BINAR ACADEMY

Fungsi & Objek

Silver- Chapter 1 - Topic 3

Selamat datang di Chapter 1 Topic 3 pada course React Native dari Binar Academy!

Haalloo Binarian! 👋

Chapter 1 Topic 3 masih lanjut ya!
Setelah sebelumnya kamu belajar tentang Algoritma, pengambilan keputusan dan alur pengulangan pada pemrograman, di sesi kali ini kita bakalan banyak belajar tentang **Fungsi dan Objek** dalam pengembangan aplikasi.

Yaudah yuk, langsung saja kita sikat sesi ini.

Have fun!

Detailnya, kita bakal bahas hal-hal berikut ini:

- Function
- Object

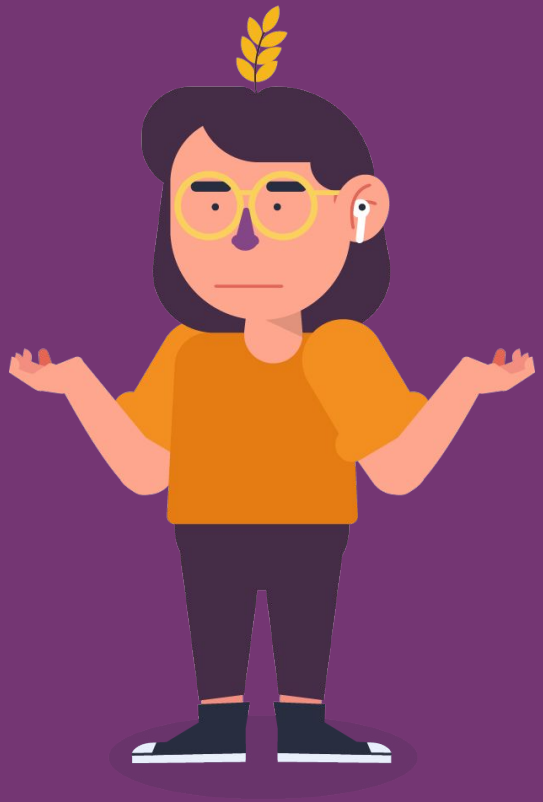
FUNCTION

Introduction

Saat bikin aplikasi biasanya kamu akan menemukan beberapa statement atau ekspresi yang sama, yang dibuat secara berulang. Kalau ini terjadi tentu bisa jadi hal yang bakal ngerepotin. Karena itulah **function (fungsi)** ada di JavaScript.

Nggak cuma JavaScript, sebagian besar bahasa pemrograman juga punya fungsi kok. Secara umum perilakunya sama, cuma cara nulisnya aja yang beda.

Menurut kamu apa itu Function?



Function

Function adalah sebuah blok kode yang disusun sedemikian rupa untuk jalanin **sebuah tindakan.**
Blok kode ini dibuat agar bisa **digunakan kembali.** Sebuah function umumnya melakukan tindakan dan sebelum function berakhir, function bisa **mengembalikan nilai** dengan cara menambahkan sintaks **return**

Kenapa pakai *function?*

Kayak kalau kita mau bikin salinan tugas kampus di komputer, tapi please deh, males banget kan ya kalau kita harus ngetik ulang. Mendingan copy-paste saja. Tinggal klik, jadi deh salinan yang kita butuhin. Simpel, cepat, ringkas, gampang!

Nah, sama saja nih..

Kadang-kadang ada sebuah **blok code/statement** yang perlu kamu pakai **berkali-kali** di dalam aplikasi.Daripada kamu **tulis** blok code tersebut **berulang-ulang**, bikin ribet dan nggak efisien, mendingan dibikin jadi sebuah *function*.Dengan *function*, kita **tulis code-nya sekali**, tapi **bisa dipakai berkali-kali**. Jadi lebih praktis.

BINAR

3 Jenis Function

Apa saja ya ketiga jenis fungsi ini?

Yuk mari kita intip satu-persatu, Binarian.

