

## COURSE 1

# Belajar membuat tampilan web dengan HTML + CSS (Sisi Frontend)

### Material Details

---

---

01 Why Fullstack ? Why now?

---

02 Hello HTML

---

03 CodeSandbox + Element

---

04 How to be a Good Engineer

---

05 Attributes

---

06 Formatting & Form

---

07 HTML 5

---

08 Week 2

---

## **Modul Sekolah Fullstack Cilsy**

Hak Cipta © 2020 PT. Cilsy Fiolution Indonesia

Hak Cipta dilindungi Undang-Undang. Dilarang memperbanyak atau memindahkan sebagian atau seluruh isi buku ini dalam bentuk apapun, baik secara elektronis maupun mekanis, termasuk mecropy, merekam atau dengan sistem penyimpanan lainnya, tanpa izin tertulis dari Penulis dan Penerbit.

Penulis : Rizal Rahman, Muhammad Fakhri Abdillah

Editor : Muhammad Fakhri Abdillah

**Revisi Batch 3**

Penerbit : **PT. Cilsy Fiolution Indonesia**

Web Site : <https://cilsyfiolution.com> , <https://sekolahfullstack.com>

### **Sanksi Pelanggaran Pasal 113 Undang-undang Nomor 28 Tahun 2014 tentang Hak Cipta**

1. Setiap orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam pasal 9 ayat (1) huruf i untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan atau pidana denda paling banyak Rp100.000.000,00 (seratus juta rupiah).
2. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak cipta melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf c, huruf d, huruf f, dan atau huruf h, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah)
3. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf a, huruf b, huruf e, dan atau huruf g, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan atau pidana denda paling banyak Rp1.000.000.000,00 (satu miliar rupiah)
4. Setiap orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan atau pidana denda paling banyak Rp4.000.000.000,00 (empat miliar rupiah)



## Daftar Isi

Daftar Isi.....	3
1. Fundamental Preparation: Why Fullstack? Why Now?.....	5
1.1. Learning Outcomes.....	5
1.2. Outline Materi.....	5
1.3. Roadmap Pembelajaran.....	5
1.3.1. Diagram Roadmap Pembelajaran.....	6
1.4. Tren Industri.....	7
1.4.1. Era Serba Online dan Aplikasi.....	7
1.5. Mengenal Dunia Developer.....	9
1.5.1. Seperti Apa Sih Pekerjaan Frontend Developer ?.....	11
1.5.2. Apa itu Fullstack Developer?.....	14
1.5.3. Bagaimana Gambaran Kerjanya ?.....	14
1.5.4. Enaknya Menjadi Fullstack Developer.....	17
1.6. Why JavaScript?.....	20
1.6.1. Alasan Dari Sisi Bisnis.....	20
1.6.2. Alasan Dari Sisi Teknis.....	23
1.7. Skill yang Harus Dimiliki Seorang Fullstack Developer.....	25
1.8. Peluang Karir Fullstack Developer JS.....	28
1.9. How to be a Good Developer.....	31
1.9.1. Cara Bertanya yang Baik.....	32
1.9.2. Asking and Googling.....	35
1.9.3. Cara Belajar yang Baik.....	38
1.10. Membangun Mindset Seorang Programmer.....	41
1.11. Pengenalan Flowchart dan Algoritma Pemrograman.....	42
1.11.1. Pedoman-Pedoman Dalam Membuat Flowchart.....	43
1.11.2. Exercise.....	45
1.11.3. Mengenal Algoritma Pemrograman.....	45



1.12. Komponen Struktur Algoritma Pemrograman.....	48
1.12.1. Sequence.....	48
1.12.2. Selection.....	49
1.12.3. Loop.....	50
1.12.4. Exercise.....	50

# 1.

## Fundamental Preparation: Why Fullstack? Why Now?

### 1.1. Learning Outcomes

Setelah selesai mempelajari bab ini, peserta mampu :

1. Mengetahui apa itu Fullstack Developer.
2. Mengetahui masalah apa saja yang dihadapi Fullstack Developer.
3. Gambaran dan Karir seorang Fullstack Developer.
4. How to be a good Developer.

