

# NODE.JS WORKSHOP

## HARI 1: MENGENAL JS

---

*Adityo Pratomo (Framework)*  
*Mozilla, 18 September 2016*



# TENTANG FRAMEWORK

---

- Framework adalah perusahaan yang memfokuskan diri pada menyediakan pelatihan software development
- Produk kami: in-house training, workshop dan kelas
- Bidang yang kami cakup: Web development, IoT dan Creative Coding
- Lokasi di BSD, berdiri sejak awal 2016



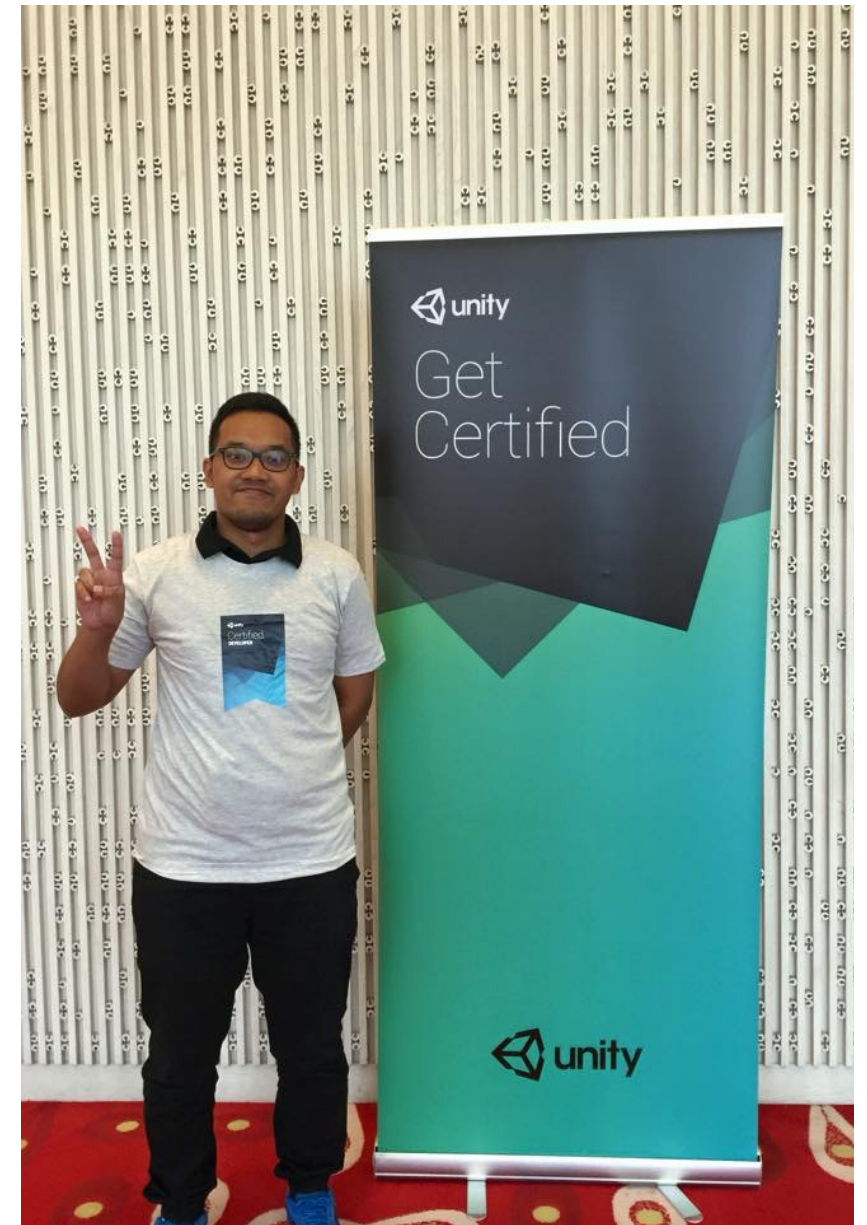
**Froyo**

**FRAMEWORK**

# TENTANG TRAINER

---

- Adityo Pratomo (Didit)
- Generalist, tapi lebih senang game programming/creative coding
- Certified Unity Developer  
#20167UCD724
- kontak: [didit@froyo.co.id](mailto:didit@froyo.co.id)



# MATERI KELAS

---

- Pertemuan 1: Mengenal JS
- Pertemuan 2: Bekerja dengan API Node JS
- Pertemuan 3: Membuat Aplikasi dengan Web Framework Node

# MATERI HARI INI

---

- Mengenal Node.js
- Menggunakan Node.js
- Konsep Pemrograman JS
- Variabel
- Function
- Array
- Object