BINAR

Function

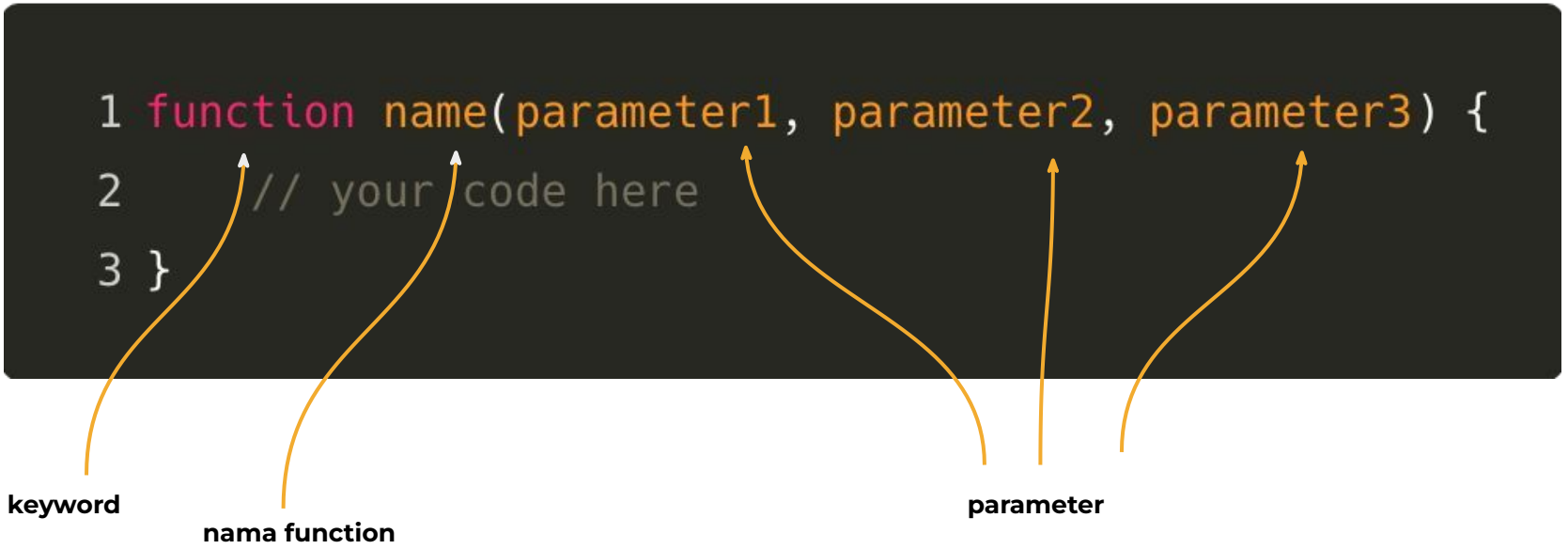
Ada 3 cara yang bisa kamu lakukan untuk bikin **function** di **JavaScript**

1. **Function Biasa**
2. **Function Expression**
3. **Arrow Function**

Function Biasa

Begini nih cara bikinnya:

```
1 function name(parameter1, parameter2, parameter3) {
2     // your code here
3 }
```



BINAR

Contoh:

Dengan parameter

```
1 function perkalian(angka1, angka2) {
2   return angka1*angka2;
3 }
4
5 let hasilPerkalianSatu = perkalian(3,4)
6 let hasilPerkalianDua = perkalian(10,5)
7 console.log(hasilPerkalianSatu) //12
8 console.log(hasilPerkalianDua) //50
```

Tanpa parameter

```
1 function helloWord() {
2     alert("Hello World!")
3 }
4
5 helloWord()
```

Function Biasa

Contoh:

Dengan parameter

```
1 function perkalian(angka1, angka2) {
2   return angka1*angka2;
3 }
4
5 let hasilPerkalianSatu = perkalian(3,4)
6 let hasilPerkalianDua = perkalian(10,5)
7 console.log(hasilPerkalianSatu) //12
8 console.log(hasilPerkalianDua) //50
```

