di Javascript

Perulangan while merupakan perulangan yang termasuk dalam perulangan uncounted loop. Perulangan while juga dapat menjadi perulangan yang counted loop dengan memberikan counter di dalamnya.

```
var ulangi = confirm("Apakah anda mau mengulang?");
var counter = 0;

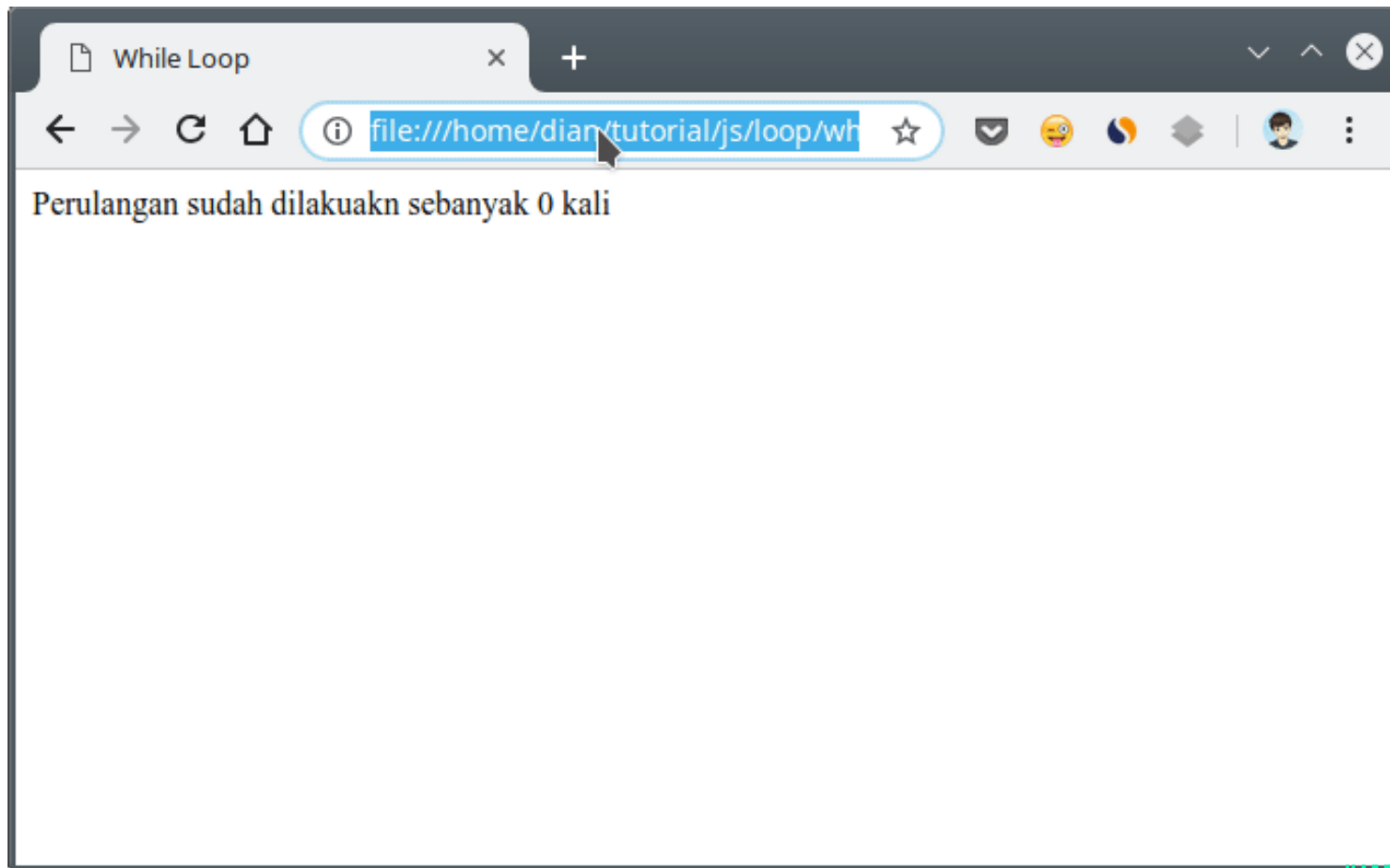
while(ulangi){
    var jawab = confirm("Apakah anda mau mengulang?")
    counter++;
    if(jawab == false){
        ulangi = false;
    }
}

document.write("Perulangan sudah dilakuakn sebanyak "+ counter + " kali")
```

Dapat disederhanakan menjadi:

```
var ulangi = confirm("Apakah anda mau mengulang?");  
var counter = 0;  
  
while(ulangi){  
    counter++;  
    ulangi = confirm("Apakah anda mau mengulang?");  
}  
  
document.write("Perulangan sudah dilakuakn sebanyak "+ counter +" kali");
```

Hasil:



### 3. Perulangan Do/While di Javascript

Perulangan do/while sama seperti perulangan while.

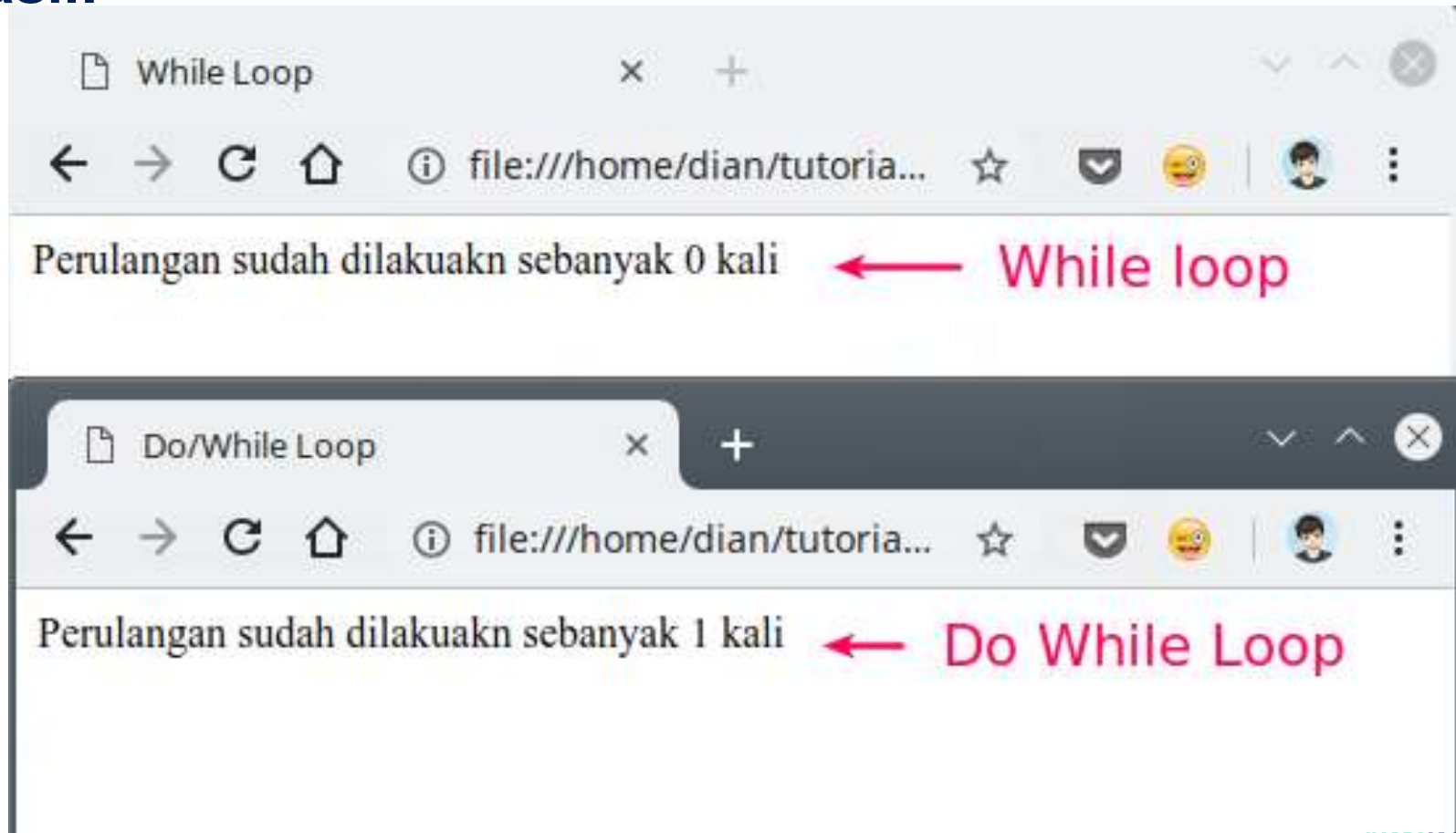
Perbedaannya adalah perulangan do/while akan melakukan perulangan sebanyak 1 kali terlebih dahulu, lalu mengecek kondisi yang ada di dalam kurung while.

```
do {  
    // blok kode yang akan diulang  
} while (<kondisi>);
```

## Contoh:

```
var ulangi = confirm("Apakah anda mau mengulang?");  
var counter = 0;  
  
do {  
    counter++;  
    ulangi = confirm("Apakah anda mau mengulang?");  
} while(ulangi)  
  
document.write("Perulangan sudah dilakuakn sebanyak "+ counter + " kali")
```

## Hasil:



## 4. Perulangan Foreach di Javascript

Perulangan foreach biasanya digunakan untuk mencetak item di dalam array. Perulangan ini termasuk dalam perulangan counted loop, karena jumlah perulangannya akan ditentukan oleh panjang dari array.

Ada dua cara menggunakan perulangan foreach di Javascript:

1. Menggunakan for dengan operator in;
2. Menggunakan method **forEach()**.



## Contoh:

Berikut ini bentuk perulangan “foreach” tanpa menggunakan operator in:

```
var languages = ["Javascript", "HTML", "CSS", "Typescript"];

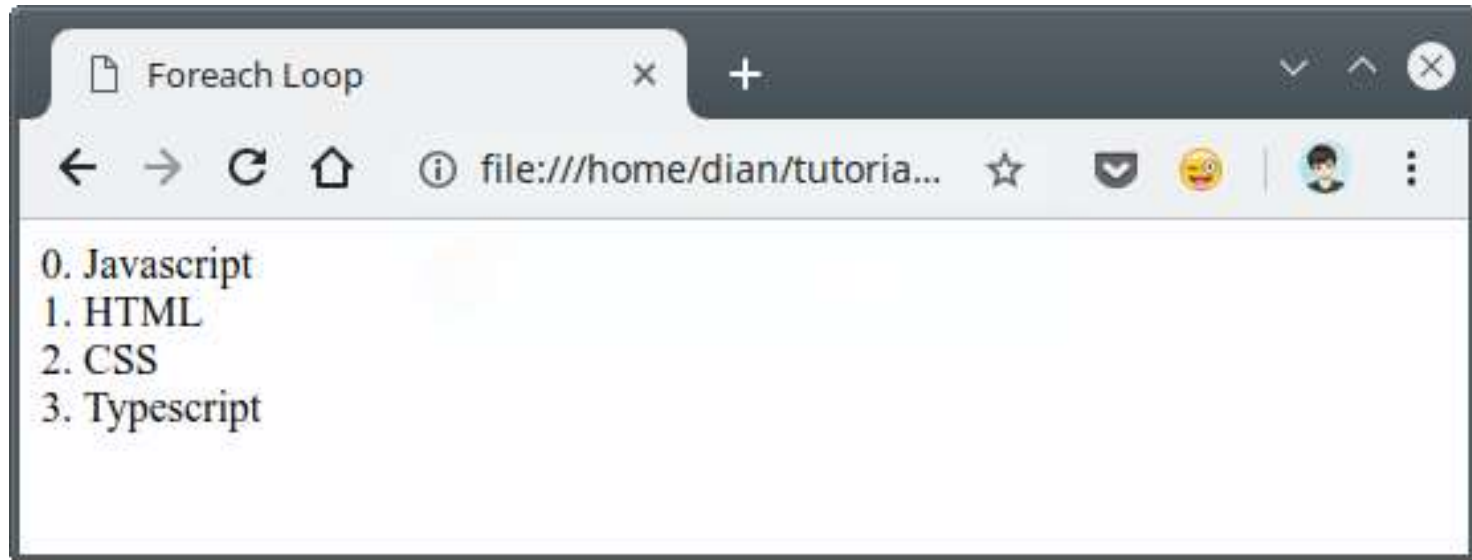
for(i = 0; i < languages.length; i++){
    document.write(i + ". " + languages[i] + "<br/>");
}
```

Perulangan ini dapat dibuat lebih sederhana lagi dengan menggunakan operator in seperti ini:

```
var languages = ["Javascript", "HTML", "CSS", "Typescript"];

for(i in languages){
    document.write(i+". " + languages[i] + "<br/>");
}
```