# MENGENAL NODE.JS

---

- Node.js adalah sebuah platform untuk pengembangan aplikasi jaringan yang berbasis JavaScript
- Digunakan untuk: web app, aplikasi di sisi server, aplikasi client, bahkan sebagai general purpose programming tool
- Bentuknya adalah runtime JavaScript
- Pemrograman dilakukan dengan JavaScript



# FITUR NODE.JS

---

- Server-side JavaScript
- Asynchronous I/O
- Asynchronous Programming
- Dibangun berdasarkan anonymous function di JS
- Single thread
- Memiliki arsitektur event-driven

# MENGAPA NODE JS?

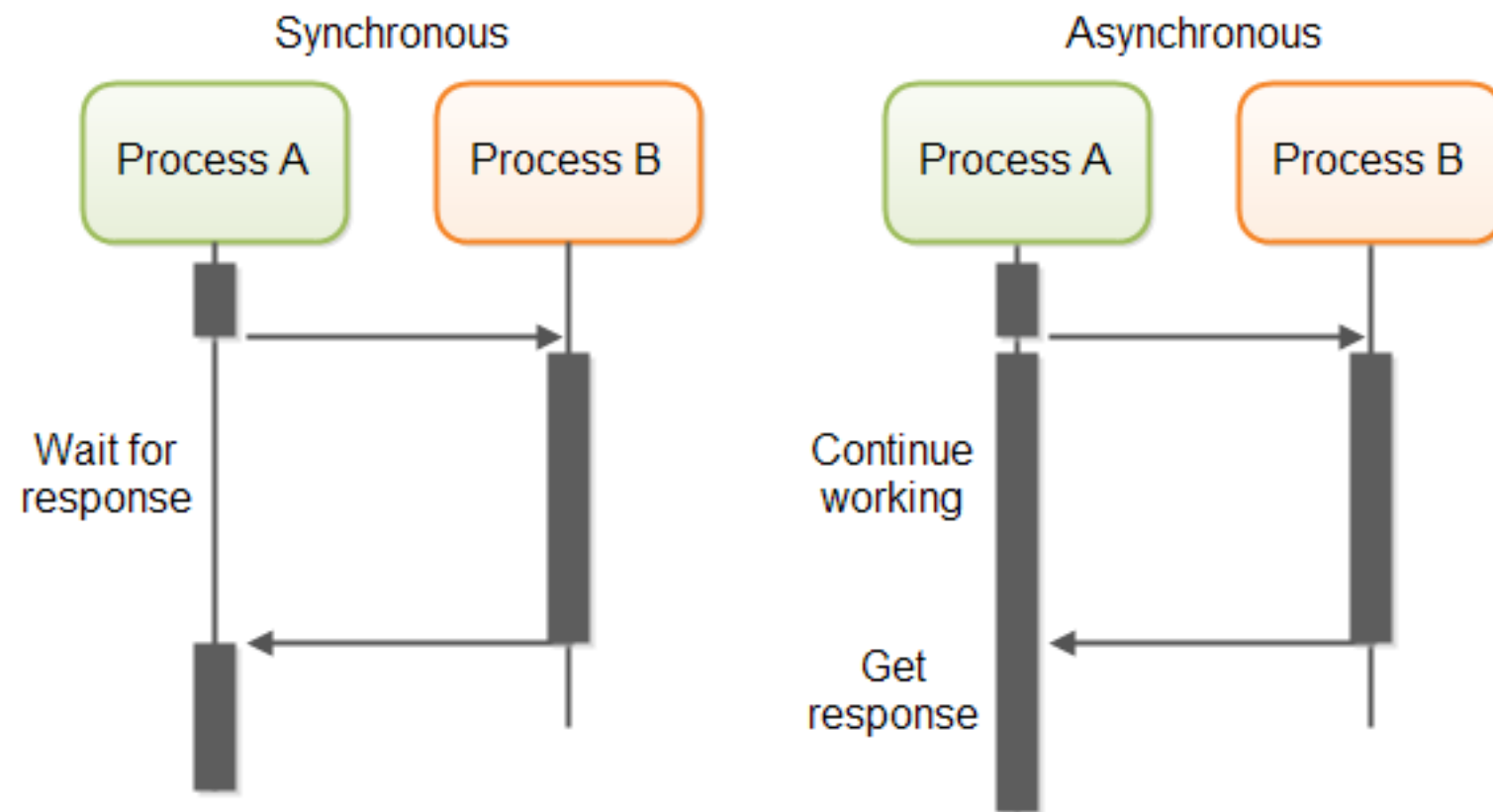
---

- Menghasilkan aplikasi server yang cepat dan tetap bisa menangani berbagai koneksi bersamaan
  - Berkat arsitektur asynchronous dan event-driven
- Memungkinkan back-end dan front-end ditulis dalam bahasa yang sama
  - Tim bisa bekerja sama dengan lebih cepat
- Perkembangannya signifikan dan cepat



