

MINGGU 11 (PEMROGRAMAN BERBASIS WEB)

Javascript (BASIC session-2)

Pada pertemuan 11 ini melanjutkan praktikum javascript dasar, akan dibahas mengenai percabangan, perulangan dan struktur data Array.

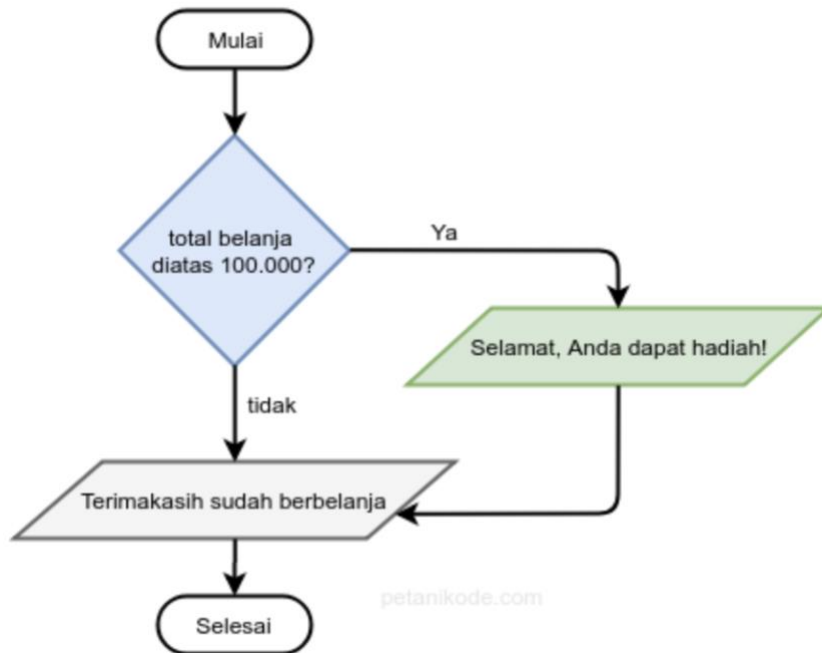
Percabangan pada Javascript.

Percabangan akan mampu membuat program berpikir dan menentukan tindakan sesuai dengan logika/kondisi yang diberikan, Pada pemrograman Javascript kurang lebih ada **6 bentuk percabangan** yang **harus** diketahui dan dikuasai.

- Percabangan If
- Percabangan if/else
- Percabangan if/else/if
- Percabangan switch/case
- Percabangan dengan operator ternary
- Percabangan bersarang (Nested)

Percabangan If

Percabangan *if* merupakan percabangan yang hanya memiliki **satu blok pilihan** saat kondisi bernilai benar, sebagai contoh :



Flowchart tersebut kurang lebih sebagai berikut:

“Jika total belanja lebih besar dari Rp 100.000, Maka tampilkan pesan **Selamat, Anda dapat hadiah**”, kalau dibawah Rp 100.000 ? maka pesan tidak ditampilkan.

Praktikum 1 : buat file **cabang-if.html**

```
<!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>
```

Perhatikan bagian:

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

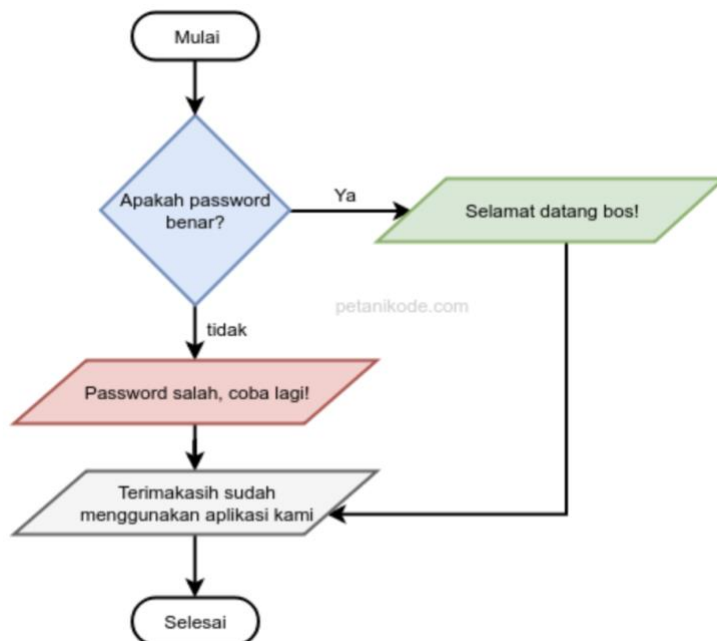
Ini yang disebut blok, blok program pada Javascript, diawali dengan tanda buka kurung kurawal { dan diakhiri dengan tutup kurung kurawal }.

