

MINGGU 12 (PEMROGRAMAN BERBASIS WEB)

Javascript (BASIC session-3)

Pada pertemuan 12 ini melanjutkan praktikum javascript session-2, akan dibahas mengenai DOM, Fungsi dan Mengenal Objek pada javascript.

DOM API

DOM merupakan singkatan dari *Document Object Model*, artinya, dokumen (HTML) yang dimodelkan dalam sebuah objek.

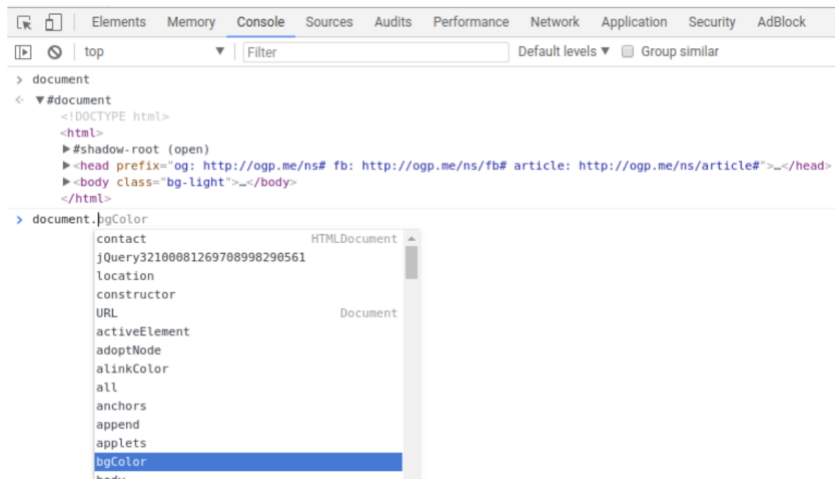
Objek dari dokumen ini menyediakan sekumpulan fungsi dan atribut/data yang bisa dimanfaatkan dalam membuat program Javascript. Inilah yang disebut API (*Application Programming Interface*).

DOM tidak hanya untuk dokumen HTML saja. DOM juga bisa digunakan untuk dokumen XML dan SVG, dan DOM juga tidak hanya ada di Javascript saja, DOM juga ada pada bahasa pemrograman lain.

Implementasi DOM

Seperti pada praktikum javascript sebelumnya, dan bahwa DOM adalah sebuah objek untuk memodelkan dokumen HTML, Objek DOM di javascript bernama document. Objek ini berisi segala hal yang dibutuhkan untuk memanipulasi HTML.

Jika coba menuliskan `document` pada *console* Javascript, maka yang akan tampil adalah kode HTML.



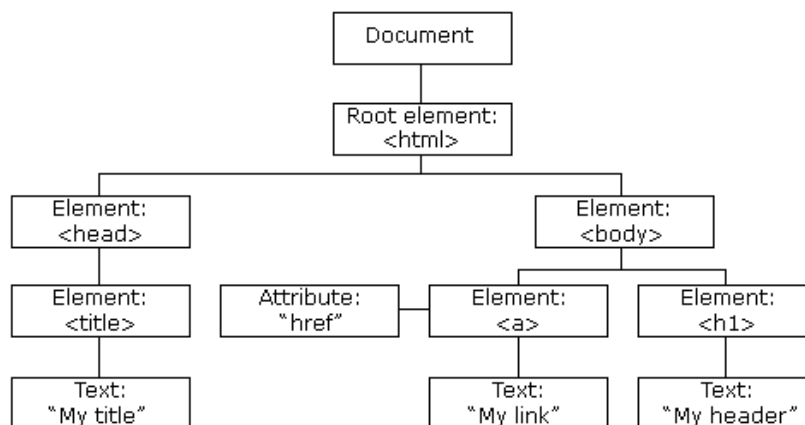
Di dalam objek `document`, terdapat fungsi-fungsi dan atribut yang bisa digunakan untuk memanipulasi dokumen HTML.

Sebagai contoh fungsi **`document.write()`**, fungsi ini digunakan untuk menulis sesuatu ke dokumen HTML.

Penggunaan DOM Hasilnya akan langsung berdampak pada dokumen HTML.

Mengakses Element tertentu dengan DOM

Objek `document` adalah model dari dokumen HTML. Objek ini berisi kumpulan fungsi dan atribut berupa objek dari elemen HTML yang bisa digambarkan dalam bentuk pohon seperti dibawah ini:



Struktur pohon diatas akan memudahkan dalam menggunakan elemen tertentu pada HTML.