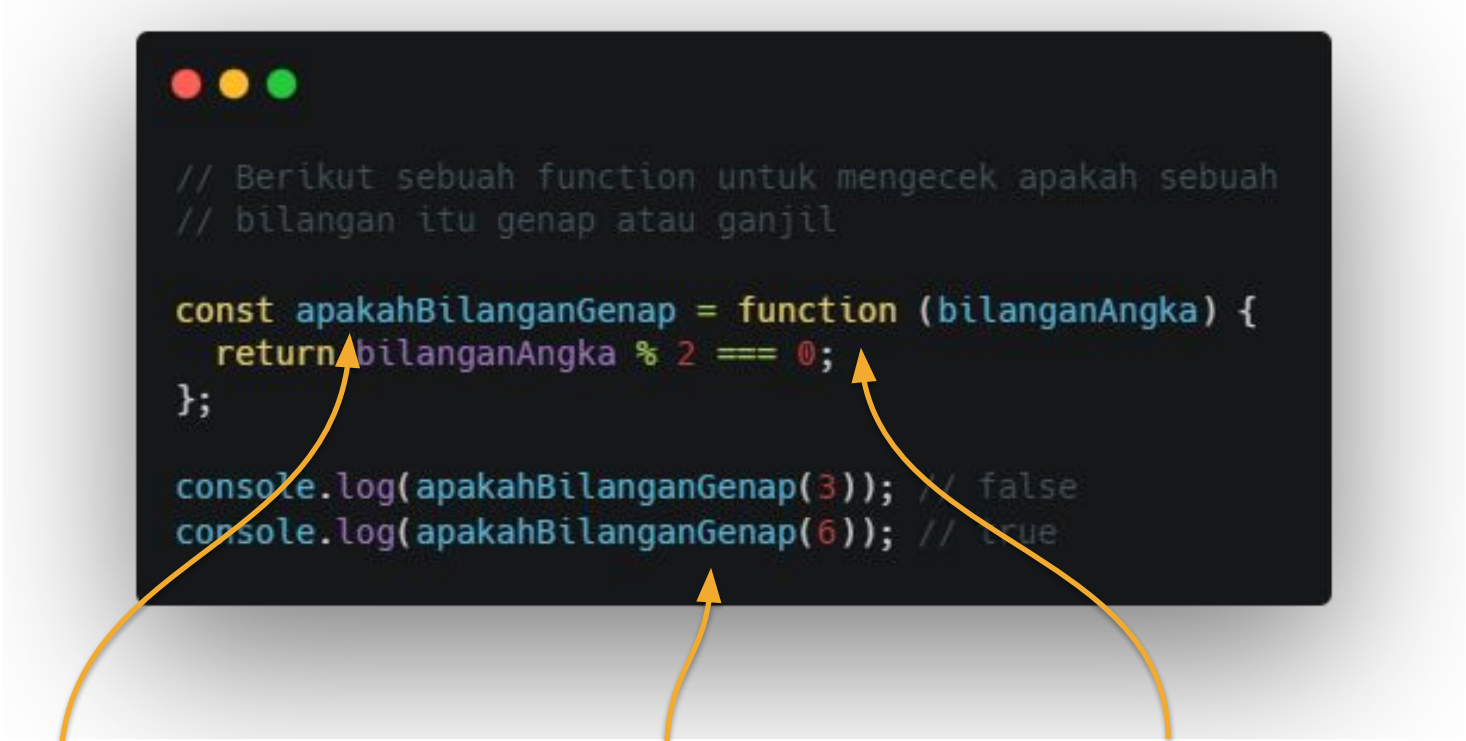
Tanpa parameter

```
1 function helloWord() {
2     alert("Hello World!")
3 }
4
5 helloWord()
```

Function Expression

Adalah jenis function yang **pakai nama variabel sebagai nama function tersebut**, karena itu nama tepat setelah kata kunci function biasanya tidak ditulis.

Yuk lihat contohnya!


// Berikut sebuah function untuk mengecek apakah sebuah
// bilangan itu genap atau ganjil
const apakahBilanganGenap = function (bilanganAngka) {
return bilanganAngka % 2 === 0;
};
console.log(apakahBilanganGenap(3)); // false
console.log(apakahBilanganGenap(6)); // true
Nama Variabel jadi nama Function
Nama variabel yang nampung function dipanggil
Functionnya tanpa nama

Arrow Function

Arrow Function atau **fungsi panah** mirip kayak Function Expression, tapi nggak perlu kata kunci function, yang diperlukan adalah tanda panah di antara kurung lengkung dan kurung kurawal.

Fungsi panah diawali dengan sepasang kurung lengkung parameter *(param1, param2, ..., paramN),* diikuti oleh tanda sama dengan lebih besar atau tanda panah (**=>**), lalu diakhiri dengan kurung kurawal { ... } sebagai batas badan fungsi.

Yuk lihat contohnya!