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>");
```

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*).



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 bukan bagian dari blok percabangan *if/else*.

Praktikum 2 : buat file **cabang-if-else.html**

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

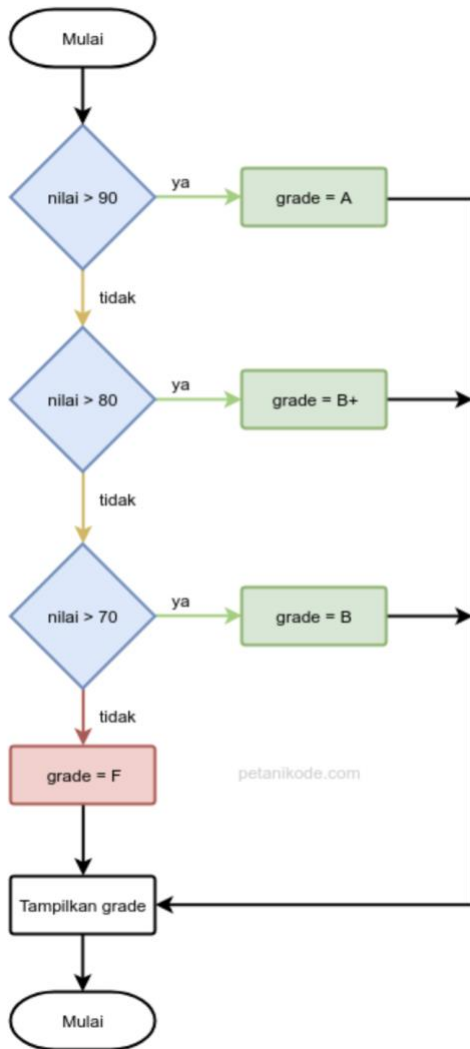
    if(password == "kopi"){
      document.write("<h2>Selamat datang bos!</h2>");
    } else {
      document.write("<p>Password salah, coba lagi!</p>");
    }

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

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

Percabangan If/else/if

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



Blok untuk percabangan *if/else/if* bisa menambahkan berapapun blok yang diinginkan.

Praktikum 3 : buat file **cabang-if-else-if.html**

```
<!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>

```

Pada blok program di atas, tidak menggunakan kurung kurawal untuk membuat blok kode untuk *if/else/if*, karena hanya terdapat satu baris perintah saja. Yaitu: **grade =**, bila akan menggunakan kurung kurawal, maka program di atas akan menjadi seperti ini dibawah ini :

```

<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>

```

Percabangan switch/case

Percabangan *switch/case* adalah bentuk lain dari percabangan *if/else/if*, strukturnya sebagai berikut:

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

Dapat dibuat blok kode (*case*) sebanyak yang diinginkan di dalam blok *switch*, pada **<value>**, bisa isi dengan nilai yang nanti akan dibandingkan dengan variabel, setiap **case** harus diakhiri dengan **break**. Khusus untuk **default**, tidak perlu diakhiri dengan **break** karena terletak di bagian akhir, pemberian **break** bertujuan agar program berhenti mengecek **case** berikutnya saat sebuah **case** terpenuhi.

Praktikum 4 : buat file **cabang-switch-case.html**

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <title>Percabangan switch/case</title>  
</head>  
<body>  
    <script>  
  
        var jawab = prompt("Kamu beruntung! Silahkan 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>Oops! 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>

```

Percabangan **switch/case** juga dapat dibuat seperti dibawah 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-tama, diberikan nilai true pada **switch**, ini agar bisa masuk ke dalam blok **switch**, Lalu didalam blok **switch**, dibuat kondisi dengan menggunakan **case**, hasilnya akan sama seperti pada contoh **percabangan if/else/if**.