Sebagai contoh akan coba mengakses objek **<head>** dan **<body>**. Pada websie dinus.ac.id, dengan coba menuliskan kode berikut pada Console:

```
// mengakses objek head
document.head;

// mengakses objek body
document.body;

// melihat panakang judul pada objek title
document.title.length
```

Hasilnya sebagai berikut :

```
> document.head
< ▶ <head>...</head>
> document.body
< ▶ <body class="pace-done">...</body>
> document.title.length
< 71
>
```

Apabila ingin mengakses elemen yang spesifik, terdapat beberapa fungsi yang bisa digunakan sebagai berikut:

- **getElementById()** fungsi untuk memilih elemen berdasarkan atribut id.
- **getElementByName()** fungsi untuk memilih elemen berdasarkan atribut name.
- **getElementByClassName()** fungsi untuk memilih elemen berdasarkan atribut class.
- **getElementByTagName()** fungsi untuk memilih elemen berdasarkan nama tag.
- **getElementByTagNameNS()** fungsi untuk memilih elemen berdasarkan nama tag.
- **querySelector()** fungsi untuk memilih elemen berdasarkan query.

- **querySelectorAll()** fungsi untuk memilih elemen berdasarkan query.
- dan lain-lain.

Fungsi-fungsi di atas akan cukup sering digunakan untuk mengakses elemen tertentu.

Misalkan ada kode HTML seperti dibawah ini:

```
<div id="tutorial"></div>
```

Maka untuk memilih element tersebut di Javascript, bisa digunakan fungsi **getElementById()** seperti dibawah ini:

```
// memilih elemen dengan id 'tutorial'
var tutorial = document.getElementById('tutorial');
```

Variabel tutorial akan menjadi sebuah objek DOM dari elemen yang dipilih.

Praktikum 1 : buat file **dom-1.html**

```
<!DOCTYPE html>
<html>
<head>
  <title>Memilih Elemen Berdasarkan ID</title>
</head>
<body>

  <!-- Elemen div yang akan dipilih dari JS -->
  <div id="tutorial"></div>

  <script type="text/javascript">
    // mengakses elemen tutorial
    var tutorial = document.getElementById("tutorial");

    // mengisi teks ke dalam elemen
    tutorial.innerText = "Tutorial Javascript";

    // memberikan CSS ke elemen
    tutorial.style.backgroundColor = "gold";
    tutorial.style.padding = "10px";

  </script>

</body>
</html>
```

Pada contoh praktikum 1 diatas, dipilih element HTML dengan fungsi **`document.getElementById()`**. kemudian membuat objek untuk elemen tersebut. Setelah itu, bisa dilakukan apapun yang diinginkan seperti mengubah **teks** dan **style CSS**.

Bagaimana kalau ada lebih dari satu elemen yang dipilih?, misalnya akan dipilih elemen berdasarkan nama tag atau atribut class, maka elemen yang akan terpilih akan menjadi sebuah array. Karena memilih sekumpulan elemen, array tersebut akan berisi objek DOM dari elemen-elemen yang terpilih.

Praktikum 2 : buat file **dom-2.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>DOM API Javascript</title>
</head>

<body>

  <p class="paragraf">Lorem ipsum dolor sit amet consectetur adipisicing
elit.
    Quo quaerat recusandae qui ullam eaque cumque ea fugit,
    debitis commodi ipsum illo dolorum consequatur sed laudantium
suscipit quis magni,
    maiores in?</p>

  <p class="paragraf">Lorem ipsum dolor sit amet consectetur adipisicing
elit.
    Quo quaerat recusandae qui ullam eaque cumque ea fugit,
    debitis commodi ipsum illo dolorum consequatur sed laudantium
suscipit quis magni,
    maiores in?</p>