Arrow Function

```
// Berikut sebuah function untuk mengecek apakah sebuah
// bilangan itu genap atau ganjil

const apakahBilanganGenap =  (bilanganAngka) => {
  return bilanganAngka % 2 === 0;
};

console.log(apakahBilanganGenap(3)); // false
console.log(apakahBilanganGenap(6)); // true
```

Nama Variabel jadi nama Function

Nama variabel yang menampung arrow **function** dipanggil

Nggak ada keyword **function**, keyword functionnya diganti dengan simbol panah (**=>**)

Yuhuu, sekarang coba satu orang dari kalian jelaskan perbedaan antara ketiga function yang sudah kita pelajari.

Siapa yang bisa?

```
// Berikut sebuah function untuk mengecek apakah sebuah
// bilangan itu genap atau ganjil

function apakahBilanganGenap (bilanganAngka) {
  return bilanganAngka % 2 === 0;
};

console.log(apakahBilanganGenap(3)); // false
console.log(apakahBilanganGenap(6)); // true
```

Function Biasa (Function Declaration)

```
// Berikut sebuah function untuk mengecek apakah sebuah
// bilangan itu genap atau ganjil

const apakahBilanganGenap =  (bilanganAngka) => {
  return bilanganAngka % 2 === 0;
};

console.log(apakahBilanganGenap(3)); // false
console.log(apakahBilanganGenap(6)); // true
```

Function Expression

```
// Berikut sebuah function untuk mengecek apakah sebuah
// bilangan itu genap atau ganjil

const apakahBilanganGenap = function (bilanganAngka) {
  return bilanganAngka % 2 === 0;
};

console.log(apakahBilanganGenap(3)); // false
console.log(apakahBilanganGenap(6)); // true
```

Arrow Function

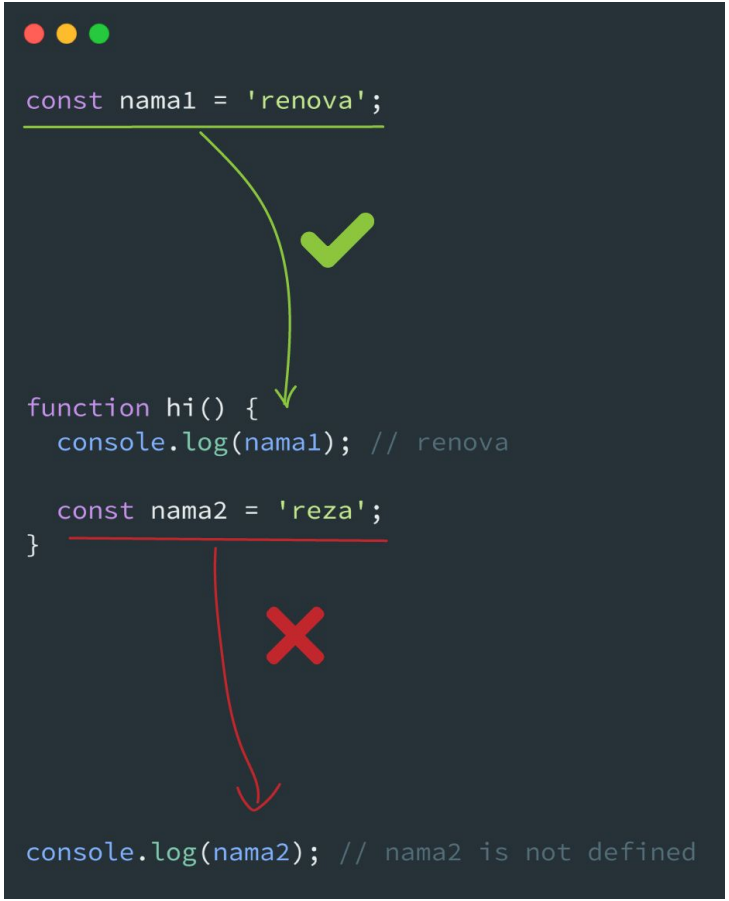
Function Scope

Function Scope

Selanjutnya, kita masuk ke cakupan fungsi alian **function scope**.

Lihat deh kode di samping ini, kita mendeklarasikan variabel bernama **nama2** di dalam function **hi**, kemudian kita coba pakai variabel **nama2** tersebut di luar function **hi** . Hasilnya, kita mendapatkan error **nama2** *is not defined*.