Percabangan dengan operator Ternary

Percabangan menggunakan operator *ternary* merupakan bentuk lain dari percabangan *if/else*, atau lebih tepatnya : **Bentuk singkatnya dari *if/else*.**

Praktikum 5 : buat file **cabang-ternary.html**

```
<!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 dari method ***toUpperCase()*** untuk mengubah teks yang diinputkan menjadi huruf kapital semua.

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".

Percabangan Bersarang (Nested)

Membuat percabangan di dalam percabangan. Ini disebut percabangan bersarang atau ***nested if***.

Praktikum 6 : buat file **cabang-nested.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```

<title>Percabangan Ternary</title>
</head>
<body>
  <script>
    var username = prompt("Username:");
    var password = prompt("Password:");

    if(username == "belajarkode"){
      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>

```

TIPS - Menggunakan Operator Logika pada Percabangan

Percabangan bersarang, bisa dibuat lebih sederhana lagi dengan menggunakan operator logika.

Contoh sebelum menggunakan operator logika :

```

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

if(username == "belajarkode"){
  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>");
}

```

Penyederhanaan dengan operator logika AND (&&):

```

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

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

```

Perulangan pada Javascript.

Perulangan akan membantu mengeksekusi kode yang berulang-ulang, berapapun yang diinginkan, ada lima macam bentuk perulangan di Javascript. Secara umum, perulangan ini dibagi dua.

Yaitu: **counted loop** dan **uncounted loop**.

- **Counted Loop** merupakan perulangan yang jelas dan sudah tentu banyak perulangannya.
- Sedangkan **Uncounted Loop**, merupakan perulangan yang tidak jelas berapa kali proses perulangannya.

Perulangan **Counted Loop**:

1. Perulangan **For**
2. Perulangan **Foreach**
3. Perulangan **Repeat**

Perulangan **Uncounted Loop**:

1. Perulangan **While**
2. Perulangan **Do/While**

Perulangan For

Syntax sebagai berikut:

```
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 akan dimulai dari **0 (i = 0)**;

- Hitungannya sampai berapa? Sampai **$i < 10$** ;
- Lalu di setiap perulangan i akan bertambah **$+1$ ($i++$)**.

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

Praktikum 7 : buat file **perulangan-for.html**

- Buatkan perulangan dari nilai 10 ke nilai 1 (angka mundur)

Perulangan While

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

Praktikum 8 : buat file **perulangan-while.html**

Sisipkan pada file html kode javascript dibawah ini:

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

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

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

atau bisa menggunakan penyederhanaan script dibawah ini:

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

while(ulang){
    counter++;
    ulang = confirm("Apakah anda mau mengulang?");
}

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

Coba perhatikan blok kode **while**:

```
while(ulangi){
    counter++;
    ulangi = confirm("Apakah anda mau mengulang?");
}
```

Perulangan akan terjadi selama variabel `ulangi` bernilai `true`, kemudian digunakan fungsi **`confirm()`** untuk menampilkan dialog konfirmasi, selama memilih **Ok** pada dialog konfirmasi, maka variabel `ulangi` akan terus bernilai `true`, dan jika dipilih **Cancel**, maka variabel `ulangi` akan bernilai **`false`**, saat variabel `ulangi` bernilai **`false`**, maka perulangan akan dihentikan.

Perulangan Do While

Perulangan **`do/while`** sama seperti perulangan **`while`**.

Perbedaannya:

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

Syntax:

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

Perbedaannya:

Perulangan **`do/while`** akan mengecek kondisi di belakang (sesudah mengulang), sedangkan **`while`** akan mengecek kondisi di depan atau awal (sebelum mengulang).

Praktikum 9 : buat file **`perulangan-do-while.html`**

Sisipkan kode javascript dibawah ini:

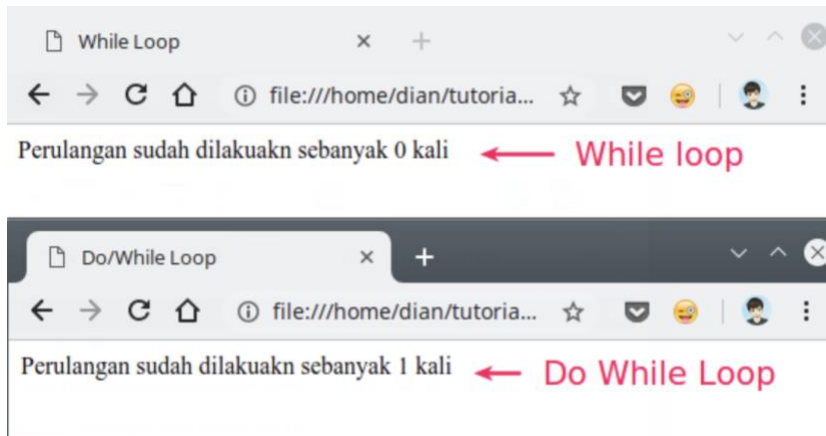
```
var ulangi = confirm("Apakah anda mau mengulang?");
var counter = 0;