### 1.2. Outline Materi

1. Roadmap Pembelajaran
2. Tren Industri
3. Mengetahui Dunia Fullstack Developer
4. Mengetahui Framework React.JS dan Node.JS
5. How to be a Good Engineer

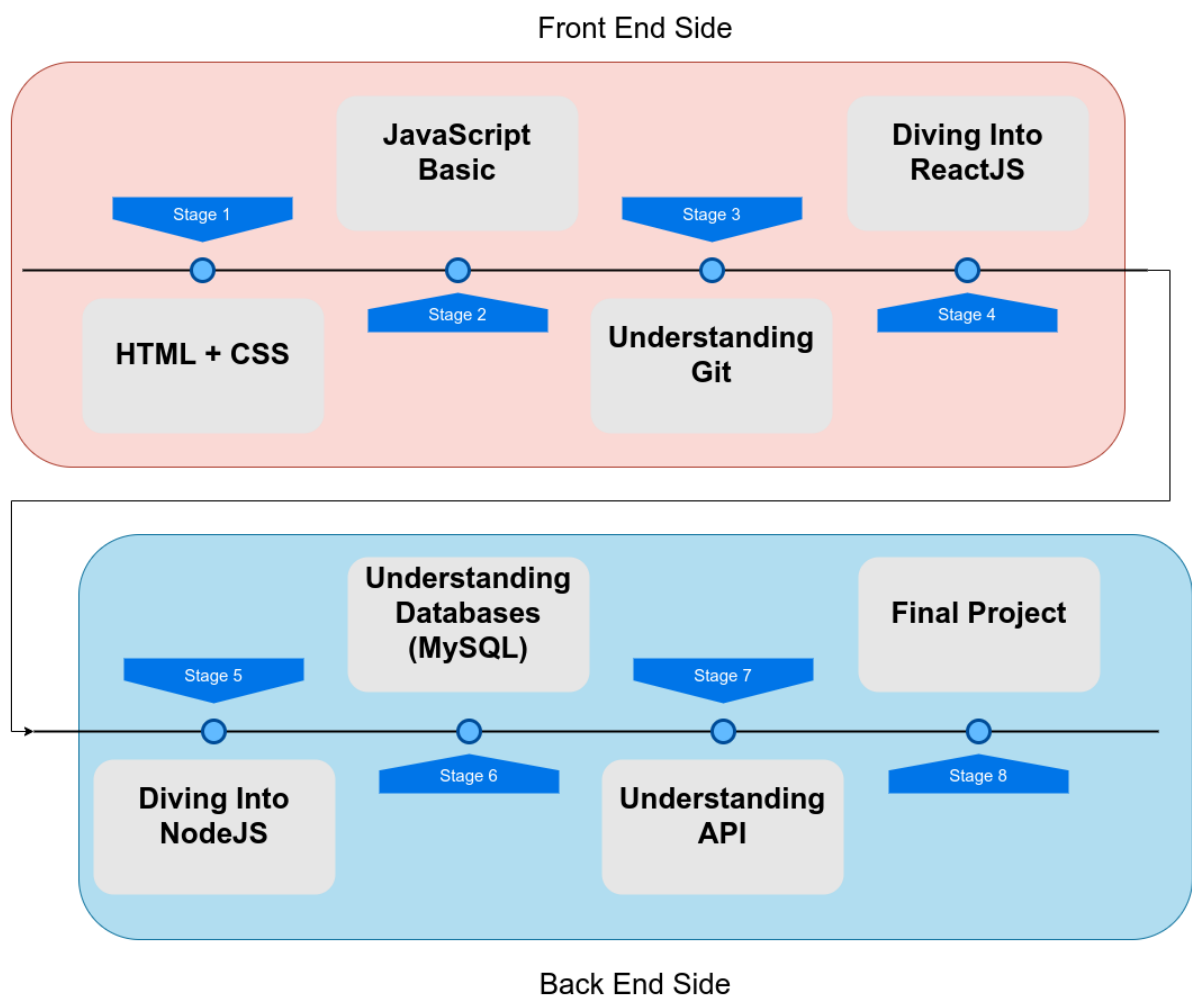
### 1.3. Roadmap Pembelajaran

Sebelum memulai pembelajaran di Sekolah Fullstack Cilsy, ada baiknya kita mengenal terlebih dahulu roadmap atau alur pembelajaran yang akan kita jalani selama masa pembelajaran 4 bulan kedepan. Diharapkan kita bisa mengetahui goal apa yang akan kita capai setelah menyelesaikan semua pembelajaran dan mengetahui sudah sampai tahap mana kita dalam proses pembelajaran yang berlangsung.

Roadmap disini akan ditampilkan dalam bentuk diagram proses, dan dijelaskan kembali nanti disetiap point pembelajaran yang akan kita jalani. Sehingga terus terpantau alur pembelajarannya.