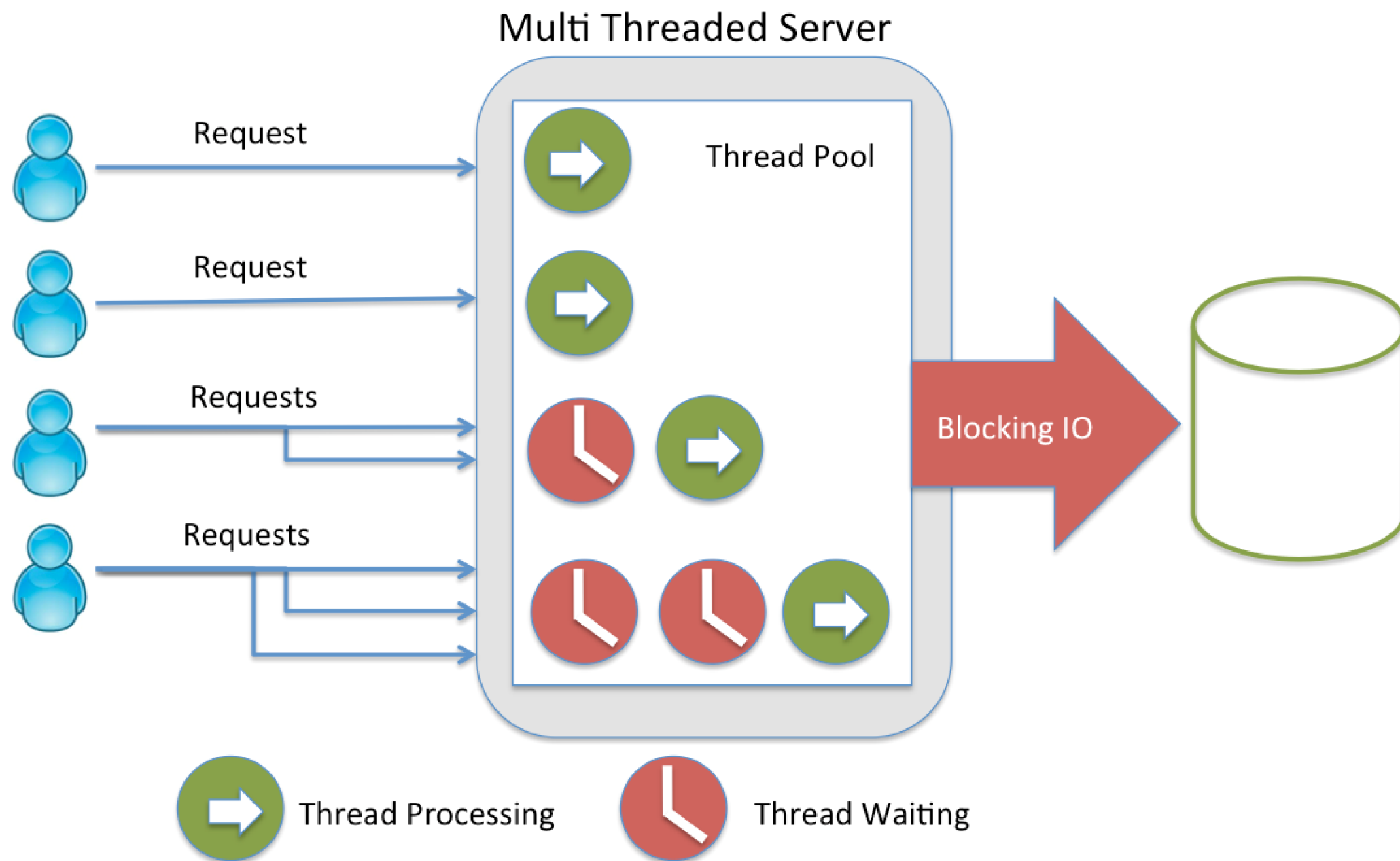
# ASYNCHRONOUS VS SYNCHRONOUS

---



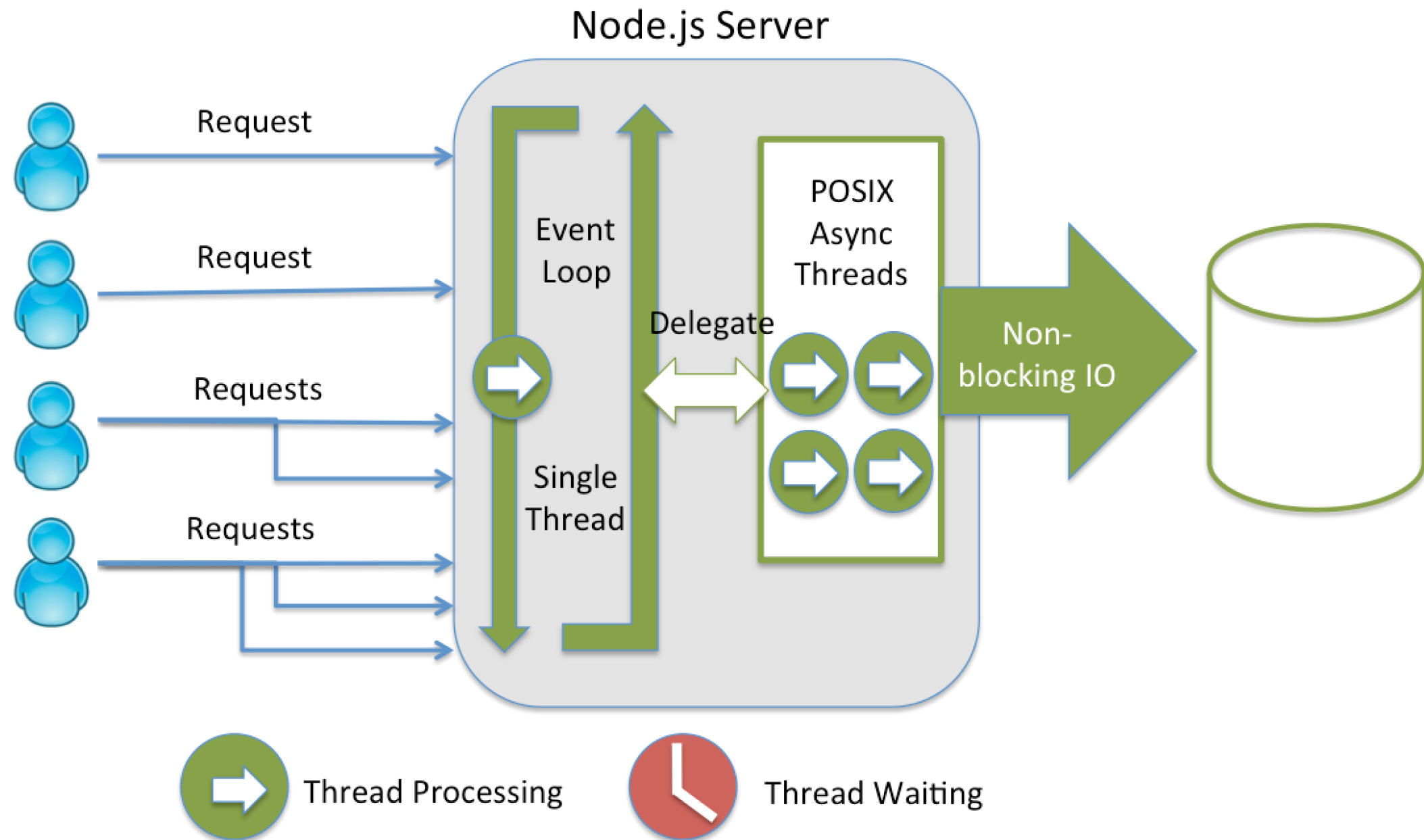
# ASYNCHRONOUS VS SYNCHRONOUS

---



# ASYNCHRONOUS VS SYNCHRONOUS

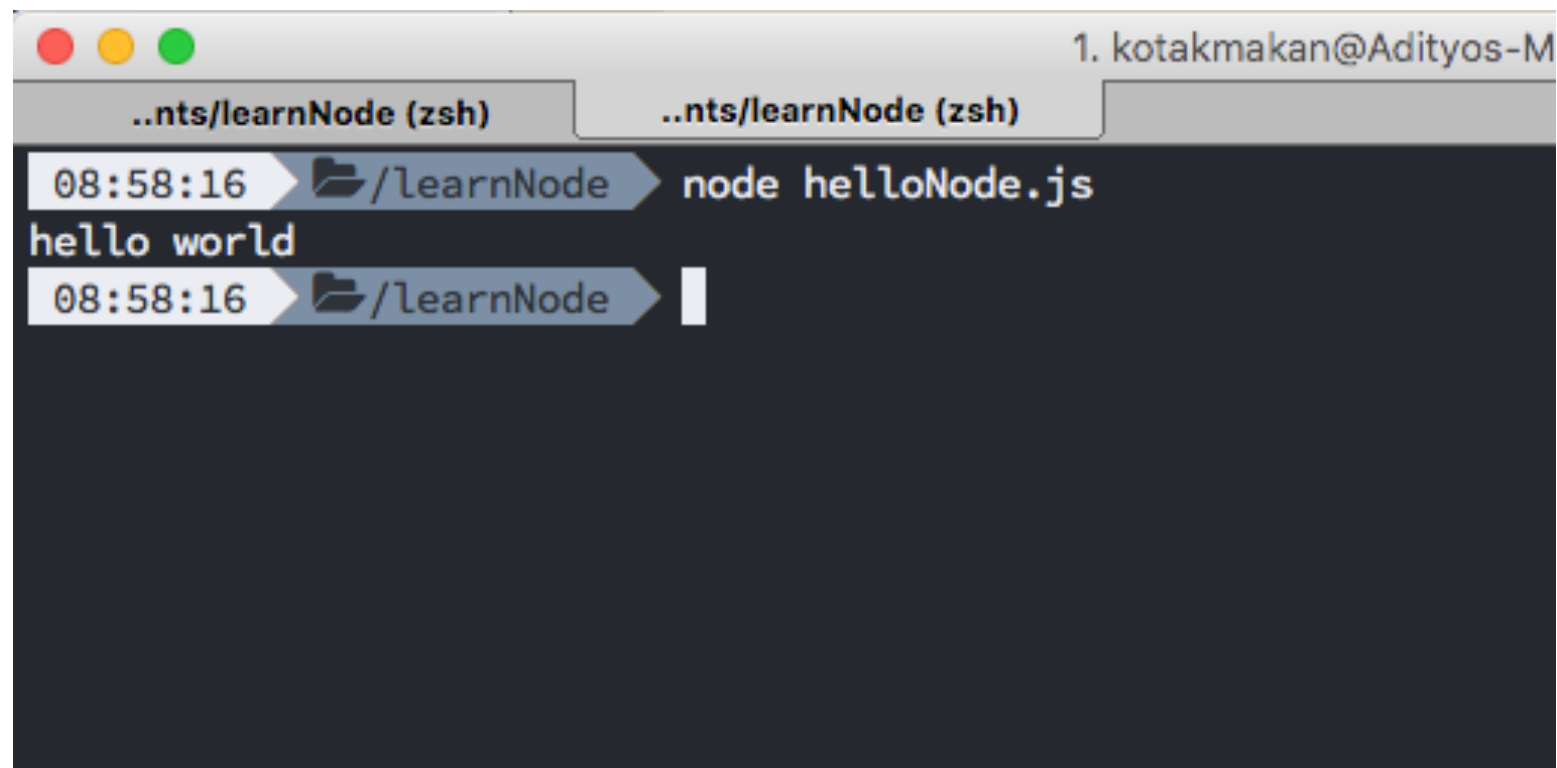
---



# MENGGUNAKAN NODE JS

---

- Penggunaan dilakukan via terminal
- Jalankan `node file.js`
- Node akan berjalan dan melakukan operasi JS di runtime environment

A screenshot of a macOS terminal window with a dark background. The window title bar shows three colored window control buttons (red, yellow, green) on the left and the text '1. kotakmakan@Adityos-M' on the right. Below the title bar, there are two tabs, both labeled '..nts/learnNode (zsh)'. The terminal content shows a command prompt '08:58:16' followed by a folder icon and '/learnNode', then the command 'node helloNode.js' is entered. The output 'hello world' is displayed on the next line. Below that, another command prompt '08:58:16' followed by a folder icon and '/learnNode' is shown with a white cursor character at the end.

```
08:58:16 /learnNode node helloNode.js
hello world
08:58:16 /learnNode
```

# MENGGUNAKAN NODE JS: HELLO WORLD

---

```
console.log("hello, world");
```

- Simpan sebagai hello.js
- Jalankan `node hello.js` di terminal
- Lihat outputnya
- Output hanya nampak di terminal, kita belum membicarakan browser

# KONSEP PEMROGRAMAN JS UNTUK NODE

---

- Saat ini, kita fokus pada paradigma imperative
- Mirip C
- Pemrograman di Node.js akan banyak menggunakan objek, serta anonymous function
- Style pemrograman perlu diubah untuk bisa memanfaatkan fitur asynchronous pada Node
- Kita berkenalan dulu dengan beberapa konsep kunci pemrograman di JS

# VARIABEL

---

- Tempat penyimpanan data, yang isinya bisa diganti-ganti selama program berjalan
- Variabel bisa berisi bilangan (bulat / desimal) atau huruf/kalimat
- Dalam JS, variabel diinisiasi dengan keyword `var` atau `let`
- Tidak ada tipe khusus yang perlu diinisiasi
- Contoh:

```
var nama = "Joni"
```

```
var x = 19;
```

```
let y = 13;
```