do {
    counter++;
    ulangi = confirm("Apakah anda mau mengulang?");
} while(ulangi)
```

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

Contoh tersebut sama seperti contoh pada perulangan while.

Saat perulangan pertama dicoba untuk membatalkan perulangannya dengan memilih **Cancel**.



Perulangan Foreach

Perulangan foreach biasa 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().

Praktikum 10 : buat file **perulangan-foreach-1.html**

```
var languages = ["Javascript", "HTML", "CSS", "Typescript"];  
for(i = 0; i < languages.length; i++){  
    document.write(i+" ". + languages[i] + "<br/>");  
}
```

Atau bisa disederhanakan dengan operator **in** seperti berikut ini:

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

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

Praktikum 11 : buat file **perulangan-foreach-2.html**

```
// 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*. Bisa juga menggunakan **arrow function** seperti berikut 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>");
});
```

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 disimpulkan :

Ini merupakan proses **singkat** dari perulangan **for**.

Apabila menggunakan perulangan **for**:

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

Praktikum 12 : buat file **perulangan-repeat.html**

menggunakan fungsi **repeat()**:

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

Perulangan Bersarang (Nested)

Di dalam blok perulangan, dapat membuat perulangan, ini disebut dengan **nested loop** atau perulangan bersarang atau perulangan di dalam perulangan.

Praktikum 13 : buat file **perulangan-nested-1.html**

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

Praktikum 14 : buat file **perulangan-nested-2.html**

```
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?");  
}
```

Struktur Data Array pada Javascript.

Struktur data merupakan cara-cara atau metode yang digunakan untuk menyimpan data di dalam memori computer, salah satu struktur data yang sering digunakan dalam pemrograman adalah **Array**.

Array merupakan struktur data yang digunakan untuk **menyimpan sekumpulan data** dalam satu tempat, setiap data dalam Array memiliki indeks, sehingga akan mudah untuk memprosesnya.

"Hardisk 2TB"	"Flashdisk 32GB"	"Modem"
0	1	2

Indeks array selalu dimulai dari angka **nol (0)**, ukuran array akan bergantung dari banyaknya data yang ditampung di dalamnya.

Pembuatan Array pada Javascript

Pada javascript, **array** dapat dibuat dengan tanda kurung **siku ([...])**.

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

Maka variabel products akan berisi sebuah array kosong.

Bisa diisi data ke dalam array, kemudian setiap data **dipisah** dengan tanda **koma (,)**.

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

dikarenakan javascript merupakan bahasa pemrograman *dynamic typing*, maka bisa dilakukan proses menyimpan dan mencampur apapun di dalam array, contoh:

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

Pengambilan Data Array

Array akan menyimpan sekumpulan data dan memberinya nomer indeks agar mudah diakses, indeks array selalu dimauli dari nol 0.

Misalkan ada array seperti ini:

```
var makanan = ["Nasi Goreng", "Mie Ayam", "Mie Gelas"];
```

Bagaimana cara mengambil nilai "Mie Ayam"?

Jawabannya adalah berikut ini:

```
makanan[1] // -> "Mie Ayam"
```

Praktikum 15 : buat file **array-data.html**

```
<!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>

```

Mencetak Array dengan Perulangan

Untuk variable array yang memiliki nilai indeks banyak, tidak dimungkinkan untuk dicetak satu per satu secara manual, alternatifnya bisa menggunakan perulangan.

Praktikum 16 : buat file **array-perulangan.html**

```

<!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>

```

Pada contoh di atas, menggunakan properti length untuk mengambil panjang array, contoh diatas ada 4 data di dalam array products, maka properti length akan bernilai 4.

Alternatif berikutnya dapat menggunakan method **foreach()** sebagai berikut:

```
<!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 kedalam Array

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

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

Contoh untuk penggunaan indeks, misal ada variabel array dengan isi sebagai berikut:

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

Terdapat tiga data di dalam array buah dengan indeks:

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

Kemudian ingin ditambahkan data lagi pada indeks ke-3, maka bisa dengan seperti ini:

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

Maka sekarang array buah akan berisi 4 data.