# Hasil:



Cara kedua membuat perulangan foreach ialah dengan menggunakan method **forEach()** dari array. Contoh:

```
// kita punya array seperti berikut
var days = ["Senin", "Selasa", "Rabu", "Kamis", "Jum'at", "Sabtu", "Minggu"];

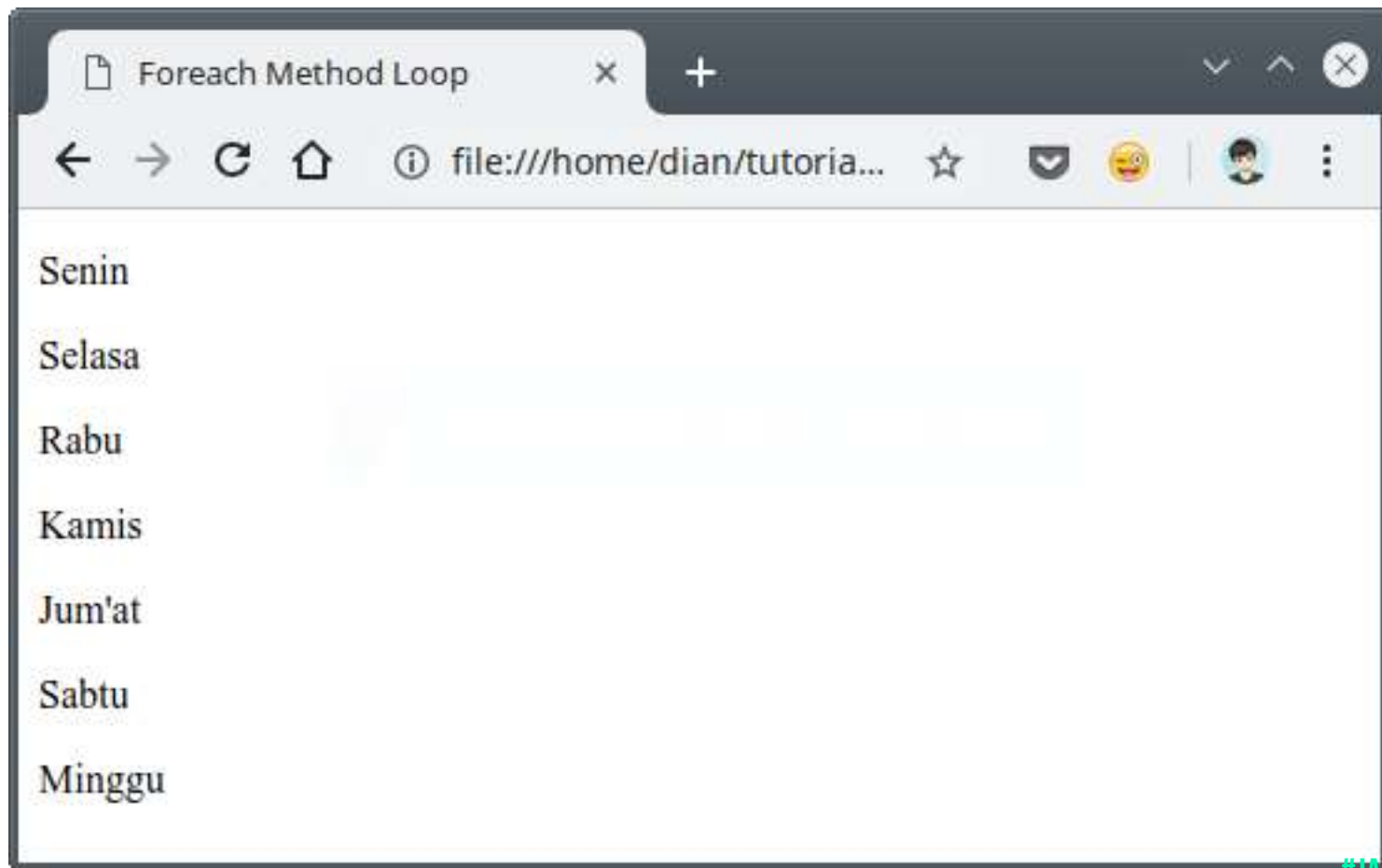
// Kemudian kita tampilkan semua hari
// dengan menggunakan method foreach
days.forEach(function(day){
    document.write("<p>" + day + "</p>");
});
```

Method **forEach()** memiliki parameter berupa fungsi callback. Sebenarnya kita juga bisa menggunakan arrow function seperti ini:

```
// kita punya array seperti berikut
var days = ["Senin", "Selasa", "Rabu", "Kamis", "Jum'at", "Sabtu", "Minggu"];

// Kemudian kita tampilkan semua hari
// dengan menggunakan method foreach
days.forEach((day) => {
  document.write("<p>" + day + "</p>");
});
```

Hasil:



## 5. Perulangan dengan Method repeat()

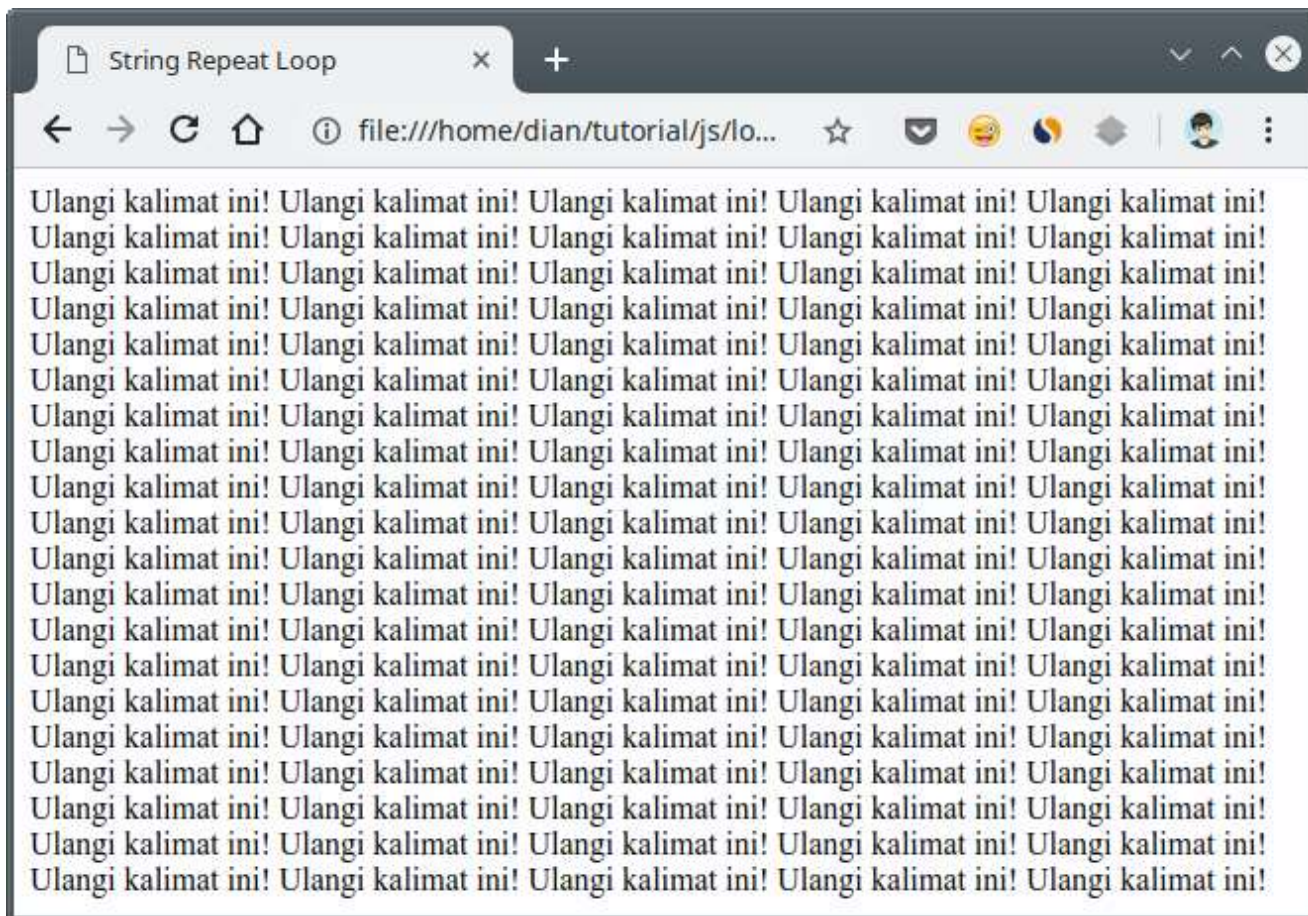
Perulangan dengan method atau fungsi **repeat()** termasuk dalam perulangan counted loop. Fungsi ini khusus digunakan untuk mengulang sebuah teks (string). Bisa dibilang ini merupakan peningkatan perulangan for. Contoh dengan for:

```
for( let i = 0; i < 100; i++){
    document.write("Ulangi kalimat ini!");
}
```

Dengan **repeat()**

```
document.write("Ulangi kalimat ini! ".repeat(100));
```

# Hasil:



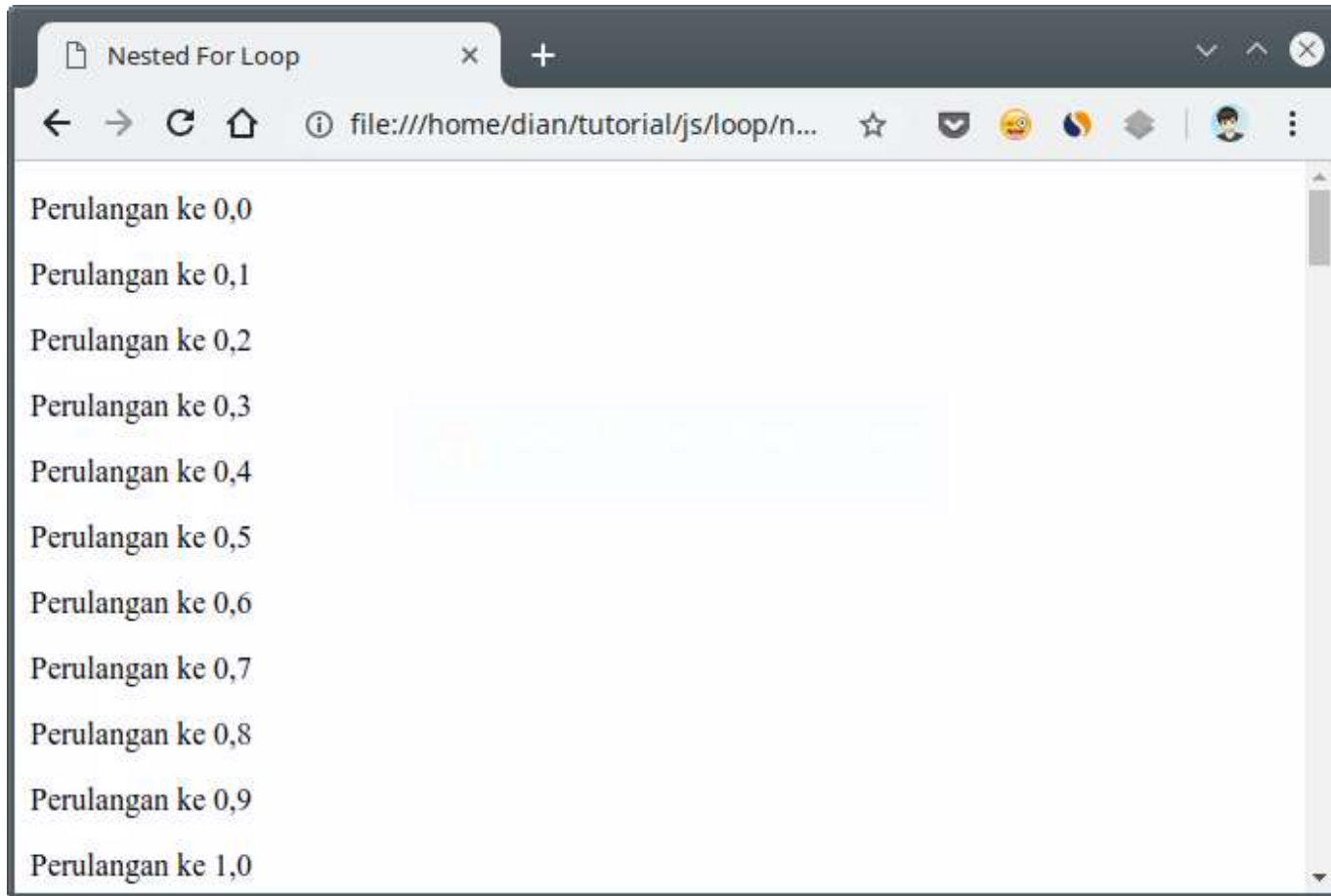


# Perulangan Bersarang (*Nested*)

Perulangan dalam perulangan :

```
for(let i = 0; i < 10; i++){  
  for(let j = 0; j < 10; j++){  
    document.write("<p>Perulangan ke " + i + "," + j + "</p>");  
  }  
}
```

# Hasil:



Pada perulangan tersebut, dua perulangan for digunakan.

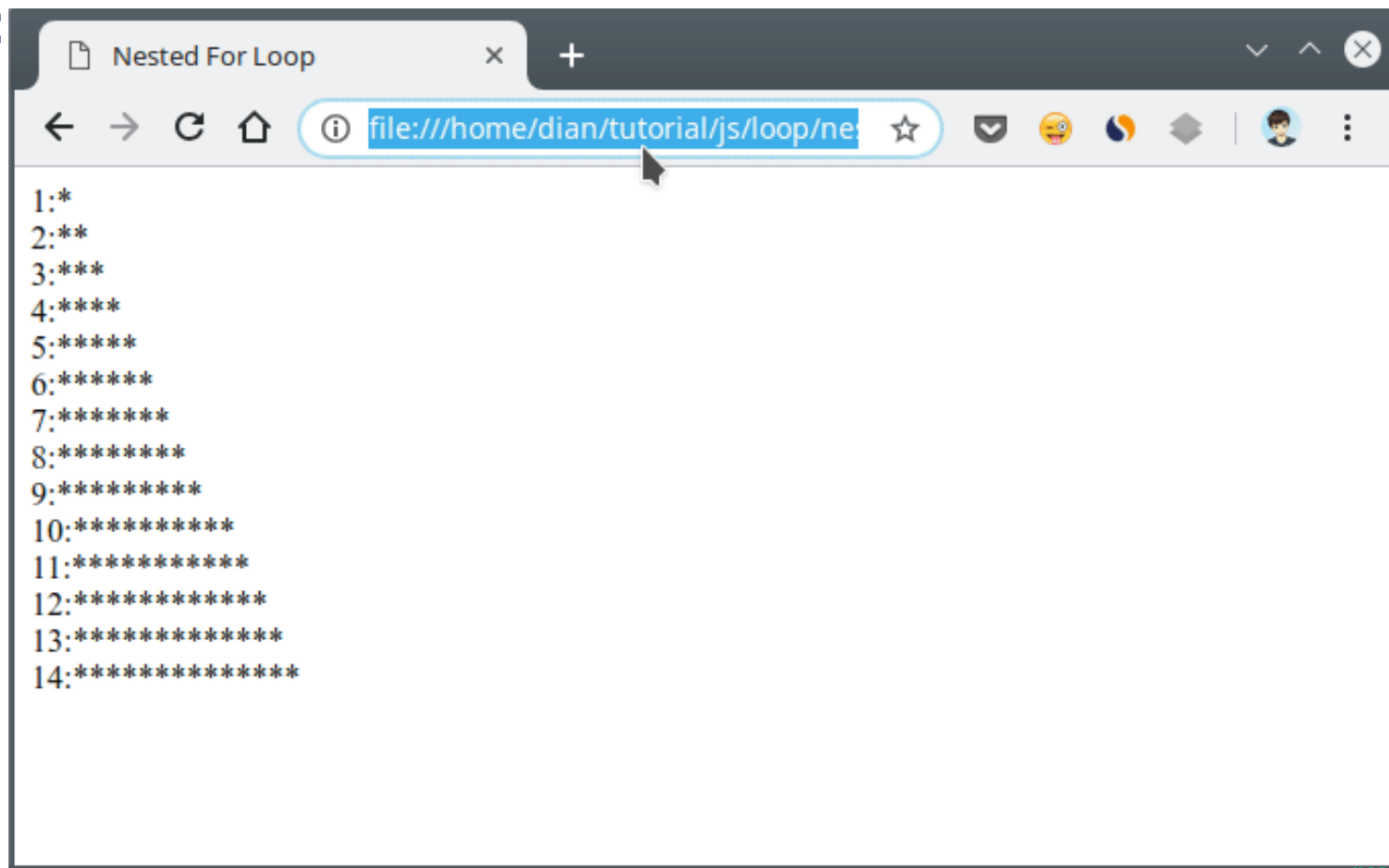
- Perulangan pertama menggunakan variabel i sebagai counter.
- Perulangan kedua menggunakan variabel j sebagai counter.