### 1.3.1. Diagram Roadmap Pembelajaran

Berikut merupakan diagram roadmap dari pembelajaran yang akan kita jalani :



## 1.4. Tren Industri

### 1.4.1. Era Serba Online dan Aplikasi

Semua bermula dari dimulainya era internet dan aplikasi. Dimana dalam 2 dekade terakhir perkembangan internet sangat pesat hingga setiap orang di zaman ini rasanya sangat ketergantungan dengan keberadaan internet, apalagi beberapa tahun kebelakang hingga saat ini hampir semua perusahaan, organisasi, bisnis, instansi pemerintahan, startup dari yang “Unicorn” hingga startup kecil buatan mahasiswa berlomba-lomba membuat produk-produk berbasis Aplikasi. Bahkan tak hanya produk yang tampil langsung ke pengguna, untuk kegiatan internal perusahaan pun sudah mulai dibuat serba online dan menggunakan aplikasi untuk meningkatkan produktifitas mereka. Sebegitu pentingnya, hingga kini pemanfaatan internet dan Aplikasi untuk perusahaan bukan lagi barang yang “nice to have” saja, tetapi sudah menjadi kebutuhan utama agar perusahaan dapat menjadi pemenang di industri mereka masing-masing.

Dewasa ini masyarakat semakin dimudahkan dengan munculnya aplikasi online yang menjawab problematika yang ada di lapangan. Banyak sekali perusahaan-perusahaan yang berlomba menawarkan solusi atas problem masyarakat. Sebutlah aplikasi transportasi online seperti GoJek, Grab, Uber, dll yang saat ini hampir semua pengguna smartphone pasti pernah menggunakannya dan menjadi game-changer company.



*Aplikasi Transportasi Online yang Semakin Terintegrasi*

Orang-orang yang dulunya apabila ingin bepergian menggunakan transportasi seperti ojek, taxi, dll yang harus pergi ke pangkalan ojek terdekat, dan terkadang dengan harga yang cenderung “seenaknya” supir ojek, sekarang sudah dimanjakan dengan aplikasi transportasi online yang sangat mudah, tinggal tentukan daerah jemput dan daerah tujuan, lalu sistem secara otomatis akan menghitung tarif yang harus dibayar. Selain mudah, konsumen pun akan mendapatkan rasa aman karena driver transportasi online tentunya harus terdaftar secara legal di perusahaan tersebut. Selain memudahkan dalam transportasi, saat ini malah aplikasi-aplikasi tersebut sudah terintegrasi dengan sistem pemesanan makanan, sehingga apabila konsumen malas mencari makanan di luar, tinggal panggil via aplikasi, dan makanan yang diinginkan pun akan diantarkan ke tempat Anda.

Tren aplikasi online ini yang diprediksi akan terus tumbuh hingga 10-20 tahun ke depan. Dalam tren membuat berbagai aplikasi akan terus diluncurkan, tentunya ada sumber daya yang diperlukan, salah satunya adalah **Software Developer**.

Semua aplikasi ini dibuat oleh Software Developer, sehingga jangan kaget betapa banyak lowongannya berseliweran. Software Developer juga merupakan salah satu karir yang hampir mustahil terkena dampak tergantikannya tenaga manusia dengan robot. Karir yang cukup aman untuk digeluti 10-20 tahun ke depan.



<https://willrobotstakemyjob.com/15-1132-software-developers-applications>

Inilah kenapa semakin hari semakin banyak orang berlomba-lomba ingin menjadi Software Developer. Tren ini pun kami yakin akan terus berlanjut bahkan hingga tahun 2030-2040.



Bagaimana, apakah Anda sudah siap memulai karir yang lebih baik dengan menjadi seorang Software Developer? :)

## 1.5. Mengenal Dunia Developer

Pada bagian awal ini, kita akan belajar memahami terlebih dahulu bahwa dalam pembuatan sebuah aplikasi, Software Developer dibagi ke dalam 2 jenis. Yaitu Backend Developer dan Frontend Developer.

Backend Developer adalah orang yang ngoding untuk sisi sistem. Sedangkan Frontend Developer adalah orang yang ngoding untuk sisi tampilan / visual. Maksudnya bagaimana ?