# SCOPE VARIABLE: VAR VS LET

---

- Sebuah variabel yang diinisiasi di luar blok manapun, disebut global variabel, dan bisa digunakan di manapun di dalam program
  - Berlaku baik untuk var dan let
- Variabel lokal memiliki perbedaan untuk var dan let
  - var berlaku di luar blok tempat ia diinisiasi
  - let hanya berlaku di dalam blok dan sub-blok tempat ia diinisiasi



# SCOPE VARIABLE: VAR VS LET

---

- Apakah outputnya?
- Coba ganti y di dalam kondisional if menjadi var, apakah perbedaan outputnya?
- Tampilkan juga isi dari variabel v dan x di ujung fungsi hitung(). Apakah outputnya?

```
var v = 30;
let x = 10;

function hitung() {
    var y = 6;
    if (x === 20) {
        let y = 60;
        console.log(y);
    }
    console.log(y);
}

hitung();
```

# FUNCTION

---

- Kumpulan statement, biasanya melakukan hal spesifik
- Function diinisiasi dengan keyword function
- Function juga bisa memiliki nilai kembali (return value) yang ditandai dengan keyword return
- Untuk digunakan, function cukup dipanggil namanya saja

```
function tambah() {  
    var x = 19;  
    var y = 11;  
    var hasil = x+y;  
    console.log(hasil);  
}
```

# FUNCTION: ARGUMEN

---

- Sebuah function juga bisa memiliki argumen, di mana berarti ia membutuhkan parameter tambahan untuk menjalankan fungsi tersebut
- Ketika fungsi ini dipanggil, ia juga harus menerima argumen

```
function buatAngka(x) {  
    var y = x + 1;  
    console.log(x+ " ditambah 1 sama dengan "+y);  
}
```

```
buatAngka(20);
```

# FUNCTION: ANONYMOUS

---

- Dalam JS, function juga diperlakukan seperti variabel, kita bisa membuat variabel berisi function
- Selanjutnya, untuk menggunakan function tersebut, kita cukup memanggil variabel yang berisi function tadi
- Function yang tidak memiliki nama ini, disebut anonymous function

```
var x = function (x, y) {return x + y};  
var z = x(13, 7);  
console.log(z);
```

# FUNCTION: FUNCTION SEBAGAI ARGUMEN

---

- Karena function juga bisa menjadi variabel, masuk akal jika function bisa digunakan sebagai argumen dari function lain
- Function yang membutuhkan function lain untuk berjalan, disebut sebagai higher-order function
- Function yang diimplementasikan sebagai argumen function lain, kemudian dikenal dengan nama callback function

```
function buatAngka(x) {  
    var y = x + 1;  
    console.log(x+ " + 1 = "+y);  
}  
  
function tesAngka(x, callback) {  
    callback(x);  
}  
  
tesAngka(10, buatAngka);
```

# FUNCTION: CALLBACK FUNCTION

---

- Callback function adalah sebuah fungsi yang berjalan di dalam fungsi lain
- Dalam hal ini, fungsi yang akan menjadi callback function menjadi sebuah argumen bagi fungsi lain
- Ini adalah salah satu konsep penting dalam pemrograman di Node.js
  - Untuk memenuhi konsep asynchronous single thread programming, dasarnya adalah callback function

# FUNCTION: CALLBACK FUNCTION

---

- Saat mendeklarasikan fungsi `tesAngka()`, kita berikan tempat bagi callback function
- Fungsi `buatAngka()` dipanggil di dalam fungsi `tesAngka()`

```
function buatAngka(x) {  
    var y = x + 1;  
    console.log(x+ " + 1 = "+y);  
}  
  
function tesAngka(x, callback) {  
    callback(x);  
}  
  
tesAngka(10, buatAngka);
```

# FUNCTION: CALLBACK FUNCTION DENGAN ANONYMOUS FUNCTION

---

- Callback function berguna saat kita ingin mengabstraksi sebanyak mungkin program kita, sehingga tidak menumpuk banyak operasi dalam 1 blok fungsi
- Ada kalanya, kita juga ingin menggunakan sebuah fungsi sebagai callback, yang berjalan sekali, tanpa perlu kita definisikan
- Untuk itu kita bisa menggunakan anonymous function sebagai callback



# FUNCTION: CALLBACK FUNCTION DENGAN ANONYMOUS FUNCTION

---

- Saat menggunakan callback function, kita mengoper keseluruhan function, bukan hanya hasilnya
- Penggunaan argumen diatur dalam definisi function yang memanggil callback function

```
function tesAngka(x, y, callback) {  
    callback(x, y);  
}
```

```
tesAngka (10, 34, function(n, m) {  
    var y = n + m;  
    console.log(n + " + " + m + " = "+ y);  
});
```