Contoh lain:

```
var ulangi = confirm("apakah anda ingin mengulang?");
var counter = 0;

while (ulangi) {
    counter++;
    var bintang = "*".repeat(counter) + "<br>";
    document.write(counter + ": " + bintang);
    ulangi = confirm("apakah anda ingin mengulang?");
}
```

Hasil:



The screenshot shows a web browser window with a single tab titled "Nested For Loop". The address bar displays the file path `file:///home/dian/tutorial/js/loop/ne`. The browser's toolbar includes navigation buttons (back, forward, refresh, home), an address bar with an information icon, a star icon for bookmarks, and several extension icons (mail, emoji, search, etc.). The main content area of the browser displays the output of a nested loop, consisting of 14 lines of text. Each line starts with a number from 1 to 14, followed by a colon and a series of asterisks. The number of asterisks increases by one for each line, starting from 1 asterisk on line 1 and ending with 14 asterisks on line 14.

```

1:*
2:**
3:***
4:****
5:*****
6:*****
7:*****
8:*****
9:*****
10:*****
11:*****
12:*****
13:*****
14:*****
  
```

# Struktur Data Array

# Pengenalan

Struktur data merupakan cara-cara atau metode yang digunakan untuk menyimpan data di dalam memori komputer.

Salah satu struktur data yang sering digunakan dalam pemrograman adalah Array.

Array merupakan struktur data yang digunakan untuk menyimpan sekumpulan data dalam satu tempat.

# Membuat Array pada Javascript

Pada javascript, array dapat kita buat dengan tanda kurung siku ([...]).

```
var products = [];
```

Maka variabel products akan berisi sebuah array kosong.

Kita bisa mengisi data ke dalam array, lalu setiap data dipisah dengan tanda koma (,).

```
var products = ["Flashdisk", "SDD", "Monitor"];
```

Dan karena Javascript merupakan Bahasa pemrograman yang dynamic typing, data yang disimpan dapat dicampur dengan tipe data lain yang berbeda di dalam array .

```
var myData = [12, 2.1, true, 'C', "Petanikode"];
```



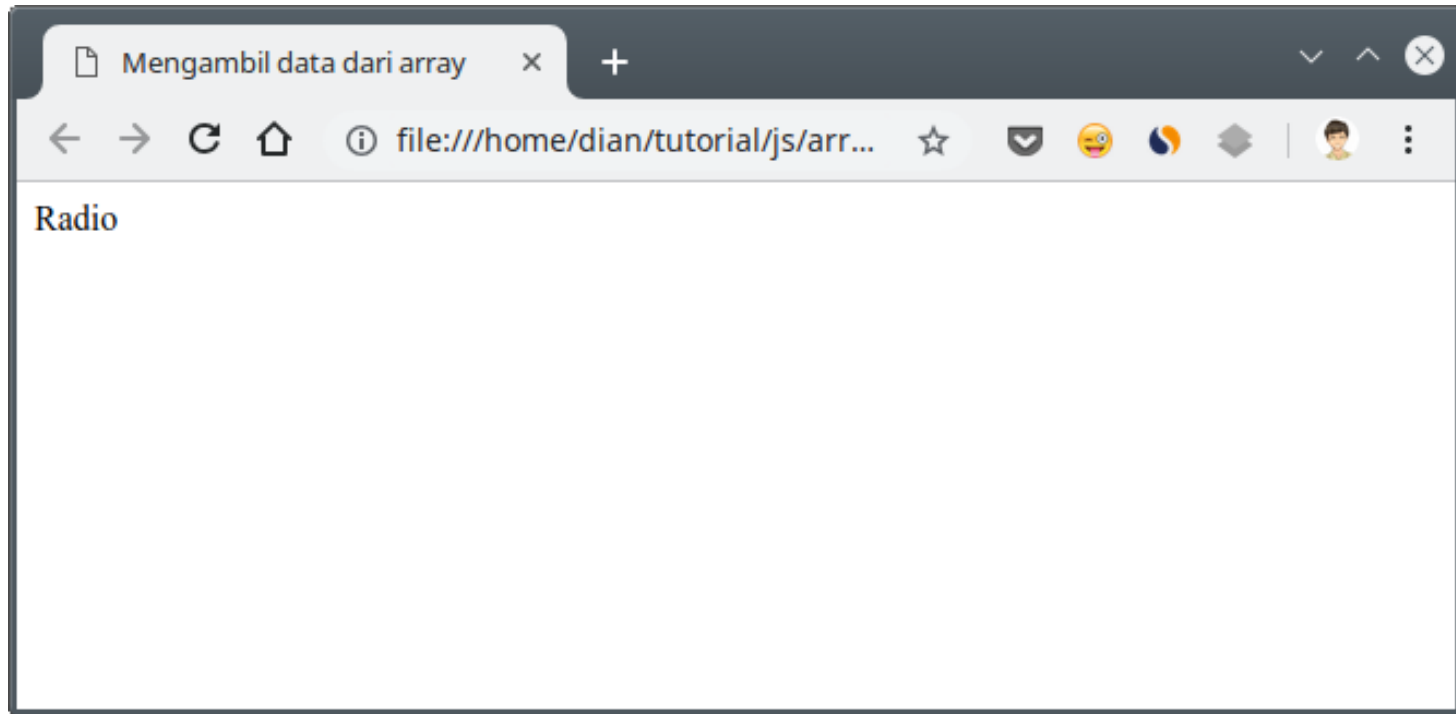
# Mengambil Data dari Array

Ingat bahwa array dimulai dari nol (0). Contoh:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Mengambil data dari array</title>
</head>
<body>
  <script>
    // membuat array
    var products = ["Senter", "Radio", "Antena", "Obeng"];

    // mengambil radio
    document.write(products[1]);
  </script>
</body>
</html>
```

# Hasil:



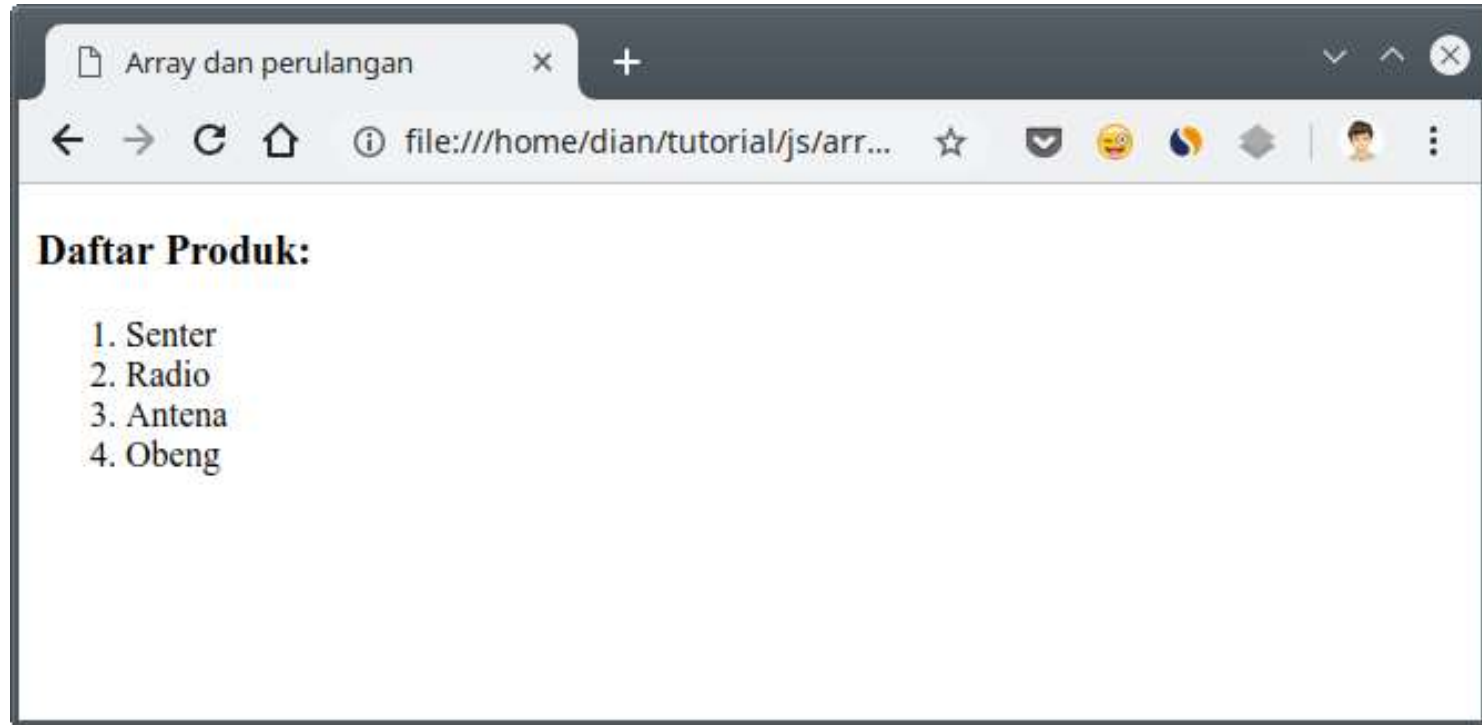
# Mencetak isi Array dengan Perulangan

Dapat digunakan perulangan bila input yang diambil berasal dari array yang berjumlah banyak. Contoh:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Array dan perulangan</title>
</head>
<body>
  <script>
    // membuat array
    var products = ["Senter", "Radio", "Antena", "Obeng"];

    document.write("<h3>Daftar Produk:</h3>");
    document.write("<ol>");
    // menggunakan perulangan untuk mencetak semua isi array
    for(let i = 0; i < products.length; i++){
      document.write("<li>${ products[i] }</li>");
    }
    document.write("</ol>");
  </script>
</body>
</html>
```

# Hasil:



Cara lainnya adalah dengan menggunakan method **forEach()**:

Hasilnya akan sama seperti sebelumnya.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Array dan perulangan</title>
</head>
<body>
  <script>
    // membuat array
    var products = ["Senter", "Radio", "Antena", "Obeng"];

    document.write("<h3>Daftar Produk:</h3>");
    document.write("<ol>");
    // menggunakan perulangan untuk mencetak semua isi array
    products.forEach((data) => {
      document.write("<li>${data}</li>");
    });
    document.write("</ol>");
  </script>
</body>
</html>
```

# Menambahkan Data ke Dalam Array

Ada dua cara yang bisa dilakukan untuk menambah data ke dalam array:

1. Mengisi menggunakan indeks;
2. Mengisi menggunakan method `push()`.

# 1. Mengisi menggunakan indeks

Terdapat tiga data di dalam array buah dengan indeks:

```
var buah = ["Apel", "Jeruk", "Manggis"];
```

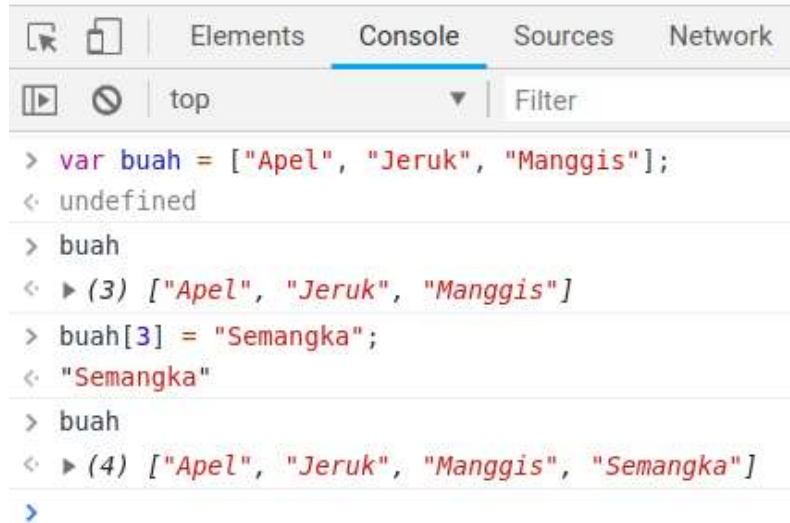
- 0: "Apel"
- 1: "Jeruk"
- 2: "Manggis"

Kita ingin menambahkan data lagi pada indeks ke-3, maka kita bisa melakukannya seperti ini:

```
buah[3] = "Semangka";
```

# Referensi

Percobaan dengan console:



