Programming Fundamental

JS

Exploring

#8 Object

Purwadhika
Startup and Coding School

Intro

In JavaScript, almost "*everything*" is an **object**. All JavaScript values, except primitives, are objects.

A primitive value is a value that has no properties or methods. A primitive data type is data that has a primitive value.

JavaScript defines 4 types of primitive data types:
String, Number, Boolean, undefined

Object

JavaScript variables can contain single values:

```
var orang = "Andi";
```

Objects are variables too. But objects can contain ***many values***. The values are written as ***name & value pairs*** (separated by a colon). A JavaScript object is a collection of named values

```
var orang = {
      namaDpn : "Andi", namaBlkg : "Susilo",
      usia : 50, pekerjaan : "Politisi"
};
```

Object

```
var orang = {
    namaDpn:'Andi',
    usia:50,
    pekerjaan:'Politisi',
    namaFull: function() {
        return this.namaDpn + " " + " Karno";}
};
```

Properties are the values associated with a JavaScript object **[namaDpn, usia, pekerjaan, namaFull]**

Values are the properties' value **[Andi, 50, Politisi]**

Method is a property containing a function definition. [**namaFull()**]

Creating a JavaScript Object

- Define and create a single object, using an ***object literal***.
- Define and create a single object, with the ***keyword 'new'***.
- Define an ***object constructor***, and then create objects of the constructed type.

Creating a JavaScript Object
#1 Using an Object Literal

```
var Andi = {
    namaDpn: "Andi",
    namaBlkg: "Susilo",
    usia: 50,
    pekerjaan: "Politisi"
};
```

Purwadhika
Startup and Coding School

Creating a JavaScript Object
#2 Using JavaScript Keyword *'new'*

```
var Andi = new Object();

Andi.namaDpn = 'Andi';
Andi.namaBlkg = 'Susilo';
Andi.usia = 50;
Andi.pekerjaan = 'PNS';
```

Creating a JavaScript Object
#3.1 Using an Object Builder*

Object Constractor

The examples #1 & #2 are limited in many situations. They only create a single object. Sometimes we like to have an "object type" that can be used to create many objects of one type.

```
var orang = function(nama, usia, job) {
        this.namaDpn = nama;
        this.umur = usia;
        this.pekerjaan = job;
    }

var Andi = new orang('Andi', 30, 'PNS');
console.log(Andi.umur)
```

Creating a JavaScript Object
#3.2 Using an Object Builder*

Object Constractor

The examples #1 & #2 are limited in many situations. They only create a single object. Sometimes we like to have an "object type" that can be used to create many objects of one type.

```
function orang(nama, usia, job) {
    this.namaDpn = nama;
    this.umur = usia;
    this.pekerjaan = job;
    }

var Andi = new orang('Andi', 30, 'PNS');
console.log(Andi.umur)
```

Creating a JavaScript Object #4 Using an JavaScript Class*

The examples #1 & #2 are limited in many situations. They only create a single object. Sometimes we like to have an "object type" that can be used to create many objects of one type.

```
class orang {
    constructor(nama, usia, job) {
        this.namaDpn = nama;
        this.umur = usia;
        this.pekerjaan = job;
    }
}
var Andi = new orang('Andi', 30, 'PNS');
console.log(Andi.umur)
```

Purwadhika
Startup and Coding School

The ‘*this*’ Keyword

In JavaScript, the thing called ***this***, is the object that "owns" the JavaScript code.

The value of ***this***, when used in a function, is the object that "owns" the function.

The value of ***this***, when used in an object, is the object itself.

The ***this*** keyword in an object constructor does not have a value. It is only a substitute for the new object.

The value of ***this*** will become the new object when the constructor is used to create an object.

Accessing Obj. Properties

```
var orang = {
    namaDpn : "Andi",
    usia : 50,
    job : "Politisi"
};

console.log(orang.namaDpn+' usianya
'+orang.usia+' tahun.')

console.log(orang["namaDpn"]+'
seorang '+orang["job"]+'.')
```

Adding/Deleting Properties

```
var orang = {
    namaDpn : "Andi",
    usia : 50,
    job : "Politisi"
};

orang.negara = "Indonesia";
delete orang.usia;

console.log(orang)
```

Purwadhika
Startup and Coding School

Adding/Deleting Properties

```
function orang(nama, usia, job) {
    this.namaDpn = nama;
    this.umur = usia;
    this.pekerjaan = job;
}

var Andi = new orang('Andi', 30, 'PNS');

Andi.negara = 'Indonesia'
console.log(Andi)
```

Property Array

```
var Andi = {
    nama : ['Andi', 'Susilo'],
    usia : 50,
    job : "Politisi"
};

console.log(Andi)
console.log(Andi.nama[0])
console.log(Andi.nama[1])
```

Property Array

```
function orang(awal, akhir, usia) {
        this.nama = [awal, akhir];
        this.usia = usia;
}

var Andi = new orang
('Andi','Susilo',20);

console.log(Andi)
console.log(Andi.nama[0])
console.log(Andi.nama[1])
```

Property Obj

```
var Andi = {
    nama : 'Andi',
    usia : 50,
    job : {
          nama: 'PNS',
          tingkat: 'Eselon IIIA',
          lokasi: 'Bali'
     }
};
console.log(Andi.job.nama)
console.log(Andi.job.lokasi)
```

Purwadhika
Startup and Coding School

Property Obj

```
function orang(awal, akhir, usia) {
        this.nama = {awal, akhir};
        this.usia = usia;
}
var Andi = new orang
('Andi','Susilo',20);

console.log(Andi)
console.log(Andi.nama.awal)
console.log(Andi.nama.akhir)
```