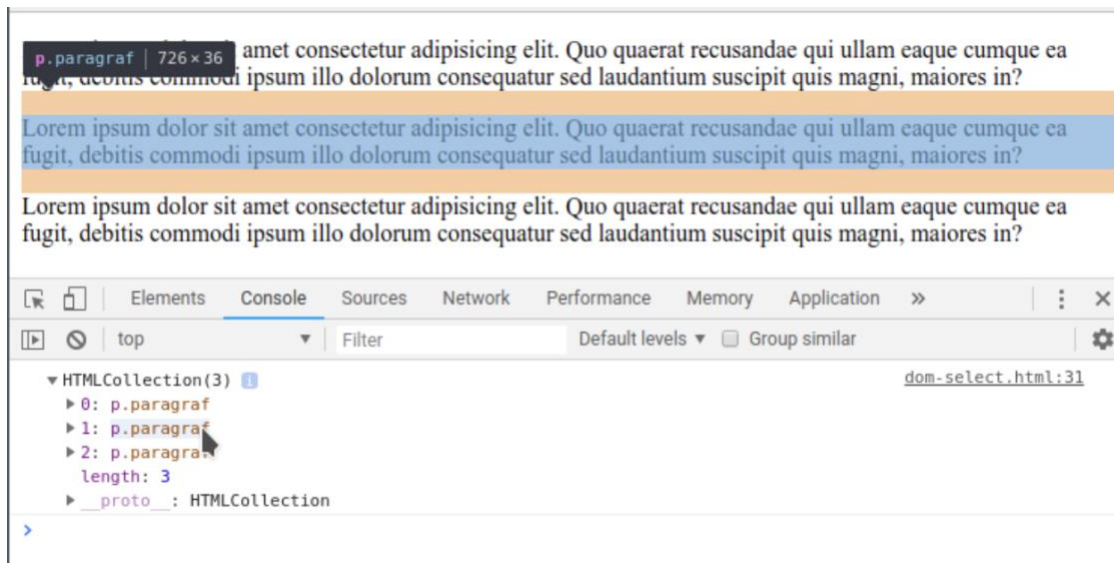
  <p class="paragraf">Lorem ipsum dolor sit amet consectetur adipisicing
elit.
    Quo quaerat recusandae qui ullam eaque cumque ea fugit,
    debitis commodi ipsum illo dolorum consequatur sed laudantium
suscipit quis magni,
    maiores in?</p>

  <script>
    var paragraf = document.getElementsByClassName("paragraf");
    console.log(paragraf);
  </script>
```

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

Pada contoh di atas, ada tiga buah paragraf dengan nama class paragraph, proses memilih ketiga paragraf melalui javascript dengan method atau fungsi `getElementsByClassName()`.

Kemudian, hasilnya bisa ditampilkan ke dalam console Javascript sebagai berikut:



Variabel paragraf akan berisi sebuah array yang di dalamnya terdapat tiga buah objek DOM dari paragraf.

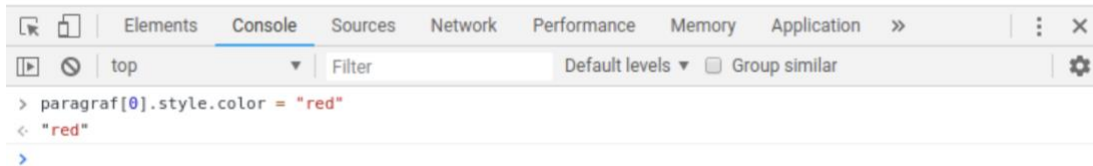
Bisa dicoba bereksperimen dengan mengubah warna teks dari paragraf pertama, paragraf pertama akan berada pada posisi indeks ke-0 di dalam array, misalkan ketik perintah berikut di dalam console Javascript:

```
paragraf[0].style.color = "red"
```

Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo quaerat recusandae qui ullam eaque cumque ea fugit, debitis commodi ipsum illo dolorum consequatur sed laudantium suscipit quis magni, maiores in?

Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo quaerat recusandae qui ullam eaque cumque ea fugit, debitis commodi ipsum illo dolorum consequatur sed laudantium suscipit quis magni, maiores in?

Lorem ipsum dolor sit amet consectetur adipisicing elit. Quo quaerat recusandae qui ullam eaque cumque ea fugit, debitis commodi ipsum illo dolorum consequatur sed laudantium suscipit quis magni, maiores in?



Paragraf Pertama akan berubah menjadi warna merah.

Berikutnya mencoba membuat animasi warna dari contoh diatas.

Praktikum 3 : buat file **dom-3.html**

Ubahlah kode javascript pada Latihan sebelumnya menjadi seperti dibawah ini:

```
<script>
var paragraph = document.getElementsByClassName("paragraf");
setInterval(function () {
    paragraph[0].style.color = "red";
    paragraph[1].style.color = "green";
    paragraph[2].style.color = "blue";

    setTimeout(function () {
        paragraph[0].style.color = "black";
        paragraph[1].style.color = "black";
        paragraph[2].style.color = "black";
    }, 500)
}, 1000);
</script>
```

Pada kode diatas, rentang waktu (*interval*) diberikan 1000 milidetik atau 1 detik, sedangkan untuk merubah warnanya menjadi hitam, diberikan waktu 500 milidetik atau 0.5 detik.

Membuat Elemen dengan DOM API

DOM juga menyediakan fungsi untuk membuat elemen HTML, salah satunya adalah fungsi `createElement()`.

Contoh :