```

> var buah = ["Apel", "Jeruk", "Manggis"];
< undefined
> buah
< ▶ (3) ["Apel", "Jeruk", "Manggis"]
> buah[3] = "Semangka";
< "Semangka"
> buah
< ▶ (4) ["Apel", "Jeruk", "Manggis", "Semangka"]
>

```

Ingat bahwa harus diketahui jumlah data dan panjang array untuk mengguakan cara ini. Bila nomor index diisi sembarangan data akan tertindih.



## 2. Mengisi Menggunakan Method **push()**

Solusi masalah sebelumnya adalah dengan menggunakan method **push()**. Tidak perlu diketahui berapa panjang array karena method **push()** akan menambahkan data ke dalam array dari ekor atau belakang.

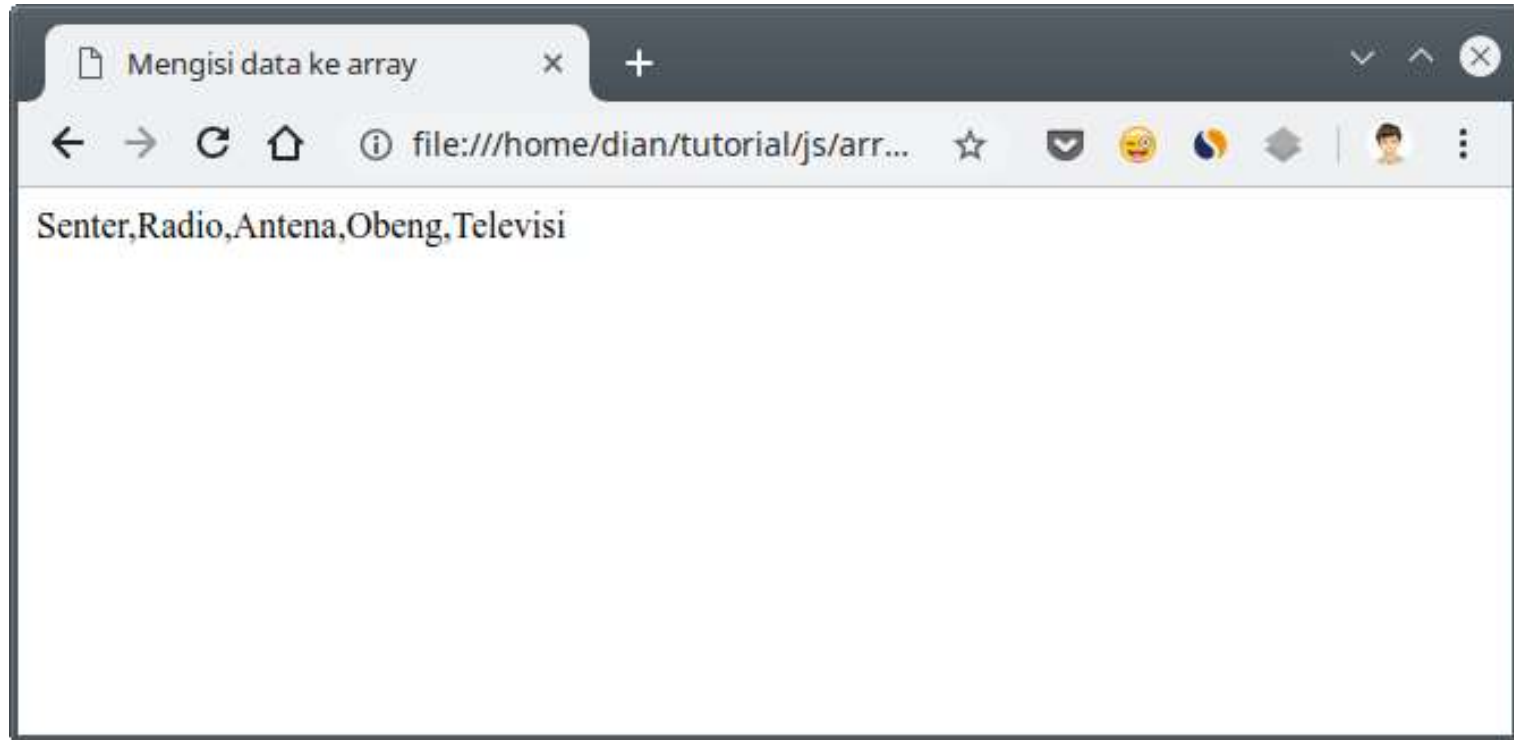
# Contoh:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Mengisi data ke array</title>
</head>
<body>
  <script>
    // membuat array
    var products = ["Senter", "Radio", "Antena", "Obeng"];

    // menambahkan tv ke dalam array products
    products.push("Televisi");

    // menampilkan isi array
    document.write(products);
  </script>
</body>
</html>
```

# Hasil:



# Menghapus Data Array

Sama seperti menambahkan data ke array, menghapus data juga memiliki dua cara:

1. Menggunakan delete;
2. Menggunakan method pop().

Contoh:

```
delete buah[2];
```

Kita dapat menghapus data dengan nomer indeks tertentu dengan delete. Sedangkan pop() akan menghapus dari belakang. Kekurangan dari delete, ia akan menciptakan ruang kosong di dalam array. Lihat percobaan di console berikut:

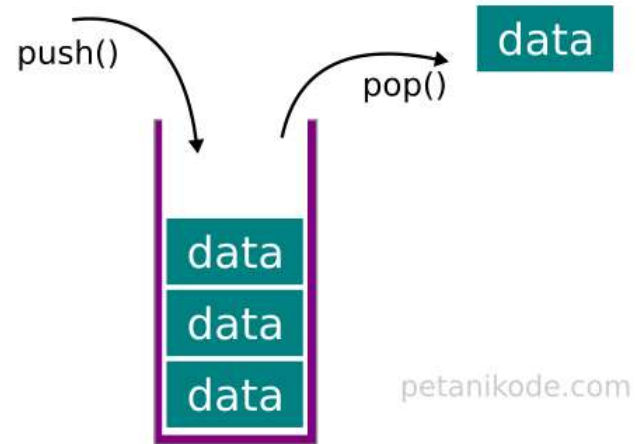


```
> buah
< ▶ (4) ["Apel", "Jeruk", "Manggis", "Semangka"]
> delete buah[2]
< true
> buah
< ▶ (4) ["Apel", "Jeruk", empty, "Semangka"]
> buah[2]
< undefined
> buah[3]
< "Semangka"
```

Cara kedua menggunakan method `pop()`, kebalikan dari method `push()`.

Method `pop()` akan menghapus array yang ada di paling belakang.

Array pada javascript dapat kita pandang sebagai sebuah stack (tumpukan), yang mana memiliki sifat LILO (Last in Last out).



```

> buah
< ▶ (4) ["Apel", "Jeruk", empty, "Semangka"]
> buah.pop()
< "Semangka"
> buah.pop()
< undefined
> buah.pop()
< "Jeruk"
> buah
< ▶ ["Apel"]
> buah.pop()
< "Apel"
> buah
< ▶ []
  
```

Kita memanggil method `pop()` sebanyak 4 kali, maka array-nya akan kosong `[]`. Karena isinya hanya 4 saja.

Method `pop()` akan mengembalikan nilai item atau data yang terhapus dari array.

# Menghapus Data dari Depan

Kita juga dapat menghapus data dari depan dengan menggunakan method `shift()`. Contoh:

```
var bunga = ["Mawar", "Melati", "Anggrek", "Sakura"];  
  
// hapus data dari depan  
bunga.shift();
```

Maka data yang terhapus adalah “Mawar”



## Menghapus Data pada Indeks Tertentu

Apabila kita ingin menghapus data pada indeks tertentu, maka fungsi atau method yang digunakan adalah **splice()**.

Fungsi ini memiliki dua parameter yang harus diberikan:

```
array.splice(<indeks>, <total>);
```

1. **<indeks>** indeks dari data di array yang ingin dihapus
2. **<total>** jumlah data yang dihapus dari index tersebut.

# Contoh:

```
var bunga = ["Mawar", "Melati", "Anggrek", "Sakura"];  
  
// hapus Anggrek  
bunga.splice(2, 1);
```

```
> bunga  
◁ ▶ (4) ["Mawar", "Melati", "Anggrek", "Sakura"]  
> bunga.splice(2,1)  
◁ ▶ ["Anggrek"]  
  
> bunga  
◁ ▶ (3) ["Mawar", "Melati", "Sakura"]  
  
> bunga.splice(1)  
◁ ▶ (2) ["Melati", "Sakura"]  
  
> |
```

Apabila **<total>** tidak diisi, maka semua data dari indeks yang terpilih akan dihapus

# Method-mothod Array

Selain method-method atau fungsi yang sudah kita coba di atas, terdapat beberapa method dalam Array yang perlu kita ketahui.

# 1. Method **filter()**

Method **filter()** berfungsi untuk menyaring data dari array.

```
const angka = [1, 2, 3, 4, 5, 6, 7, 8, 9];

// Kita ambil data yang hanya habis dibagi dua saja
const filteredArray = angka.filter((item) => {return item % 2 === 0});

console.log(filteredArray) // -> [2, 4, 6, 8]
```

Pada contoh di atas, kita memberikan arrow function sebagai fungsi callback yang akan melakukan penyaringan terhadap array. Sebenarnya kita bisa buat lebih sederhana lagi seperti ini:

```
const filteredArray = angka.filter(item => item % 2 === 0);
```

## 2. Method **includes()**

<Method ini berfungsi untuk mengecek apakah sebuah data ada di dalam array atau tidak.

Contoh:

```
var tanaman = ["Padi", "Kacang", "Jagung", "Kedelai"];

// apakah kacang sudah ada di dalam array tanaman?
var adaKacang = tanaman.includes("Kacang");

console.log(adaKacang); // -> true

// apakah bayam ada?
var adaBayam = tanaman.includes("Bayam");

console.log(adaBayam); // -> false
```

### 3. Method **sort()**

Method `sort()` berfungsi untuk mengurutkan data pada array. Contoh:

```
var alfabet = ['a', 'f', 'z', 'e', 'r', 'g'];
var angka = [3, 1, 2, 6, 8, 5];

console.log(alfabet.sort()); // -> ["a", "e", "f", "g", "r", "z"]
console.log(angka.sort()); // -> [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

# Fungsi di Javascript

# Pengenalan

Fungsi adalah sub-program yang bisa digunakan kembali baik di dalam program itu sendiri, maupun di program yang lain.

Fungsi di dalam Javascript adalah sebuah objek. Karena memiliki properti dan juga *method*.



## 4 Cara Membuat Fungsi di Javascript

Ada 4 cara yang bisa kita lakukan untuk membuat fungsi di Javascript:

1. Menggunakan cara biasa;
2. Menggunakan ekspresi;
3. Menggunakan tanda panah (`=>`);
4. dan menggunakan Constructor.

## 1. Membuat Fungsi dengan Cara Biasa

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

## 2. Membuat Fungsi dengan Ekspresi

```
var namaFungsi = function(){  
    console.log("Hello World!");  
}
```

Kita menggunakan variabel, lalu diisi dengan fungsi. Fungsi ini sebenarnya adalah fungsi anonim (*anonymous function*) atau fungsi tanpa nama.

### 3. Membuat Fungsi dengan Tanda Panah

```
var namaFungsi = () => {
    console.log("Hello World!");
}

// atau seperti ini (jika isi fungsi hanya satu baris):
var namaFungsi = () => console.log("Hello World!");
```

Sebenarnya hampir sama dengan yang menggunakan ekspresi. Bedanya, kita menggunakan tanda panah ( $\Rightarrow$ ) sebagai ganti function.

Pembuatan fungsi dengan cara ini disebut arrow function.

## 4. Membuat Fungsi dengan Kostruktur

Cara ini sebenarnya tidak direkomendasikan oleh Developer Mozilla, karena terlihat kurang bagus. Soalnya body fungsinya dibuat dalam bentuk string yang dapat mempengaruhi kinerja engine javascript. Contoh:

```
var namaFungsi = new Function('console.log("Hello World!");');
```

# Cara Memanggil/Eksekusi Fungsi

Kita bisa memanggil fungsi di dalam kode Javascript dengan menuliskan nama fungsinya seperti ini:

Contoh:

```
namaFungsi().
```

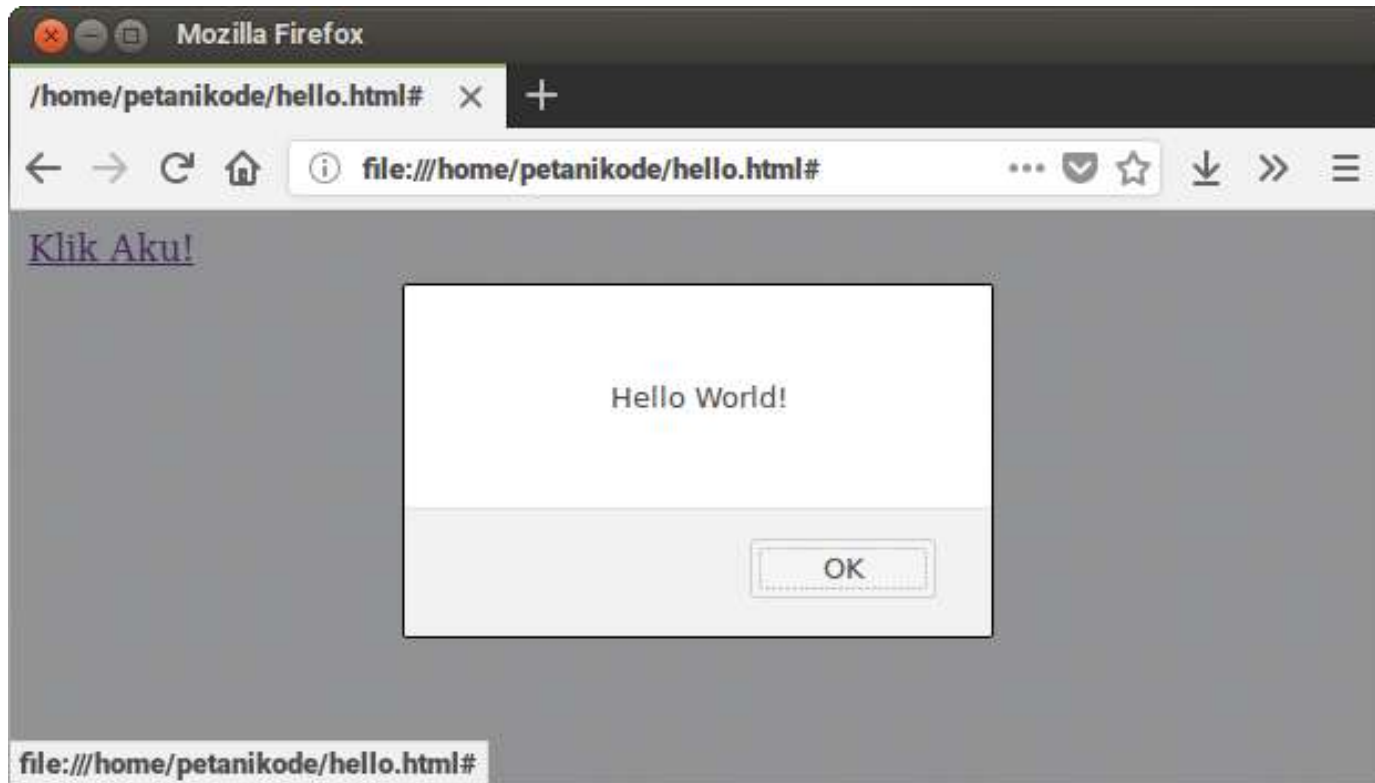
```
// membuat fungsi
function sayHello(){
    console.log("Hello World!");
}