# ARRAY

---

- Sebuah variabel yang memuat sekumpulan data, serupa dengan himpunan
- Data yang dimuat tidak harus setipe
- Setiap data mendapatkan 1 index yang menunjukkan urutannya di dalam array

```
var perangkat = ["komputer", "printer", "monitor", "scanner"];
```

# MENGAKSES ARRAY

---

- Untuk mengakses sebuah elemen di dalam array, kita cukup memanggil nama array, diikuti dengan indeks elemen tersebut di dalam sebuah kurung siku
- Indeks dimulai dari angka 0

```
var perangkat = ["komputer", "printer", "monitor", "scanner"];  
perangkat[1] //berisi printer
```

```
for (let i=0; i<perangkat.length; i++) {  
    console.log(perangkat[i]);  
}
```

# METHOD BAWAAN ARRAY

---

- Array memiliki beberapa method bawaan yang bisa digunakan untuk berbagai keperluan
- Contoh, untuk menghitung ukuran array, maka kita bisa menggunakan method “length”

```
var perangkat = ["komputer", "printer", "monitor", "scanner"];  
perangkat.length //4
```

# MENGAkses ARRAY

---

- Untuk mengakses elemen array menggunakan loop, maka kita bisa menggunakan method `forEach` milik Array
- Method ini juga otomatis mengekspos indeks elemen array

```
var perangkat = ["komputer", "printer", "monitor", "scanner"];  
perangkat[1] //berisi printer
```

```
perangkat.forEach(function(item, index) {  
    console.log(item, index);  
});
```

# MENAMBAH DAN MENGURANGI ELEMEN DALAM ARRAY

---

- Untuk menambah elemen baru dalam array, kita gunakan method push
- Sebaliknya, untuk mengeluarkan elemen terakhir dalam array, kita gunakan pop

```
perangkat.push("keyboard");
```

```
for (let i=0; i<perangkat.length; i++) {  
    console.log(perangkat[i]);  
}
```

```
perangkat.pop();
```

```
for (let i=0; i<perangkat.length; i++) {  
    console.log(perangkat[i]);  
}
```

# MENAMBAH DAN MENGURANGI ELEMEN DALAM ARRAY

---

- Untuk menambah elemen baru dalam array yang akan masuk sebagai elemen pertama, kita gunakan method shift
- Sebaliknya, untuk mengeluarkan elemen pertama dalam array, kita gunakan unshift

```
perangkat.shift();
```

```
perangkat.forEach(function(item, index) {  
    console.log(item, index);  
});
```

```
perangkat.unshift("mouse");
```

```
perangkat.forEach(function(item, index) {  
    console.log(item, index);  
});
```

# MENGCOPY ARRAY

---

- Untuk mengcopy sebuah array menjadi array lain, maka kita gunakan method slice
- Slice bisa diberikan argumen untuk menandakan elemen mana saja yg ingin dicopy

```
var belanja = perangkat.slice();  
console.log(belanja); //berisi semua elemen array perangkat
```

```
var belanjaSaya = perangkat.slice(2,4);  
console.log(belanjaSaya); //berisi elemen 2-4 dari array perangkat
```



# OBJECT

---

- Object adalah sebuah struktur data sederhana, berupa kumpulan dari beberapa properti yang mendefinisikan suatu entitas
- Sebuah properti adalah pasangan nama (key) dan nilai (value)
- Properti juga bisa berupa function -> dikenal dengan nama method
- Semua di JS adalah object (termasuk array dan functions)

# MEMBUAT OBJECT DENGAN INITIALIZER

---

- Ada beberapa cara membuat objek
- Cara yang paling umum adalah dengan initializer
- Objek dibuat, diberikan nama beserta propertinya

```
var hewan = {  
  nama : "kukang",  
  kaki : 2,  
  tangan : 2,  
  makan : function() {  
    console.log("saya sedang makan");  
  }  
};
```

# MEMBUAT OBJECT DENGAN CONSTRUCTOR FUNCTION

---

- Cara lain adalah dengan constructor function
- Objek didefinisikan dengan function sebagai constructornya (berisi prototype dari objek tersebut)
- Objek lalu dibuat dengan keyword new
- This digunakan untuk mengakses properti milik prototype

```
function Hewan(_nama, _kaki, _makanan) {  
    this.nama = _nama;  
    this.kaki = _kaki;  
    this.makanan = _makanan;  
}
```

```
var sapi = new Hewan("sapi", 4, "rumput");
```

# MEMBUAT OBJECT DENGAN CONSTRUCTOR FUNCTION

---

- Alternatif menggunakan constructor, adalah dengan langsung mengassign constructor function ke sebuah variabel
- Namun untuk membuat object baru, kita perlu menginisiasi variabel tadi menggunakan new, baru kemudian setiap properti diisi

```
var hewanSaya = function Hewan(_nama, _kaki, _makanan) {  
    Hewan.nama = _nama;  
    Hewan.kaki = _kaki;  
    Hewan.makanan = _makanan;  
}
```

```
hewanSaya("laba-laba", 8, "serangga");
```

```
var sapi = new hewanSaya();  
sapi.kaki = 4;
```