```
document.createElement('p');
```

Maka, akan tercipta elemen `<p>` baru. Namun tidak akan ditampilkan ke dalam halaman web, karena belum ditambahkan ke dalam body dokumen.

Cara menambahkannya ke body dokumen, bisa menggunakan fungsi **`append()`**.

Praktikum 4 : buat file `dom-createElement.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>DOM API Javascript</title>
</head>

<body>

  <script>
    // membuat element h1
    var judul = document.createElement("h1");

    // mengisi konten elemen
    judul.textContent = "Belajar Javascript";

    // menambahkan elemen ke dalam tag body
    document.body.append(judul);
  </script>

</body>
</html>
```

Menghapus Elemen dengan DOM API

Untuk menambahkan elemen dengan menggunakan fungsi **`append()`**, maka untuk menghapusnya dapat menggunakan fungsi **`remove()`**.

Praktikum 5 : buat file **dom-remove.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>DOM API Javascript</title>
</head>

<body>

  <h2 id="judul2">Delete Saya!</h2>

  <script>
    // memilih elemen berdasarkan ID
    var h2 = document.getElementById('judul2');

    // menghapus elemen yang sudah dipilih
    h2.remove();

    console.log("Elemen sudah dihapus");
    console.log(h2);
  </script>

</body>
</html>
```

Berikutnya praktikum penggunaan DOM pada sebuah kasus merubah latar belakang dan teks pada sebuah halaman website.

Praktikum 6 : buat file **dom-contohKasus.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>DOM API Javascript</title>
</head>

<body>

  <h1>Aplikasi Ubah Warna</h1>
  <label>Warna latar: </label>
  <input type="color" id="bg-color" />
  <br>
  <label>Warna teks: </label>
  <input type="color" id="text-color" />

  <script>
```

```

    var bgColor = document.getElementById('bg-color');
    var txtColor = document.getElementById('text-color');

    bgColor.addEventListener("change", (event) => {
        document.body.style.backgroundColor = bgColor.value;
    });

    txtColor.addEventListener("change", (event) => {
        document.body.style.color = txtColor.value;
    });
</script>
</body>
</html>

```

FUNGSI JAVASCRIPT

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 dilakukan untuk membuat fungsi di Javascript:

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

Membuat Fungsi dengan Cara Biasa

```

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

```

Menggunakan ekspresi

```

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

```

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

Membuat Fungsi dengan Tanda Panah

Cara ini sering digunakan di kode Javascript masa kini, karena lebih sederhana, fungsi ini mulai muncul pada standar ES6.

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

Hampir sama dengan yang menggunakan ekspresi, bedanya menggunakan tanda panah (\Rightarrow) sebagai ganti function, pembuatan fungsi dengan cara ini disebut **arrow function**.

Membuat Fungsi dengan Kostruktur

Cara ini terlalu direkomendasikan oleh [Developer Mozilla](#), karena terlihat kurang bagus, karena **body** fungsinya dibuat dalam bentuk string yang dapat mempengaruhi kinerja **engine** javascript.

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

Kesimpulan : untuk tahap pemula dapat dipakai cara pertama, selanjutnya bisa menggunakan cara ke-2 dan ke-3.

Pemanggilan / Eksekusi Fungsi

Pemanggilan fungsi di dalam kode Javascript dengan menuliskan nama fungsinya seperti dibawah ini:

```
namaFungsi();
```

Contoh lengkap:

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

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

Praktikum 7 : buat file **fungsi-1.html**

```
<!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>
```

Fungsi dengan parameter

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

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

Pada contoh di atas, a dan b adalah sebuah parameter, kemudian cara memanggil fungsi yang memiliki parameter adalah sebagai berikut:

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

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
```

Berikutnya studi kasus pembuatan program berisi CRUD (*Create, Read, Update, Delete*) data barang yang tersimpan dalam sebuah array.

Praktikum 8 : ikut instruksi dibawah ini:

Buat dua file baru

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

File **index.html** adalah file yang menampilkan halaman web. Sedangkan file **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();
```

Bisa dimodifikasi dengan model deklarasi fungsi yang sudah dipelajari.