// memanggil fungsi
sayHello() // maka akan menghasilkan -> Hello World!
```

Fungsi juga dapat di panggil melalui atribut *event* pada HTML. Contoh:

```
<!DOCTYPE html>
<html>
<head>
  <script>
    // membuat fungsi
    var sayHello = () => alert("Hello World!");
  </script>
</head>
<body>
  <!-- Memanggil fungsi saat link diklik -->
  <a href="#" onclick="sayHello()">Klik Aku!</a>
</body>
</html>
```

# Hasil:





# Fungsi dengan Parameter

Parameter adalah variabel yang menyimpan nilai untuk diproses di dalam fungsi.

Contoh:

```
function kali(a, b){  
    hasilKali = a * b;  
    console.log("Hasil kali a*b = " + hasilKali);  
}
```

Pada contoh di atas, a dan b adalah sebuah parameter. Lalu cara memanggil fungsi yang memiliki parameter adalah seperti ini:

```
kali(3, 2); // -> Hasil kali a*b = 6
```

Memberikan 3 untuk a, dan 2 untuk b

# Fungsi yang Mengembalikan Nilai

Agar hasil pengolahan nilai di dalam fungsi dapat digunakan untuk proses berikutnya, maka fungsi harus mengembalikan nilai. Pengembalian nilai pada fungsi menggunakan kata kunci `return` kemudian diikuti dengan nilai atau variabel yang akan dikembalikan. Contoh:

```
function bagi(a,b){  
    hasilBagi = a / b;  
    return hasilBagi;  
}  
  
// memanggil fungsi  
var nilai1 = 20;  
var nilai2 = 5;  
var hasilPembagian = bagi(nilai1, nilai2);  
  
console.log(hasilPembagian); //-> 4
```

## Contoh Program Javascript dengan Fungsi

Buat 2 file, fungsi.js dan index.html

```
js-fungsi/
├── fungsi.js
└── index.html
```

File index.html untuk menampilkan halaman web, dan fungsi.js adalah programnya.

# index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Belajar Fungsi di Javascript</title>
</head>
<body>

  <fieldset>
    <legend>Input Form</legend>
    <input type="text" name="barang" placeholder="input nama barang..." />
    <input type="button" onclick="addBarang()" value="Tambah" />
  </fieldset>

  <div>
    <ul id="list-barang">
    </ul>
  </div>

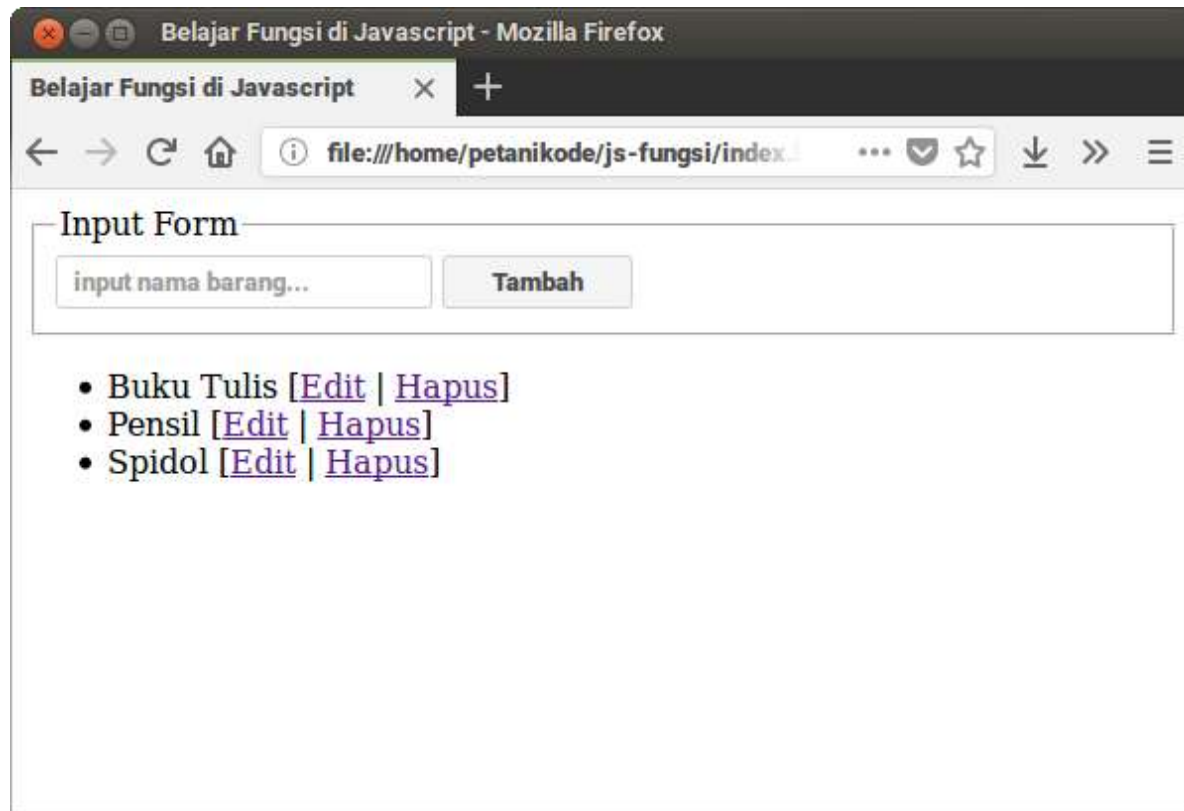
  <script src="fungsi.js"></script>
</body>
</html>
```

# Fungsi.js

```
var dataBarang = [  
  "Buku Tulis",  
  "Pensil",  
  "Spidol"  
];  
  
function showBarang(){  
  var listBarang = document.getElementById("list-barang");  
  // clear list barang  
  listBarang.innerHTML = "";  
  
  // cetak semua barang  
  for(let i = 0; i < dataBarang.length; i++){  
    var btnEdit = "<a href='#' onclick='editBarang(\"+i+\")'>Edit</a>";  
    var btnHapus = "<a href='#' onclick='deleteBarang(\"+i+\")'>Hapus</a>";  
  
    listBarang.innerHTML += "<li>" + dataBarang[i] + " [" + btnEdit + " | " + btnHapus + "]</li>";  
  }  
}  
  
function addBarang(){  
  var input = document.querySelector("input[name=barang]");  
  dataBarang.push(input.value);  
  showBarang();  
}
```

```
function editBarang(id){  
    var newBarang = prompt("Nama baru", dataBarang[id]);  
    dataBarang[id] = newBarang;  
    showBarang();  
}  
  
function deleteBarang(id){  
    dataBarang.splice(id, 1);  
    showBarang();  
}  
  
showBarang();
```

# Hasil:

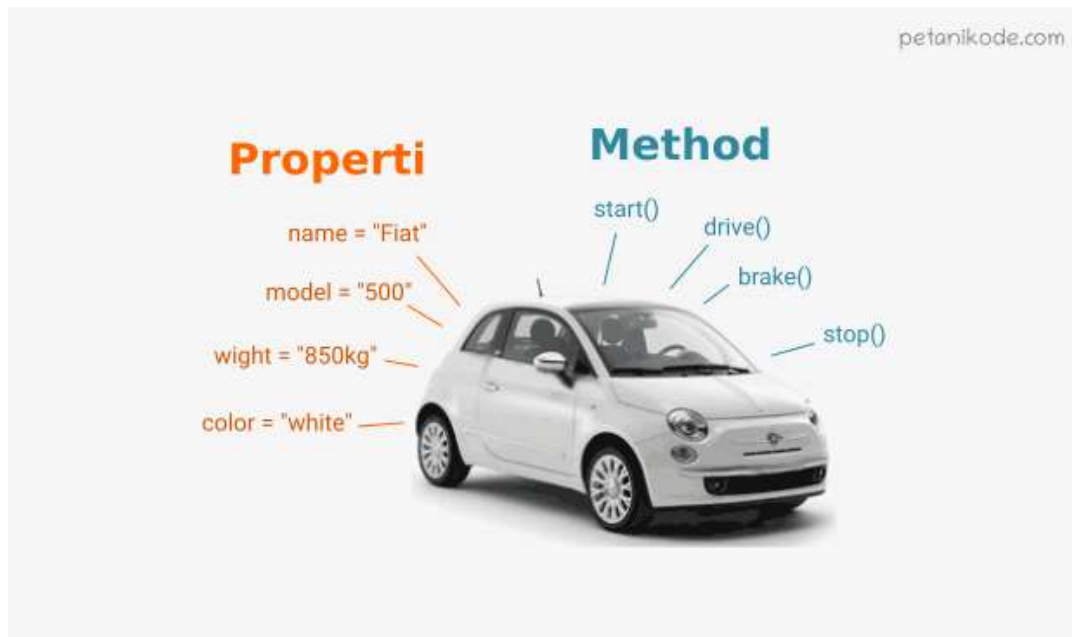


# Obyek di Javascript



# Pengenalan

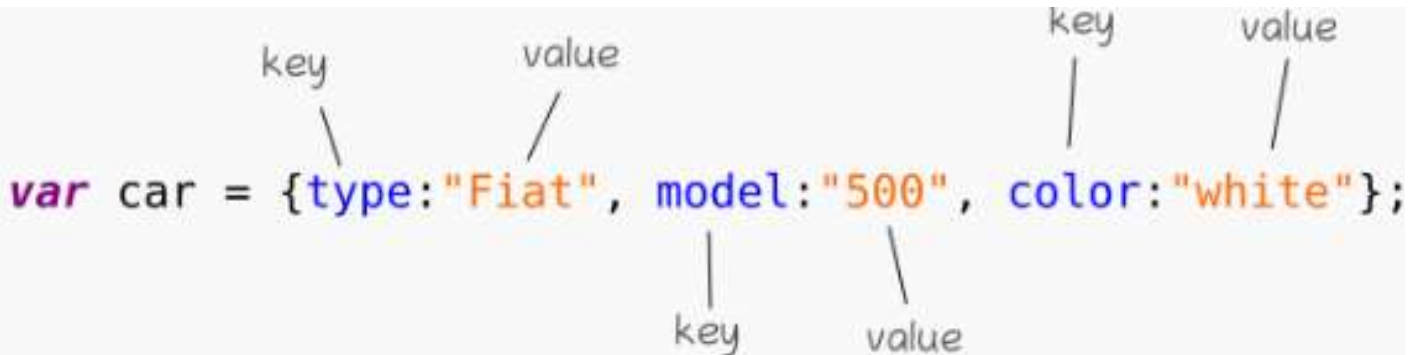
Objek sebenarnya adalah sebuah variabel yang menyimpan nilai (property) dan fungsi (*method*). Contoh objek mobil:



Mobil itu dapat dimodelkan ke dalam kode program seperti ini:

```
var car = "Mobil Fiat";
```

Tapi variabel car hanya akan menyimpan nama mobil saja. Karena itu, kita harus menggunakan objek. Objek pada javascript, dapat dibuat dengan tanda kurung kurawal dengan isi berupa **key** dan **value**.



```
var car = {type:"Fiat", model:"500", color:"white"};
```

The diagram illustrates the structure of a JavaScript object. It shows the code `var car = {type:"Fiat", model:"500", color:"white"};` with handwritten labels and arrows pointing to the key-value pairs. The first pair is `type:"Fiat"`, where `type` is labeled as the 'key' and `"Fiat"` as the 'value'. The second pair is `model:"500"`, where `model` is labeled as the 'key' and `"500"` as the 'value'. The third pair is `color:"white"`, where `color` is labeled as the 'key' and `"white"` as the 'value'.

Atau bisa ditulis seperti ini:

```
var car = {  
  type: "Fiat",  
  model: "500",  
  color: "white"  
};
```

# Perbedaan Properti dan Method

Properti adalah ciri khas dari objek (variabel). Sedangkan *method* adalah perilaku dari objek (fungsi).

Method dapat dibuat dengan cara mengisi nilai (value) dengan sebuah fungsi.

# Contoh:

```
var car = {  
  // properti  
  type: "Fiat",  
  model: "500",  
  color: "white",  
  
  // method  
  start: function(){  
    console.log("ini method start");  
  },  
  drive: function(){  
    console.log("ini method drive");  
  },  
  brake: function(){  
    console.log("ini method brake");  
  },  
  stop: function(){  
    console.log("ini method stop");  
  }  
};
```

## Cara Mengakses Properti dan Method Objek

Caranya menggunakan tanda titik atau dot (.), lalu diikuti dengan nama properti atau method.

```
console.log(car.type);
console.log(car.color);

car.start();
car.drive();
```



Untuk mengakses properti, cukup menggunakan nama objek.properti. Sedangkan method harus menggunakan tanda kurung ( ) yang menyatakan kalau kita ingin mengeksekusi fungsi.