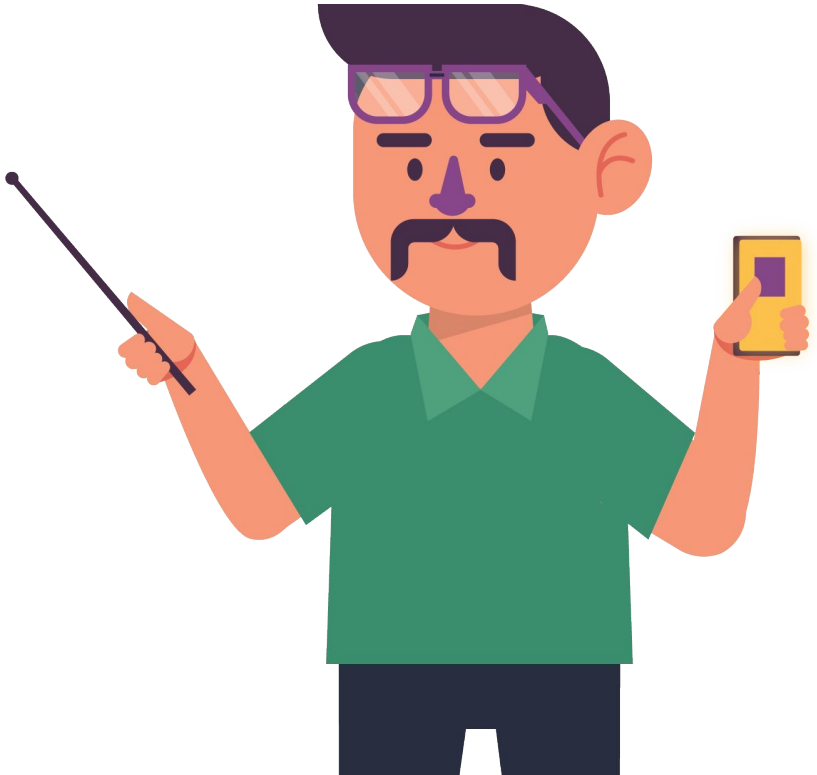
Ada yang tahu nggak, kenapa?



Karena...

Variabel yang dideklarasikan **di dalam sebuah function nggak bisa diakses di luar function**, sedangkan variabel yang dideklarasikan di luar function bisa diakses di dalam function.

Inilah yang dinamakan dengan **Function Scope.**
Sejauh mana fungsi bisa diakses.



Function Scope

BINAR

Global Scope

```
//Global Scope
var globalVariable = 10;

function globalFunction() {
  //Function Scope
  var functionVariable = 7;

  if (functionVariable > 3) {
    //Block Scope
    let blockVariable = 4;
    const canNotBechanged = 8;

    console.log(blockVariable);
  }
}

globalFunction();

console.log(globalVariable);
```

Functiom Scope

Block Scope

BINAR

Object

Object

Dalam kehidupan nyata, kita pasti akan ketemu dan ngelihat objek. Objek adalah segala sesuatu yang ada di dunia ini. Apapun yang kita lihat adalah objek. Benda mati atau makhluk hidup, semuanya objek.

Objek-objek inilah yang bisa kita modelkan di dalam pemrograman.

Apa itu objek pada Javascript?
Gimana cara bikinnya?

Begini, Binarian..

objek adalah..

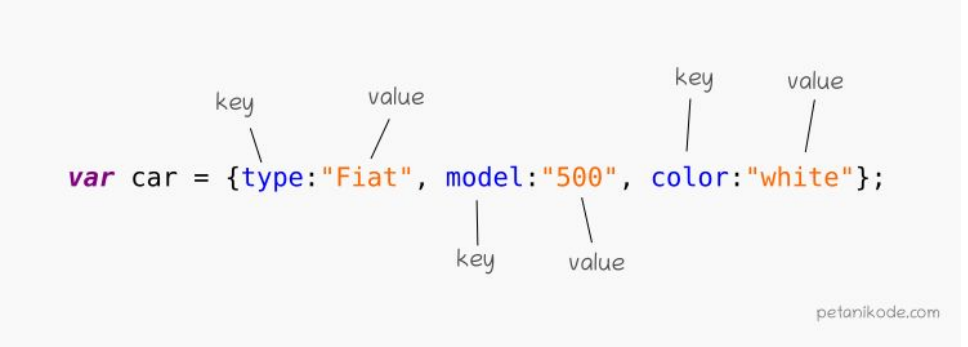
Objek (object) sebenarnya adalah sebuah variabel yang nyimpan nilai (properti) dan fungsi (method).