Purwadhika
Startup and Coding School

Accessing Object Methods

```
var orang = {
    namaDpn : "Andi",
    namaBlk : "Susilo",
    usia : 50,
    namaFull : function() {
        return this.namaDpn + " " +
        this.namaBlk;
    }
};
console.log(orang.namaFull)
console.log(orang.namaFull())
```

Adding New Methods

```
function person(x, y, z) {
     this.namaDpn = x,
     this.namaBlk = y,
     this.usia = z,
     this.gantiNama = function(a) {
          this.namaBlk = a
     }
}

let Andi = new person("Andi","Susilo",24)
console.log(Andi.namaBlk)

Andi.gantiNama("Darmawan")
console.log(Andi.namaBlk)
```

Check Properties

```
var manusia = {
        kepala: 1,
        mata: 2,
        telinga: 2,
        tangan: 2,
        kaki: 2,
};

var namaProp;
for (namaProp in manusia) {
console.log(namaProp + ":" +
manusia[namaProp]);
}
```

Prototype

All JavaScript objects inherit the properties and methods from their prototype.

Objects created using an object literal, or with new Object(), inherit from a prototype called Object.prototype.

Objects created with new Date() inherit the Date.prototype.

The Object.prototype is on the top of the prototype chain.

All JavaScript objects (Date, Array, RegExp, Function, ...) inherit from the Object.prototype.

Creating a Prototype

```
function Orang(nama, lahir, job) {
     this.nama = nama;
     this.lahir = lahir;
     this.job = job;
}
Orang.prototype.marga = 'Hasibuan';
Orang.prototype.usia = function() {
    return 2017 - this.lahir;
};

var budi = new Orang('Budi', 1992, 'PNS');
console.log(budi.marga);
console.log(budi.usia());
```

Purwadhika
Startup and Coding School

Inheritance

```
function orang(nama, usia, job) {
    this.namaDpn = nama;
    this.usia = usia;
    this.pekerjaan = job;
}
function atlit(nama, usia, job, cab, pres){
    orang.call(this, nama, usia, job)
    this.cabor = cab;
    this.prestasi = pres;
}
var Andi = new atlit();
console.log(Andi)
```

Purwadhika
Startup and Coding School

Inheritance

```
function orang(nama, usia, job) {
    this.namaDpn = nama;
    this.usia = usia;
    this.pekerjaan = job;
}
function atlit(nama, usia, job, cab, pres){
    orang.call(this, nama, usia, job)
    this.cabor = cab;
    this.prestasi = pres;
}
var Andi = new atlit
('Andi',20,'PNS','Kuda', '0');
console.log(Andi)
```

Purwadhika
Startup and Coding School

Inheritance w/out Argument*

Parameter

```
function pizza() {
        this.diameter = 30;
    }
function pizzaKeju(){
        pizza.call(this)
        this.topping = 'Keju';
        this.harga = '50K';
    }

var satu = new pizzaKeju();
console.log(satu)
```

Purwadhika
Startup and Coding School

Inheritance w/out Argument*

Parameter

```
function pizza() {
        this.diameter = 30;}

function pizKeju(){
        pizza.call(this)
        this.topping = 'Keju';
        this.harga = '50K';}

function pizJamur(){
        pizza.call(this)
        this.topping = 'Jamur';
        this.harga = '65K';}

var satu = new pizKeju();
var dua = new pizJamur();
console.log(satu)
console.log(dua)
```

Purwadhika
Startup and Coding School

Contoh Aplikasi

Buatlah sebuah Object Builder [Class] untuk membuat Object Persegi dengan properti utama: Luas & Keliling!

Persegi

```
function persegi(sisi) {
    this.sisi = sisi;
    this.luas = function(){
        return Math.pow(this.sisi,2);
    };
    this.kll = function(){
        return 4 * this.sisi;
    };
};

var x = new persegi(8);
console.log('Luas = '+x.luas());
console.log('Keliling = '+x.kll());
```

Objects Assign

```
var jomblo = true

var Andi =
{usia:27, job:'Polisi'};

var Budi =
Object.assign({}, {usia:32, job:'Pilot'});

var Caca =
Object.assign({}, Budi, {job:'Akuntan'});

var Dedi =
Object.assign({}, {job:'Guru'}, {jomblo});

console.log(Budi);
console.log(Caca);
console.log(Dedi);
```

Purwadhika
Startup and Coding School

Array of Objects

```
var arrayku = [
    {nama : 'andi', usia: 27},
    {nama: 'budi', usia: 25},
    {nama: 'caca', usia: 23},
]

console.log(arrayku[0].nama)
console.log(arrayku[0].usia)
console.log(arrayku[1].nama)
console.log(arrayku[2].usia)
```

Sorting Array of Objects

```
// Number rendah ke tinggi
arrayku.sort(function(x,y){
    return x.usia - y.usia
})

// Number tinggi ke rendah
arrayku.sort(function(x,y){
    return y.usia - x.usia
})
```

Purwadhika
Startup and Coding School

Sorting Array of Objects

```
// String dari A ke Z (Ascending)
arrayku.sort(function(x,y){
    if (x.nama < y.nama){return -1}
    if (x.nama > y.nama){return 1}
    return 0
})

// String dari Z ke A (Descending)
arrayku.sort(function(x,y){
    if (x.nama > y.nama){return -1}
    if (x.nama < y.nama){return 1}
    return 0
})
```

Purwadhika
Startup and Coding School

Programming Fundamental

JS

Exploring

#8 Object

Purwadhika
Startup and Coding School