# Menggunakan Keyword this

Kata kunci this digunakan untuk mengakses properti dan method dari dalam method (objek). Contoh:

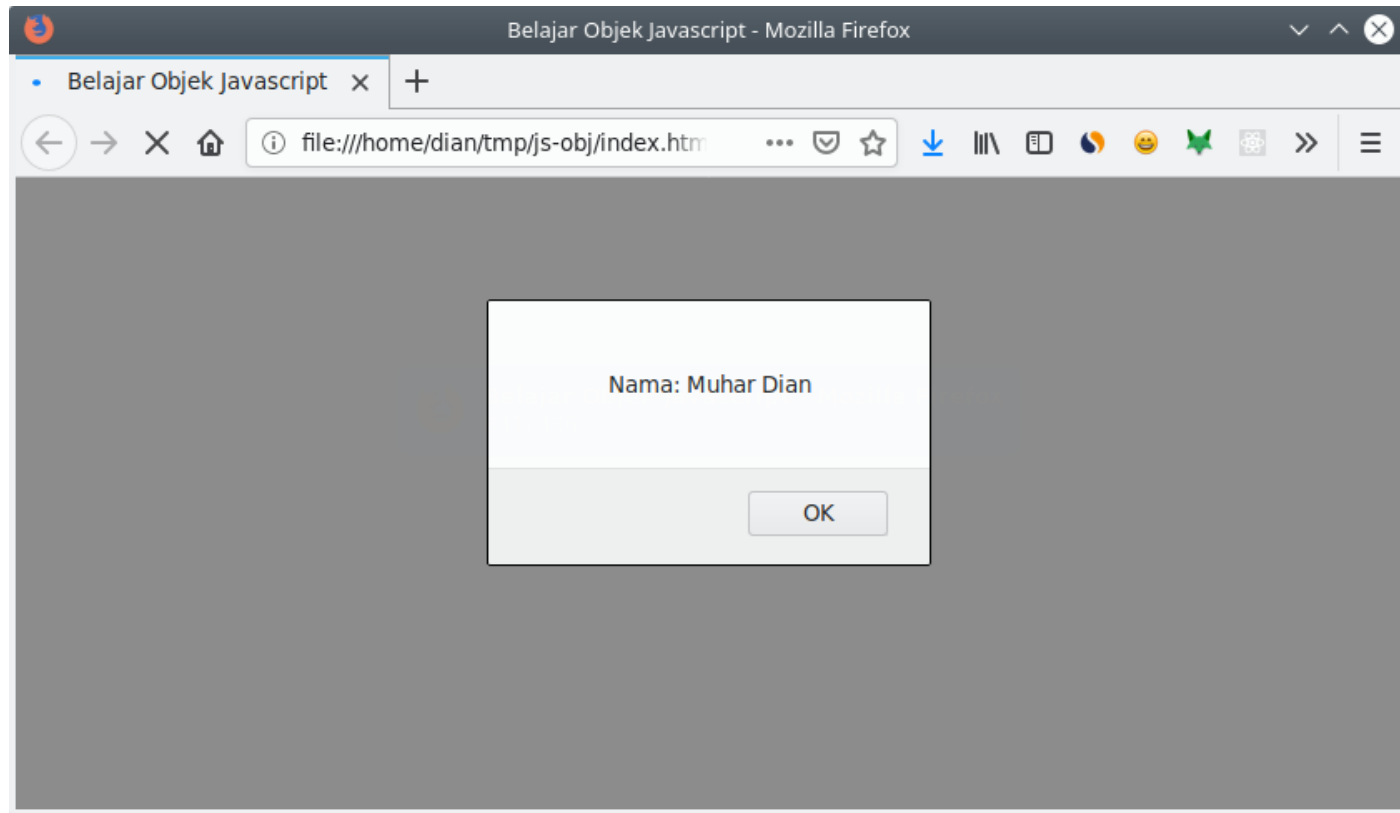
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Belajar Objek Javascript</title>

  <script>
    var person = {
      firstName: "Muhar",
      lastName: "Dian",
      showName: function(){
        alert(`Nama: ${this.firstName} ${this.lastName}`);
      }
    };

    person.showName();
  </script>
</head>
<body>
```



# Hasil:



Kata kunci **this** pada contoh diatas akan mengacu pada obyek **person**.

## Membuat Class Objek dengan **this**

Sebelumnya pembuatan obyek dilakukan dengan membuat **instance** dari **class**. Akan tetapi pada contoh sebelumnya, obyek dibuat secara langsung. Tetapi obyek lain dengan properti yang sama dapat juga dibuat dengan cara seperti ini:

```
var person = {  
  firstName: "Muhar",  
  lastName: "Dian",  
  showName: function(){  
    alert(`Nama: ${this.firstName} ${this.lastName}`);  
  }  
};
```

```
var person2 = {  
  firstName: "Petani",  
  lastName: "Kode",  
  showName: function(){  
    alert(`Nama: ${this.firstName} ${this.lastName}`);  
  }  
};
```

Ini tentu tidak efektif karena bila ada banyak obyek yang ingin dibuat kode yang sama harus ditulis ulang.

Solusinya adalah dengan menggunakan **class**.

Pada Javascript versi ES5, **class** belum ada. Fitur ini baru ditambahkan pada Javascript versi ES6. Pada ES5, kita bisa membuat **class** dengan fungsi. Lalu di dalamnya menggunakan kata kunci **this**.

## Contoh:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Belajar Objek Javascript</title>

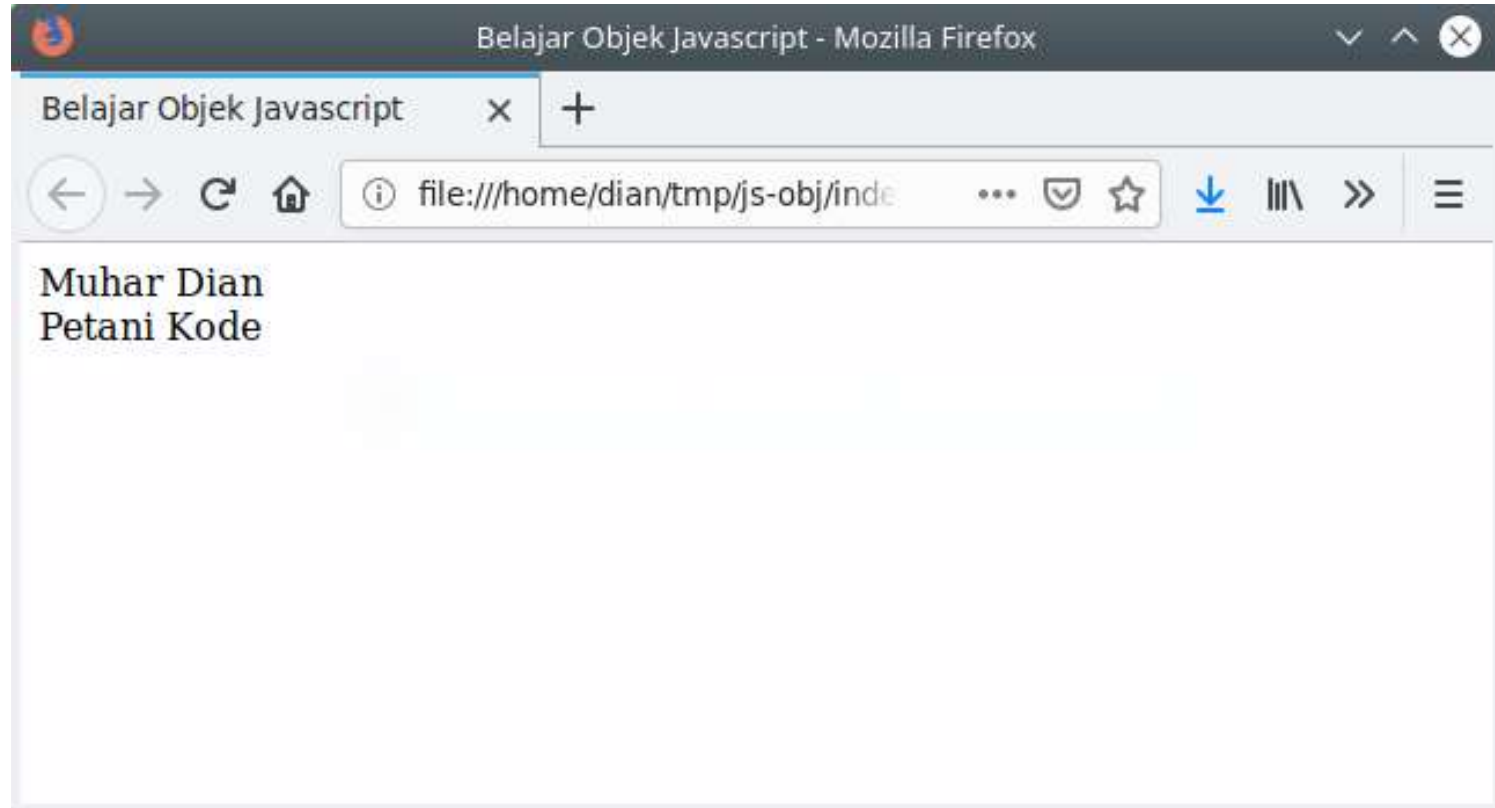
  <script>
    function Person(firstName, lastName){
      // properti
      this.firstName = firstName;
      this.lastName = lastName;

      // method
      this.fullName = function(){
        return `${this.firstName} ${this.lastName}`
      }
      this.showName = function(){
        document.write(this.fullName());
      }
    }
  </script>

```

```
var person1 = new Person("Muhar", "Dian");  
var person2 = new Person("Petani", "Kode");  
  
person1.showName();  
document.write("<br>");  
person2.showName();  
</script>  
</head>  
<body>
```

# Hasil:



Pada contoh yang tadi, kita membuat obyek dengan kata kunci **new**, lalu diberikan parameter **firstName** dan **lastName**.

```
var person1 = new Person("Muhar", "Dian");
```

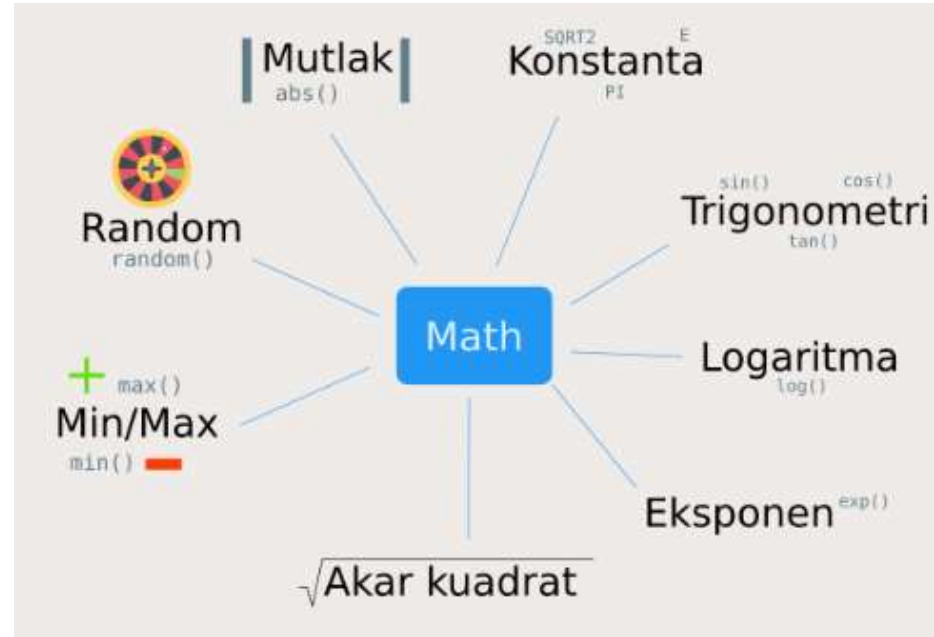
Maka dari itu, berapapun obyek yang ingin dibuat cukup gunakan kata kunci **new**.



# Obyek Math

# Pengenalan

Objek Math adalah objek yang berisi fungsi-fungsi matematika dan beberapa konstanta untuk melakukan perhitungan matematika seperti sin, cos, tan, eksponen, akar kuadrat, dll. Ini adalah fungsi-fungsi yang umum digunakan dalam perhitungan matematis.



# Fungsi Trigonometri di Javascript

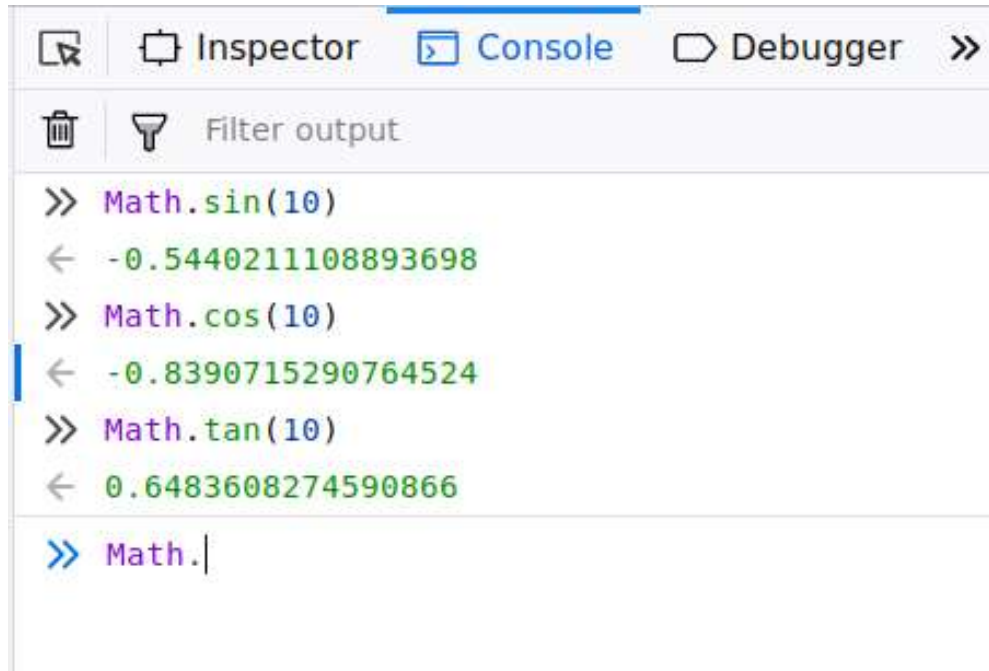
Trigonometri adalah cabang ilmu matematika yang mempelajari tentang sudut dan panjang pada segitiga.