Contoh:
Object dari sebuah mobil pada gambar disamping. Bagaimana cara kita memodelkan mobil ini di dalam kode program?

Caranya, bisa saja dengan bikin sebuah variabel car kemudian memasukan nama mobilnya.

Karena variable car yang kamu bikin pada kode di samping cuma nyimpan nama mobil saja, maka kamu juga harus menggunakan **objek agar bisa menyimpan informasi value terkait** dengan lebih rapi.

Terus, gimana cara membuatnya menggunakan objek?

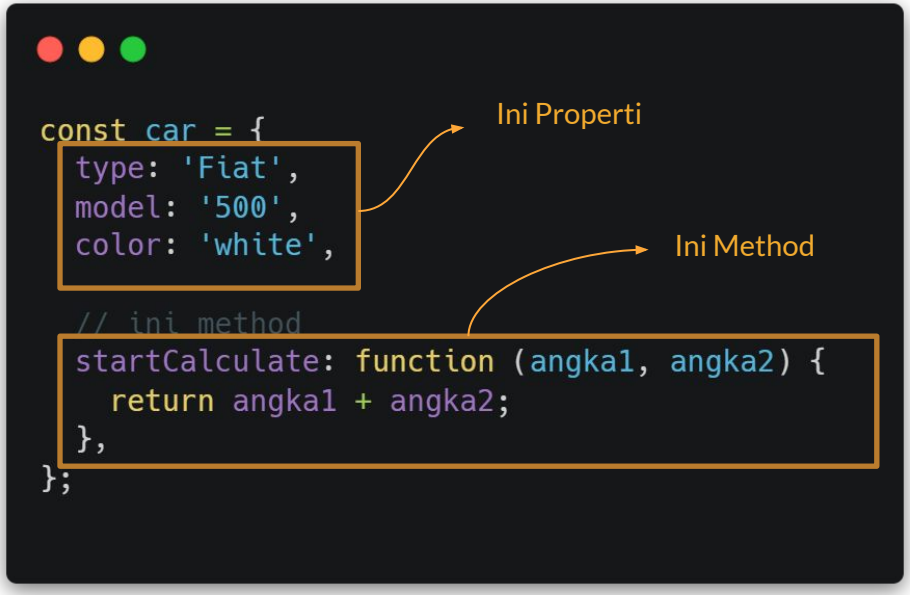


Objek pada JavaScript, bisa dibikin dengan tanda kurung kurawal dengan isi berupa *key* dan *value*.

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

Apa Perbedaan Properti dan Method?

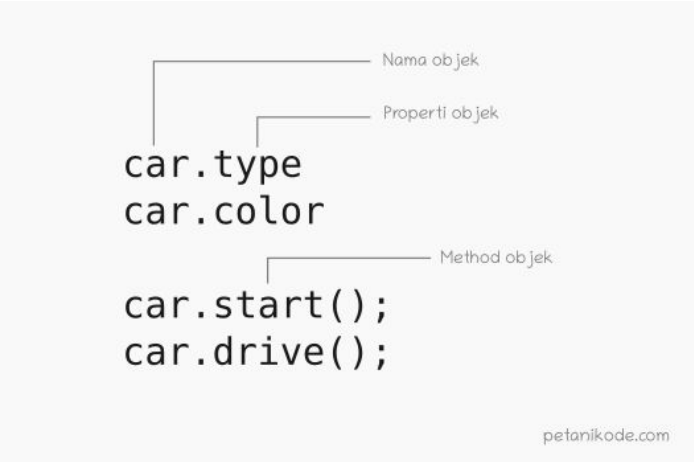
Properti adalah ciri khas dari objek (variabel). Sedangkan *method* adalah perilaku dari objek (function) atau dengan kata lain method adalah sebuaH function yang berada didalam suatu objek.