Kekurangan dari cara ini adalah **harus tahu** dulu jumlah data atau panjang array-nya, barulah bisa menambahkan, karena jika memasukan nomer indeks tidak urut, maka yang akan terjadi data yang ada di dalam indeks akan tertindih.

Cara lain bisa digunakan method **push()**, tidak perlu tahu berapa panjang **array-nya**, karena method push() akan menambahkan data ke dalam array **dari** belakang.

Praktikum 17 : buat file **array-tambah-data.html**

```
<!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>
```

Bisa menambahkan beberapa data sekaligus dengan cara sebagai berikut:

```
products.push("Alarm", "Gembok", "Paku");
```

Menghapus Array

Menghapus data pada array bisa menggunakan dua cara:

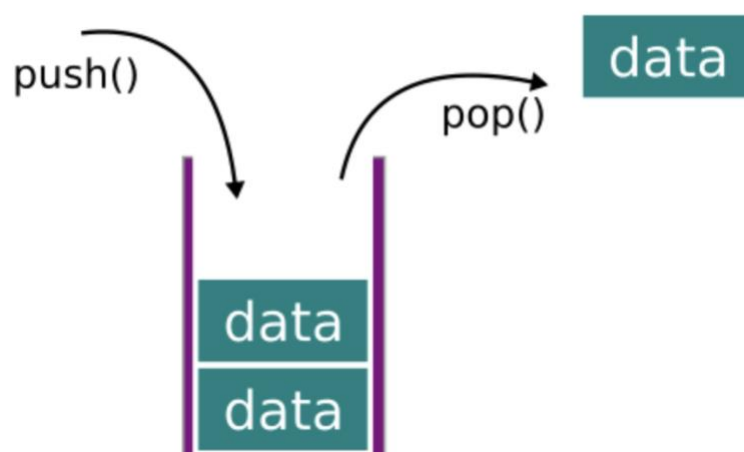
1. Menggunakan **delete**;
2. Menggunakan **method pop()**.

Menghapus data dengan nomer indeks tertentu bisa menggunakan **delete**. Sedangkan **pop()** akan menghapus dari belakang, kekurangan dari **delete**, akan menciptakan **ruang kosong** di dalam **array**.

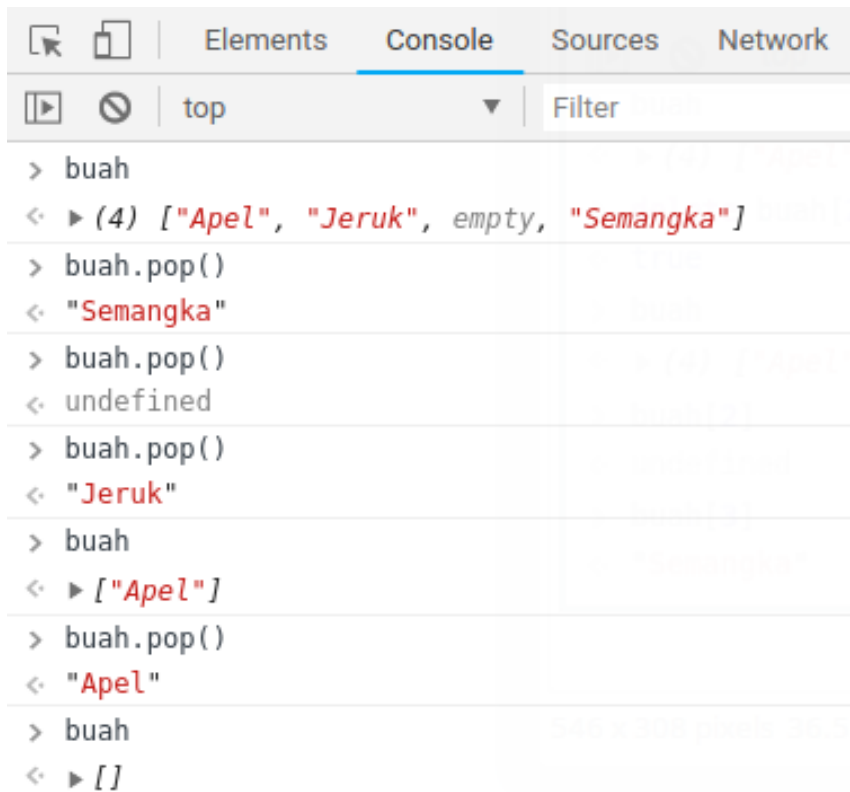
Percobaan di dalam *console*:

```
Elements Console Sources Network Performance
top Filter Default levels
> buah
< ▶ (4) ["Apel", "Jeruk", "Manggis", "Semangka"]
> delete buah[2]
< true
> buah
< ▶ (4) ["Apel", "Jeruk", empty, "Semangka"]
> buah[2]
< undefined
> buah[3]
< "Semangka"
```