Dalam dunia komputer, ilmu ini biasanya diimplementasikan dalam komputer grafis.

Di objek Math terdapat fungsi-fungsi untuk menghitung trigonometri. Misalkan untuk menghitung nilai sin dari 10, maka dapat ditulis seperti ini:

```
var n = Math.sin(10);
```

Variabel n akan berisi -0.5440211108893698 karena  $\sin 10$  adalah -0.5.



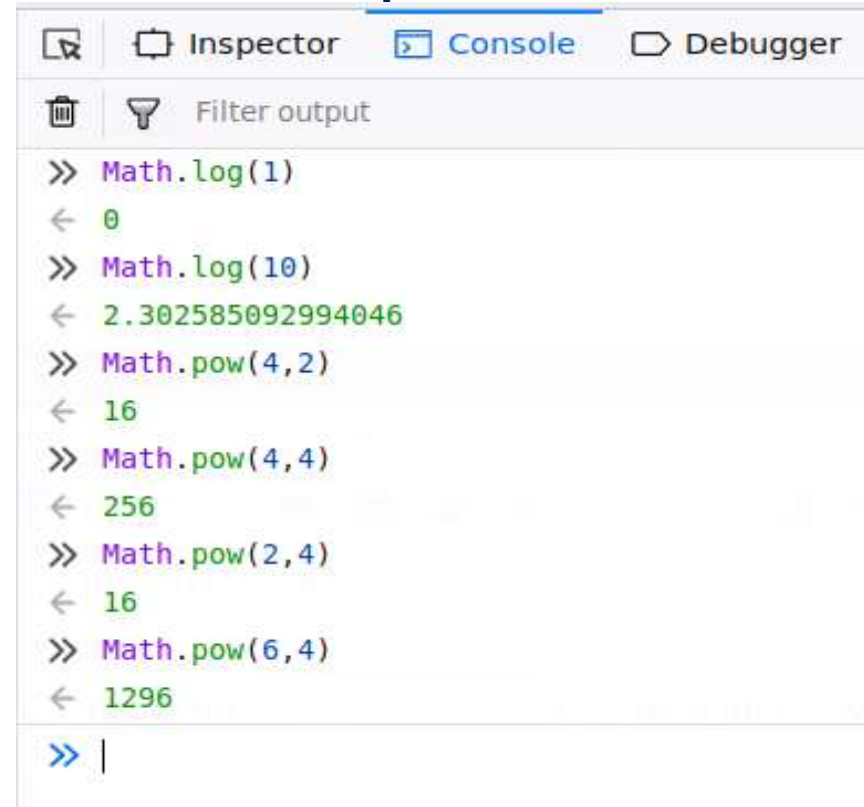
```

>> Math.sin(10)
< -0.5440211108893698
>> Math.cos(10)
< -0.8390715290764524
>> Math.tan(10)
< 0.6483608274590866
>> Math.|
  
```

## Fungsi Logaritma, Pangkat, dan Eksponensial di Javascript

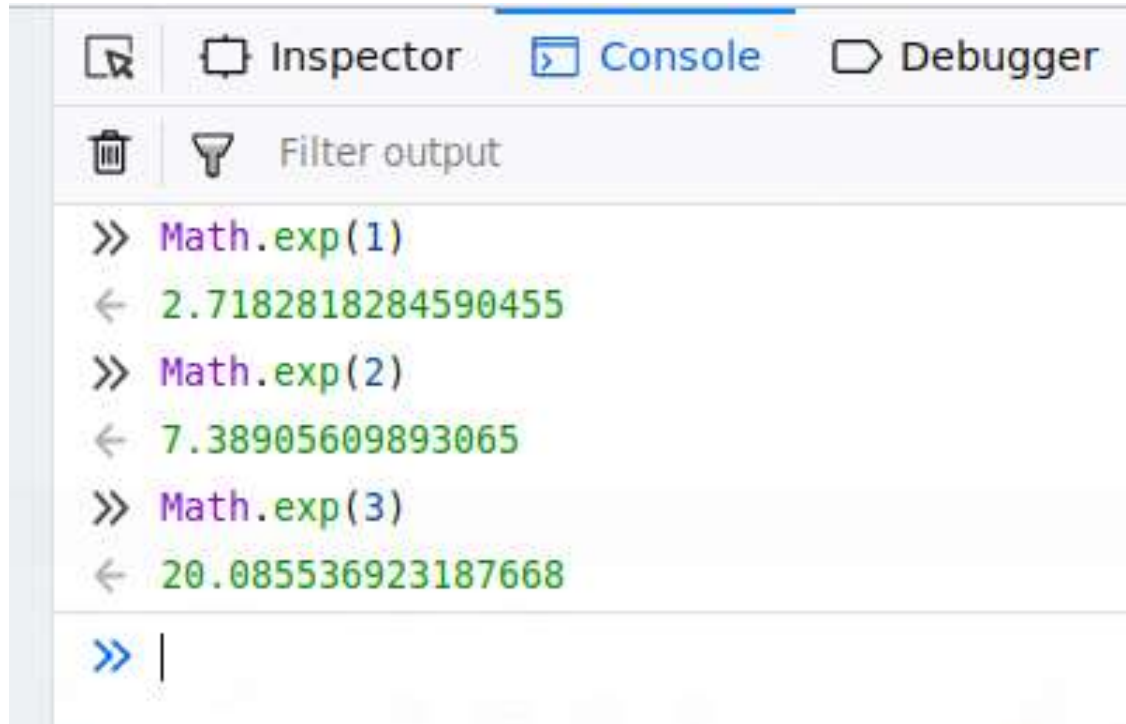
Logaritma adalah operasi matematika yang merupakan kebalikan (atau invers) dari eksponen atau pemangkatan.

Objek Math di Javascript juga menyediakan fungsi `log()` untuk logaritma dan `pow()` untuk pemangkatan.



```
>> Math.log(1)
< 0
>> Math.log(10)
< 2.302585092994046
>> Math.pow(4,2)
< 16
>> Math.pow(4,4)
< 256
>> Math.pow(2,4)
< 16
>> Math.pow(6,4)
< 1296
>> |
```

Untuk menghitung exponensial dapat digunakan `exp()`.



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays three lines of code and their corresponding output values:

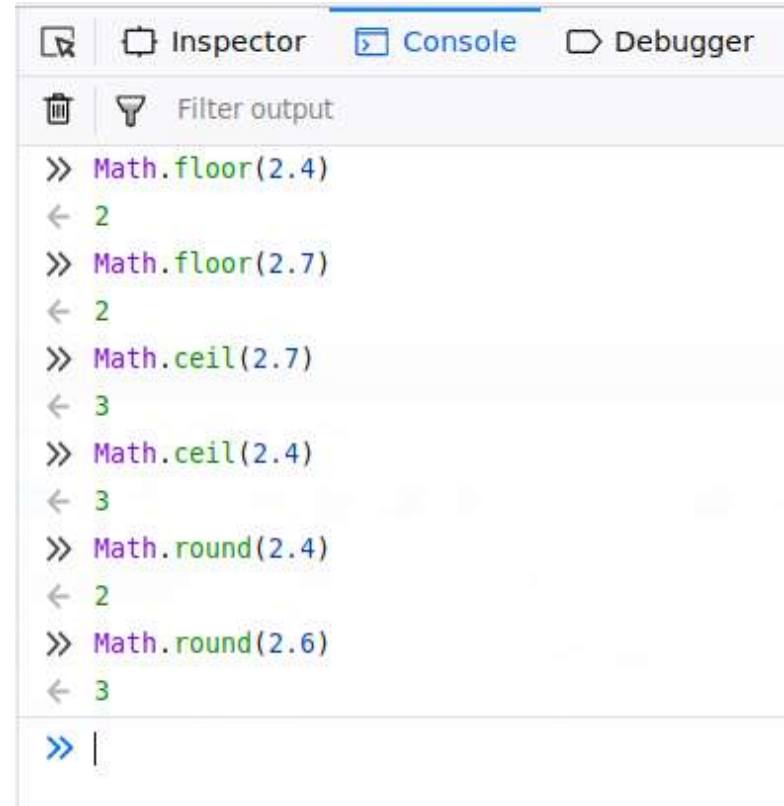
```
>> Math.exp(1)
< 2.7182818284590455
>> Math.exp(2)
< 7.38905609893065
>> Math.exp(3)
< 20.085536923187668
```

Below the last output, there is a prompt character '>>' followed by a vertical bar '|', indicating the cursor is ready for the next command.

# Fungsi Pembulatan di Javascript

Ada beberapa fungsi yang sering digunakan:

1. **floor()** membulatkan ke bawah;
2. **round()** membulatkan ke yang terdekat, bisa ke bawah dan ke atas;
3. **ceil()** membulatkan ke atas.

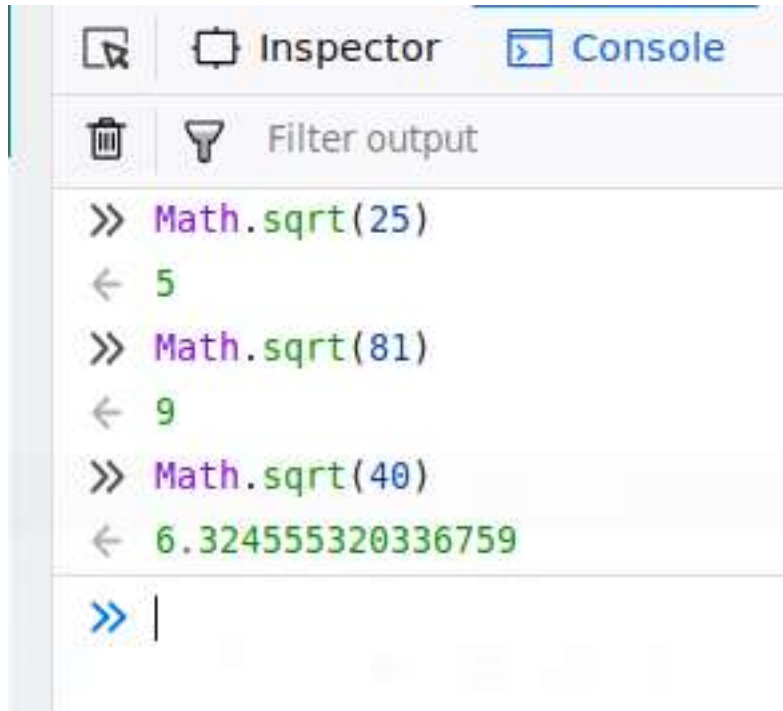
A screenshot of a web browser's developer console. The 'Console' tab is selected, showing a list of log entries. Each entry consists of a prompt character '»' followed by a function call, and a return character '←' followed by the result. The functions shown are Math.floor(2.4), Math.floor(2.7), Math.ceil(2.7), Math.ceil(2.4), Math.round(2.4), and Math.round(2.6). The results are 2, 2, 3, 3, 2, and 3 respectively. At the bottom, there is a prompt character '»' followed by a vertical bar '|', indicating the console is ready for input.

```
» Math.floor(2.4)
← 2
» Math.floor(2.7)
← 2
» Math.ceil(2.7)
← 3
» Math.ceil(2.4)
← 3
» Math.round(2.4)
← 2
» Math.round(2.6)
← 3
» |
```

# Fungsi Akar di Javascript

Fungsi akar menggunakan fungsi **sqrt()**

Sementara kubik menggunakan **cbrt()**



```
» Math.sqrt(25)
← 5
» Math.sqrt(81)
← 9
» Math.sqrt(40)
← 6.324555320336759
» |
```

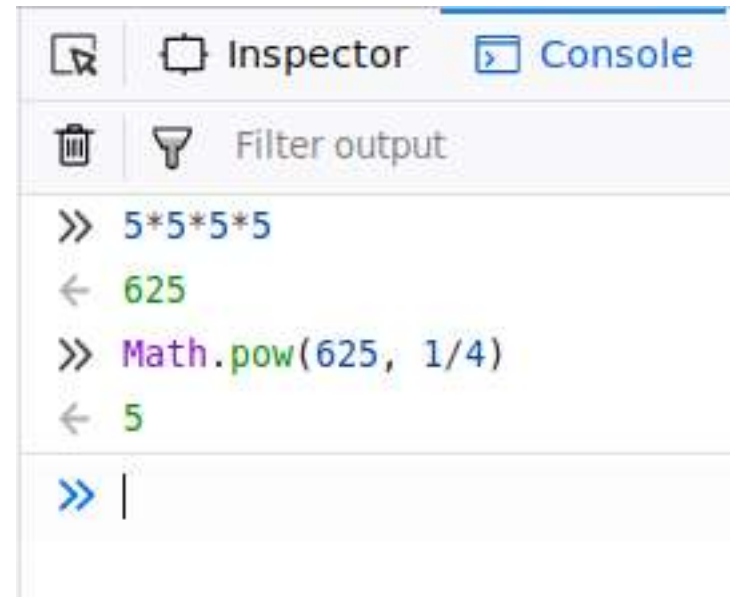


```
» Math.cbrt(125)
← 5
» |
```



untuk akar **n** atau **nth** root, dapat dihitung menggunakan fungsi **pow()** yang di akali. Contoh:

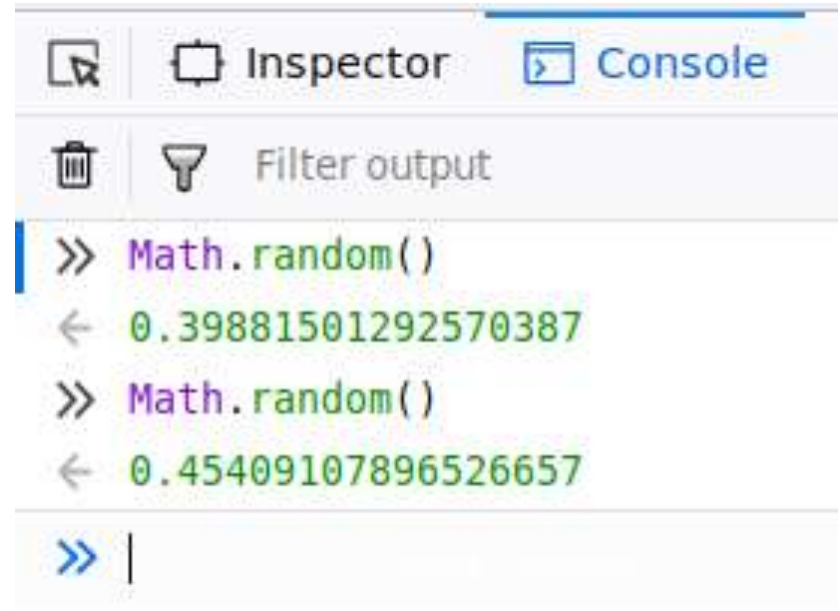
```
Math.pow(n, 1/root);
```



**n** adalah nilai yang dicari, **root** adalah akar dari **n**.

## Fungsi Random dan Mutlak di Javascript

Adalah fungsi yang menghasilkan nilai acak antara 0.0 sampai 1.0.



```

>> Math.random()
< 0.39881501292570387
>> Math.random()
< 0.45409107896526657
>> |
  
```

Jika ingin membuat nilai acak dari rentang nilai tertentu, maka kita bisa menggunakan bantuan fungsi `floor()` untuk membulatkan lalu dikali dengan nilai min dan max.