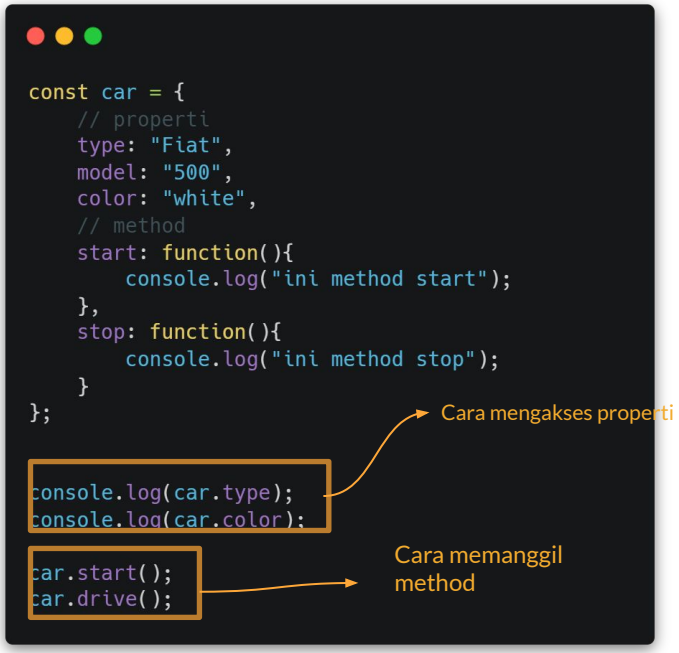
Sebuah method dapat dibikin dengan cara mengisi nilai/value dengan sebuah function

```
const 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");
    }

};
```

Terus bagaimana cara akses nilai/value properti dan method pada objek?



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



BINAR

Akses properti pada object secara dynamis

Kelebihan mengakses property dengan bracket ([]) adalah kamu bisa masukin variable saat kamu nyoba akses property tersebut pada sebuah object.

```
const car = {
    type: "Toyota Avanza",
    model: "500",
    color: "white",
};

const namaPropertiYangAkanDiakses = "color"


// tanpa tanda kutip, karena kita memanggil sebuah variabel
car[namaPropertiYangAkanDiakses];
```

Akses properti pada object secara dynamis

Selain pakai dot(.), kamu juga bisa pakai bracket untuk akses property. Perhatiin deh contoh kode di samping.

```
const car = {
    type: "Toyota Avanza",
    model: "500",
    color: "white",
};

// Mengakses properti menggunakan dot (.)
car.type;
car.model;
car.color;

// Mengakses properti menggunakan bracket ([])
// boleh pake tanda kutip satu atau kutip dua.

car['type'];
car["model"];
car["color"];
```

Menambahkan properti pada sebuah objek yang telah dibuat

Pada sebuah objek JavaScript kamu bisa nambahin property atau method saat objek tersebut sudah dibikin.

Misalnya kamu sudah bikin sebuah objek car dengan property type, model, dan color. Lalu tiba-tiba kamu mau nambahin property jenisBensin pada objek car yang telah kamu buat sebelumnya.

Bagaimana caranya?
Yuk perhatikan kode disamping

```
const car = {
    type: "Toyota Avanza",
    model: "500",
    color: "white",
};


// menambahkan properti pada objek mengunakan dot
car.jenisBensin = 'Solar';
// menambahkan properti pada objek menggunakan bracket
car['jenisBensin'] = 'Solar';


// Ketika di console.log yang akan muncul nantinya
// adalah sebuah objek dan terdapat properti baru yang
// telah ditambahkan yaitu 'jenisBensin'
console.log(car);
```

Menghapus properti pada sebuah objek

Selain itu, kamu juga bisa hapus sebuah property pada objek yang sudah kamu bikin.

Kamu bisa pakai keyword ‘delete’ pada JavaScript, lalu diikutin oleh nama objek dan nama property yang akan kamu hapus pada objek tersebut.

```
const car = {
    type: "Toyota Avanza",
    model: "500",
    color: "white",
};

// Kamu bisa menggunakan keyword delete objek.properi
// untuk menghapus properti atau method pada sebuah objek
delete car.color;


// Ketika di console.log isi dari object car hanya ada
// tersisa dua properti type dan model, karena kita
// sebelumnnya telah menghapus properti color
console.log(car)
```

BINAR

Saatnya kita Quiz!