Menggunakan method **pop()** adalah kebalikan dari method **push()**, method **pop()** akan menghapus array yang ada di paling belakang, array pada javascript dapat di posisikan sebagai **stack (tumpukan)**, yang memiliki sifat **LIFO (Last in Last out)**.



Percobaan pada console :



Menghapus data dari depan

Menghapus data dari depan dengan menggunakan method **shift()**.

Praktikum 18 : buat file **array-hapus-depan.html**

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

Menghapus data pada indeks tertentu

Jika 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>);
```

- **<indeks>** adalah indeks dari data di dalam array yang akan dihapus;

- **<total>** adalah jumlah data yang akan dihapus dari indeks tersebut.

Praktikum 19 : buat file **array-hapus-tertentu.html**

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

Eksperimen melalui console



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

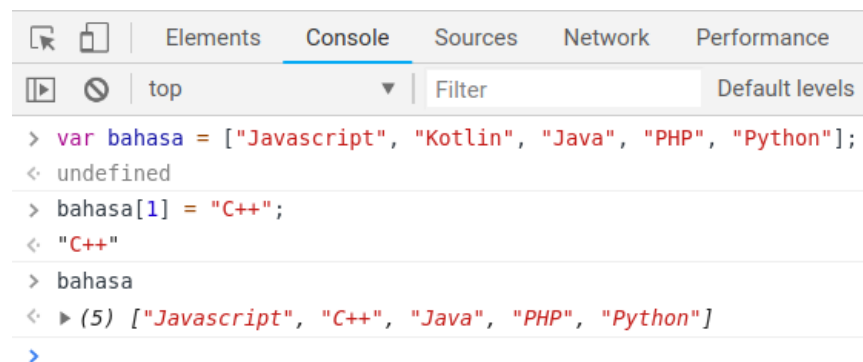
Mengubah isi Array

Untuk mengubah isi array, bisa dengan mengisi ulang seperti berikut:

```
var bahasa = ["Javascript", "Kotlin", "Java", "PHP", "Python"];  
bahasa[1] = "C++";
```

Maka "Kotlin" akan diganti dengan "C++".

Percobaan pada *console*:



```
> var bahasa = ["Javascript", "Kotlin", "Java", "PHP", "Python"];  
< undefined  
> bahasa[1] = "C++";  
< "C++"  
> bahasa  
< ▶ (5) ["Javascript", "C++", "Java", "PHP", "Python"]  
>
```

Contoh method-method array

- Method filter() : Menyaring data dari array
- Method includes() : Mengecek apakah data ada dalam array
- Method sort() : Mengurutkan data array

Praktikum 20 : buat file **array-contoh-method.html**

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

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

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

// Method Includes
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

//Method Short
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]
```

Repositori Pertemuan 11

Push hasil latihan ke Github (repository webdas) dan kirim urlnya melalui kulino pada blok (Repositori Pertemuan 11).

Untuk susunan folder dan file sebagai berikut

- repominggu11 (folder utama)
 - latihanJavascript2
 - cabang-if.html
 - dan seterusnya sesuai susunan folder Latihan diatas.