Rumusnya akan seperti ini:

```
Math.floor(Math.random() * (max - min) ) + min;
```

Yang dapat di bentuk menjadi fungsi:

```
function getRndInteger(min, max) {  
  return Math.floor(Math.random() * (max - min) ) + min;  
}
```

# Hasil:

```
>> function getRndInteger(min, max) {  
    return Math.floor(Math.random() * (max - min) ) + min;  
}  
← undefined  
>> getRndInteger(1, 100)  
← 39  
>> |
```

# Fungsi Mutlak

adalah fungsi yang menghasilkan nilai mutlak atau *absolute*. Contoh:

```
var x = Math.abs(-2)
```

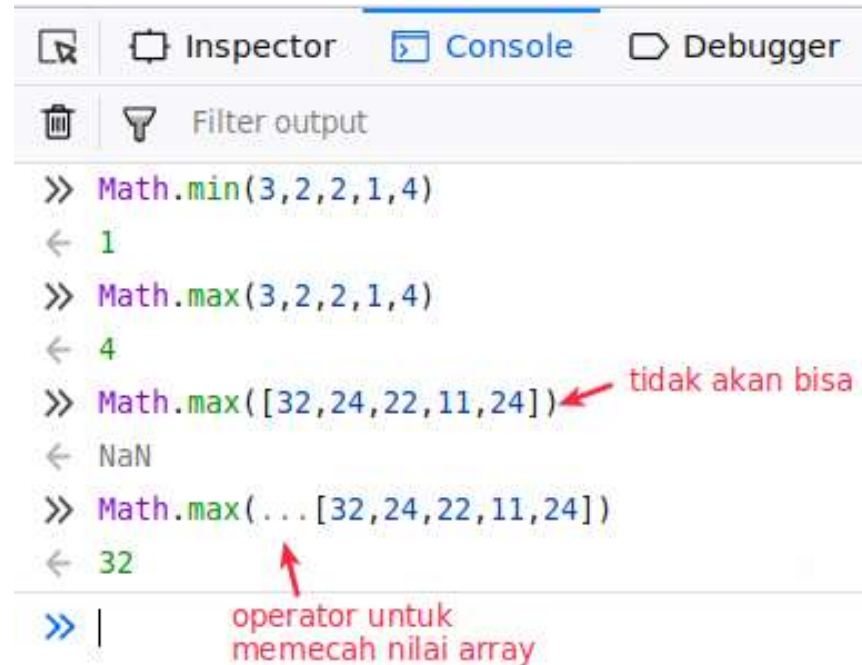
Variabel **x** akan bernilai **2**, karena fungsi **abs()** akan selalu memberikan nilai mutlak atau positif.

## Fungsi Minimum dan Maksimum di Javascript

Fungsi minimum dan maksimum adalah fungsi untuk menentukan nilai paling kecil dan paling besar pada sekumpulan nilai.

Fungsi ini bisa kita berikan input berupa urutan bilangan.

Apabila kita ingin memberikan input array, maka array tersebut harus kita pecah isinya.



```
>> Math.min(3,2,2,1,4)
< 1
>> Math.max(3,2,2,1,4)
< 4
>> Math.max([32,24,22,11,24])
< NaN
>> Math.max(...[32,24,22,11,24])
< 32
>> |
```

tidak akan bisa

operator untuk memecah nilai array

# Konstanta di Objek Math

Selain menyediakan fungsi-fungsi matematika, objek Math juga menyediakan konstanta seperti **PI**, **E**, **LN10**, dll. yang bisa kita manfaatkan untuk perhitungan rumus tertentu.

```
Math.E           // returns Euler's number
Math.PI          // returns PI
Math.SQRT2       // returns the square root of 2
Math.SQRT1_2     // returns the square root of 1/2
Math.LN2         // returns the natural logarithm of 2
Math.LN10        // returns the natural logarithm of 10
Math.LOG2E       // returns base 2 logarithm of E
Math.LOG10E      // returns base 10 logarithm of E
```

# AJAX Menggunakan JQuery

JQuery adalah library Javascript yang menyederhanakan fungsi-fungsi Javascript. Pada JQuery, AJAX dapat dibuat seperti ini:

```
// load data ke elemen tertentu via AJAX
$(selector).load(URL,data,callback);

// ambil data dari server
$.get(URL,callback);

// kirim data dari Server
$.post(URL,data,callback);
```

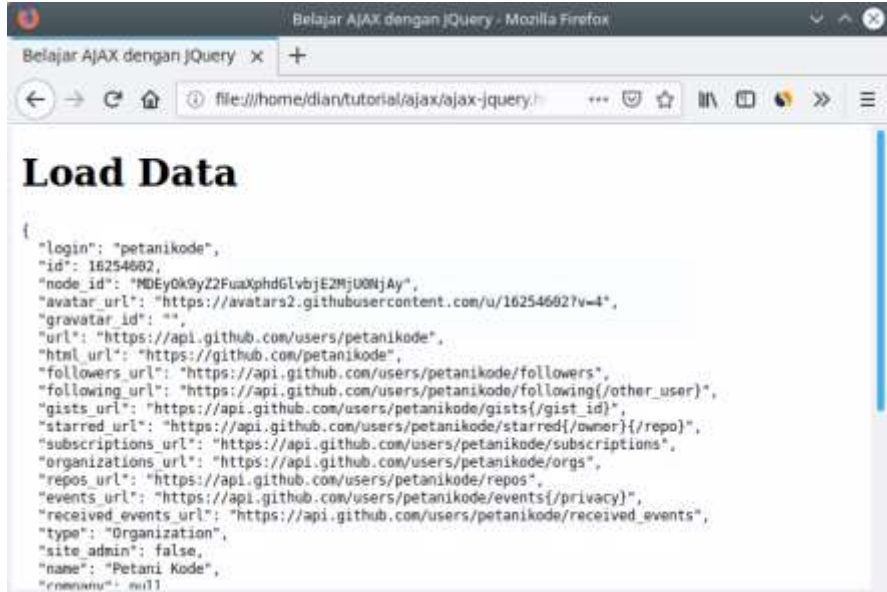


## Buatlah file ajax-jquery.html dengan isi:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Belajar AJAX dengan JQuery</title>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>
</head>
<body>
  <h1>Load Data</h1>
  <pre id="result"></pre>

  <script>
    $("#result").load("https://api.github.com/users/petanikode");
  </script>
</body>
</html>
```

# Hasil:



The screenshot shows a web browser window titled "Belajar AJAX dengan JQuery - Mozilla Firefox". The address bar shows the file path "file:///home/dian/tutorial/ajax/ajax-jquery...". The page content displays the title "Load Data" followed by a JSON object representing a GitHub user profile for "petanikode".

```
{
  "login": "petanikode",
  "id": 16254662,
  "node_id": "MDEyOk9yZ2FuaXphdGlvbWJlZ2MjU0RjAy",
  "avatar_url": "https://avatars2.githubusercontent.com/u/16254662?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/petanikode",
  "html_url": "https://github.com/petanikode",
  "followers_url": "https://api.github.com/users/petanikode/followers",
  "following_url": "https://api.github.com/users/petanikode/following{/other_user}",
  "gists_url": "https://api.github.com/users/petanikode/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/petanikode/starred{/owner}/{repo}",
  "subscriptions_url": "https://api.github.com/users/petanikode/subscriptions",
  "organizations_url": "https://api.github.com/users/petanikode/orgs",
  "repos_url": "https://api.github.com/users/petanikode/repos",
  "events_url": "https://api.github.com/users/petanikode/events{/privacy}",
  "received_events_url": "https://api.github.com/users/petanikode/received_events",
  "type": "Organization",
  "site_admin": false,
  "name": "Petani Kode",
  "email": null
}
```

Dengan fungsi `$("#result").load()`, kita bisa mengambil data dengan AJAX dan langsung menampilkannya pada elemen yang dipilih.

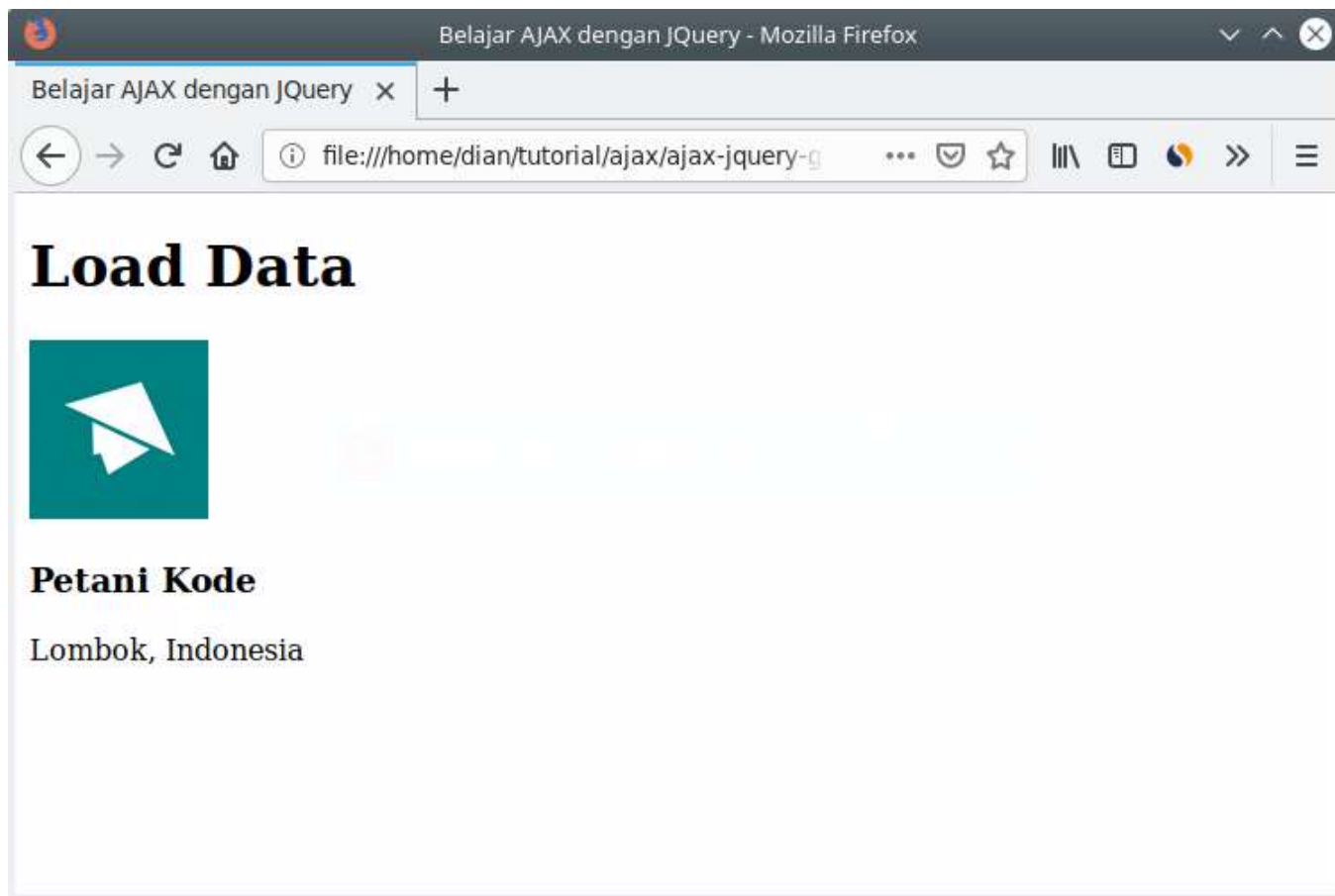
Fungsi JQuery `load()` cocoknya untuk mengambil bagian dari HTML untuk ditampilkan.

## Contoh lainnya menggunakan metode GET:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Belajar AJAX dengan JQuery</title>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>
</head>
<body>
  <h1>Load Data</h1>
  <img id="avatar" src="" width="100" height="100">
  <br>
  <h3 id="name"></h3>
  <p id="location"></p>

  <script>
    var url = "https://api.github.com/users/petaniicode";
    $.get(url, function(data, status){
      $("#avatar").attr("src", data.avatar_url);
      $("#name").text(data.name);
      $("#location").text(data.location);
    });
  </script>
</body>
</html>
```

# Hasil:



Untuk mengirim data dengan AJAX di JQuery, caranya hampir sama seperti mengambil data dengan **\$.get()**.

```
<script>
var url = "https://jsonplaceholder.typicode.com/posts";
var data = {
  title: "Tutorial AJAX dengan JQuery",
  body: "Ini adalah tutorial tentang AJAX",
  userId: 1
};
$.post(url, data, function(data, status){
  // data terkirim, lakukan sesuatu di sini
});
</script>
```