BINAR

1. **Berikut ini adalah pernyataan yang benar seputar tentang objek, kecuali..**

   A. Properti pada objek bisa ditambahin, meskipun objek tersebut sudah dibuat

   B. Function yang ada di dalam sebuah objek disebut Methode

   C. Properti pada objek nggak bisa dihapus kalau objek tersebut telah dibuat

BINAR

1. **Berikut ini adalah pernyataan yang benar seputar tentang objek, kecuali..**

A. Properti pada objek bisa ditambahin, meskipun objek tersebut sudah dibuat

B. Function yang ada di dalam sebuah objek disebut Methode

C. Properti pada objek nggak bisa dihapus kalau objek tersebut telah dibuat

Properti pada objek selalu bisa dihapus pakai keyword ‘delete ‘

BINAR

2. Apa output yang akan dihasilkan dari kode di bawah?

A. Error: ‘hargaCar’ is not defined

B. Nggak terjadi apa-apa, cuma akan hasilin output objek car yang sudah dibuat

C. Error: car is invalid data type

```
const car = {
    type: "Toyota Avanza",
    model: "500",
    color: "white",
};

delete car.hargaCar;
console.log(car)
```

BINAR

2. Apa output yang akan dihasilkan dari kode di bawah?

A. Error: ‘hargaCar’ is not defined

B. Nggak terjadi apa-apa, cuma akan hasilin output objek car yang sudah dibuat

C. Error: car is invalid data type

```
const car = {
    type: "Toyota Avanza",
    model: "500",
    color: "white",
};

delete car.hargaCar;
console.log(car)
```

Ketika kamu menghapus properti yang nggak ada pada sebuah objek, maka JavaScript nggak akan menganggap itu sebagai sebuah error.

BINAR

3. Apa output console.log yang akan dari kode di bawah?

A. Madeline Daniela Pranata

B. AlexeiInessa Daniela Pranata

C. Alexei Daniela Pranata

```
const dataPeserta = {
    firstName: 'Aldi',
    middleName: 'Daniela',
    lastName: 'Pranata',
    age: 15,
    hobbies: ['Photography', 'Singing', 'Badminton'],
}

dataPeserta.firstName = 'Madeline';
dataPeserta['firstName'] = 'Alexei';
dataPeserta.firstName = dataPeserta.firstName + 'Inessa'


console.log(`${dataPeserta.firstName} ${dataPeserta.middleName} ${{dataPeserta.lastName}`)
```

BINAR

3. Apa output console.log yang akan dari kode di bawah?

A. Madeline Daniela Pranata

B. AlexeiInessa Daniela Pranata

C. Alexei Daniela Pranata

```
const dataPeserta = {
    firstName: 'Aldi',
    middleName: 'Daniela',
    lastName: 'Pranata',
    age: 15,
    hobbies: ['Photography', 'Singing', 'Badminton'],
}

dataPeserta.firstName = 'Madeline';
dataPeserta['firstName'] = 'Alexei';
dataPeserta.firstName = dataPeserta.firstName + 'Inessa'


console.log(`${dataPeserta.firstName} ${dataPeserta.middleName} ${{dataPeserta.lastName}`)
```

Karena properti firstName pada object dataPeserta sudah di re-assign

```
const dataPeserta = {
  firstName: 'Aldi',
};
const dataPeserta2 = {
  middleName: 'Daniela',
};

const hasilDataPeserta = { ...dataPeserta, ...dataPeserta2 };

console.log(hasilDataPeserta);
```

4. Apa output console.log yang akan dari kode di bawah?

A. Akan memunculkan objek dengan property firstName

B. Akan memunculkan objek dengan property firstName dan middleName

C. Akan memunculkan objek dengan property middleName

```
const dataPeserta = {
  firstName: 'Aldi',
};
const dataPeserta2 = {
  middleName: 'Daniela',
};

const hasilDataPeserta = { ...dataPeserta, ...dataPeserta2 };

console.log(hasilDataPeserta);
```

4. Apa output console.log yang akan dari kode di bawah?

A. Akan memunculkan objek dengan property firstName

B. Akan memunculkan objek dengan property firstName dan middleName

C. Akan memunculkan objek dengan property middleName

...namaObjek => akan mengekstrak isi dari objek tersebut, teknik ini juga sering disebut dengan object desctructur

Penutup

Nah, selesai sudah pembahasan Chapter 1 Topic 3 bagian Fungsi dan Objek. Selanjutnya, kita bakal bahas Topic 3 dengan materi yang lain.

Penasaran kayak gimana?

Cus langsung ke topik selanjutnya yaa!

Terima Kasih!

Next Topic

loading...

BINAR