OBJEK JAVASCRIPT

Dalam kehidupan nyata, dapat dijumpai objek, objek adalah segala sesuatu yang ada baik itu benda mati ataupun makhluk hidup, semuanya adalah objek, objek-objek ini dapat dimodelkan di dalam pemrograman, yang paling umum dijumpai adalah pada paradigma OOP atau pemrograman beorientasikan objek, paradigma OOP ini merupakan sebuah teknik atau cara di dalam pemrograman dimana segala sesuatu di pandang sebagai objek, objek-objek dapat saling berinteraksi sehingga membentuk sebuah program.

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

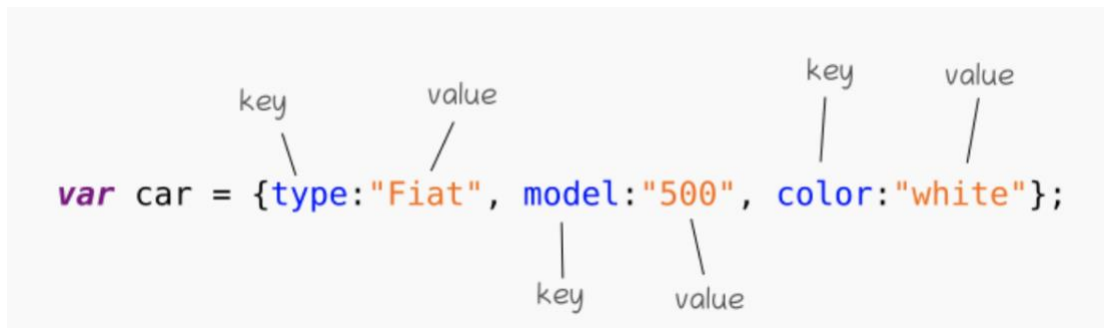
Sebagai contoh Objek = Mobil, maka bisa dilihat property dan atributnya sebagai berikut :



Cara memodelkan mobil ini di dalam kode program bisa saja seperti berikut ini:

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

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



The diagram shows the code `var car = {type:"Fiat", model:"500", color:"white"};` with labels pointing to the key-value pairs. For the first pair, 'type' is the key and 'Fiat' is the value. For the second pair, 'model' is the key and '500' is the value. For the third pair, 'color' is the key and 'white' is the value.

Kode di atas bisa ditulis seperti berikut:

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

Perbedaan Properti dan Method

Pada contoh sebelumnya baru sebatas membuat property, 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

Pada contoh sebelumnya sudah bisa membuat objek, kemudian berikutnya bagaimana cara mengakses properti dan method dari objek?, caranya adalah menggunakan tanda titik atau dot (.), dan diikuti dengan nama properti atau method.

```

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

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

```



Untuk mengakses properti, cukup dengan menggunakan nama **objek.properti**. Sedangkan untuk method, harus menggunakan tanda kurung. Ini dimaksudkan kalau akan mengeksekusi fungsi.

Menggunakan Keyword this

Kata kunci `this` digunakan untuk mengakses properti dan method dari dalam method (**objek**).

Praktikum 9 : buat file **objek-1.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 Objek Javascript</title>

```

```

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

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

```

Kata kunci **this** pada contoh di atas akan mengacu pada objek person.

Membuat Class Objek dengan this

Pada pemrograman berorientasikan objek, biasa membuat objek dengan membuat **instance** dari **class**, akan tetapi pada contoh-contoh di atas, membuat objeknya secara langsung.

Lalu bagaimana kalau ingin membuat objek yang lain dengan properti yang sama, maka bisa dibuat seperti contoh dibawah ini:

```

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

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

```

Ini belum efektif, jika ingin membuat banyak objek, karena harus menulis ulang kode yang sama, maka solusinya bisa menggunakan class.

Praktikum 10 : buat file **objek-class.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 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());
        }
    }

    var person1 = new Person("Agus", "Santoro");
    var person2 = new Person("Belajar", "Kode");

    person1.showName();
    document.write("<br>");
    person2.showName();
</script>
</head>
<body>

```

Repositori Pertemuan 12

Push hasil latihan ke Github dan kirim urlnya melalui kulino pada blok (Repositori Pertemuan 12).

Untuk susunan folder dan file sebagai berikut

- repominggu12 (folder utama)
 - latihanJavascript3
 - dom-1.html
 - dan seterusnya sesuai susunan folder Latihan diatas.