# MENGAKSES PROPERTI OBJECT

---

- Properti sebuah object bisa diakses dengan menggunakan titik setelah nama object tersebut
- Atau, bisa juga dengan menggunakan kurung siku, seperti mengakses elemen array

```
var hewanSaya = function Hewan(_nama, _kaki, _makanan) {  
    Hewan.nama = _nama;  
    Hewan.kaki = _kaki;  
    Hewan.makanan = _makanan;  
}
```

```
hewanSaya("laba-laba", 8, "serangga");  
console.log (hewanSaya.nama);  
console.log (hewanSaya["kaki"]);
```

# MENGGANTI PROPERTI OBJECT

---

- Setelah diakses, maka nilai sebuah properti objek bisa diganti, seperti halnya mengganti isi sebuah variabel

```
var hewanSaya = function Hewan(_nama, _kaki, _makanan) {  
    Hewan.nama = _nama;  
    Hewan.kaki = _kaki;  
    Hewan.makanan = _makanan;  
}
```

```
hewanSaya("laba-laba", 8, "serangga");  
hewanSaya.nama = "kuda"  
console.log(hewanSaya);
```

# METHOD

---

- Method adalah function yang menjadi properti sebuah objek
- Ia bisa digunakan sebagaimana mengakses properti lain milik objek
- Penggunaannya juga sama dengan function

# METHOD

---

```
function makan(makanan) {  
  this.jenisMakanan = makanan;  
}
```

```
function lari(jarak){  
  this.perpindahan += jarak;  
}
```

```
function binatang (nama, kaki) {  
  this.nama = nama;  
  this.kaki = kaki;  
  this.diet = makan;  
  this.perpindahan = 0;  
  this.lari = lari;  
}
```

```
var labaLaba = new binatang("laba-laba", 8);  
labaLaba.lari(20);  
labaLaba.diet("serangga");  
console.log(labaLaba.perpindahan);  
console.log(labaLaba.jenisMakanan);
```



# METHOD

---

```
var hewan = {  
  nama : "kukang",  
  kaki : 2,  
  tangan : 2,  
  makanan : "serangga",  
  makan : function() {  
    console.log("saya sedang makan " + this.makanan);  
  }  
};  
  
hewan.makan();
```

# ARRAY DAN OBJECT

---

- Sebagai sebuah model data, kita bisa membuat array berisi object
- Ini mirip dengan yang dimiliki oleh sebuah file JSON

```
var menu = [  
  {  
    nama: "nasi goreng",  
    harga: 17000,  
    pesan: false  
  },  
  {  
    nama: "rawon",  
    harga: 20000,  
    pesan: false  
  },  
  {  
    nama: "sate padang",  
    harga: 15000,  
    pesan: false  
  }  
];  
  
menu[0].nama; // nasi goreng
```

# FILTER

---

- Method bawaan Array yang bisa kita gunakan untuk membuat array baru dengan cara memfilter array lama

```
var hargaMakanan = [30000, 45000, 10000];
```

```
var hargaSekarang = hargaMakanan.filter(function(element){  
    return element > 10000;  
})
```

```
console.log(hargaSekarang);
```

# MAP

---

- Melakukan fungsi pada masing-masing elemen array,
- Hasilnya adalah array baru

```
var harga = [10000, 30000, 15000];
```

```
var hargaBaru = harga.map(function(element){  
    return element*2;  
})
```

```
console.log(hargaBaru);
```

# REDUCE

---

- Melakukan fungsi pada semua elemen array
- Hasilnya digabungkan menjadi 1 nilai

```
var angka = [8, 9, 12];  
  
var total = angka.reduce(function(total, num){  
    return total+num  
}, 0);  
  
console.log(total);
```

# MAP DAN FILTER

---

- Ketiganya bisa digabungkan
- Misal memfilter menu tertentu untuk kemudian diubah nilainya
- Dalam hal ini, penggunaan anonymous function bisa membuat kode menjadi lebih bersih

# MAP DAN FILTER

---

```
var menuHematKami = buatHargaBaru(menu);

function buatHargaBaru(harga) {
  return harga.filter(function(element) {
    return element.harga > 15000;
  }).map(function(element) {
    var hargaMakananBaru = element.harga - 10000;
    var menuHemat = element.nama;
    var daftarMenuHemat = {
      nama: menuHemat,
      harga : hargaMakananBaru
    }
    return daftarMenuHemat;
  })
}

console.log(menuHematKami);
```