Anggaplah kita ingin membuat sebuah fitur registrasi dan login user seperti web Facebook seperti berikut :

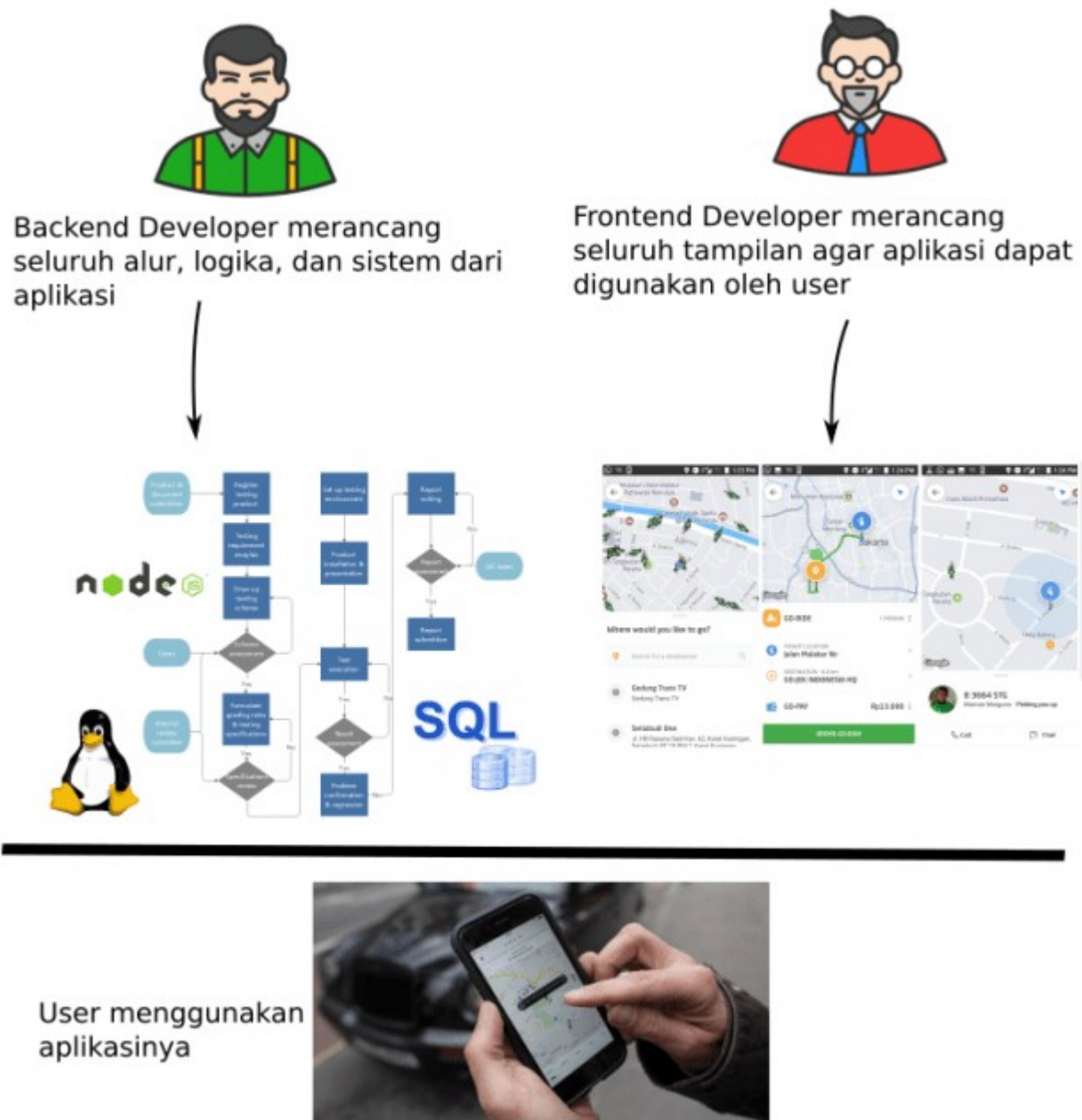


Backend Developer akan ngoding dari sisi : bagaimana data-data penting yang di isi ketika user registrasi, berhasil tersimpan dengan baik di database. Juga memastikan ketika data username dan password sudah cocok dengan yang ada di database, maka user diperbolehkan login, jika tidak cocok maka tidak boleh login.

Sedangkan Frontend Developer akan ngoding dari sisi : bagaimana menampilkan seluruh komponen web tersebut persis sesuai desain / gambar. Tekstnya muncul di tempat yang



tepat, form registrasinya muncul di tempat yang tepat, tombol “Create an Account” muncul dengan warna dan di tempat yang tepat hingga bagaimana ketika tombol diklik akan ada animasi loading.



Ibarat mobil, kita bisa analogikan Backend Developer adalah para insinyur yang bertugas mendesain dan membuat mesin dari motor tersebut. Sedangkan Frontend Developer adalah yang bertugas mendesain dan memasang body mobil, lampu, jok, dll.





Backend Developer merancang mesin agar mobil bisa berjalan



Frontend Developer merealisasikan body, kaca, lampu, dll agar mobil bisa tampil indah di mata user

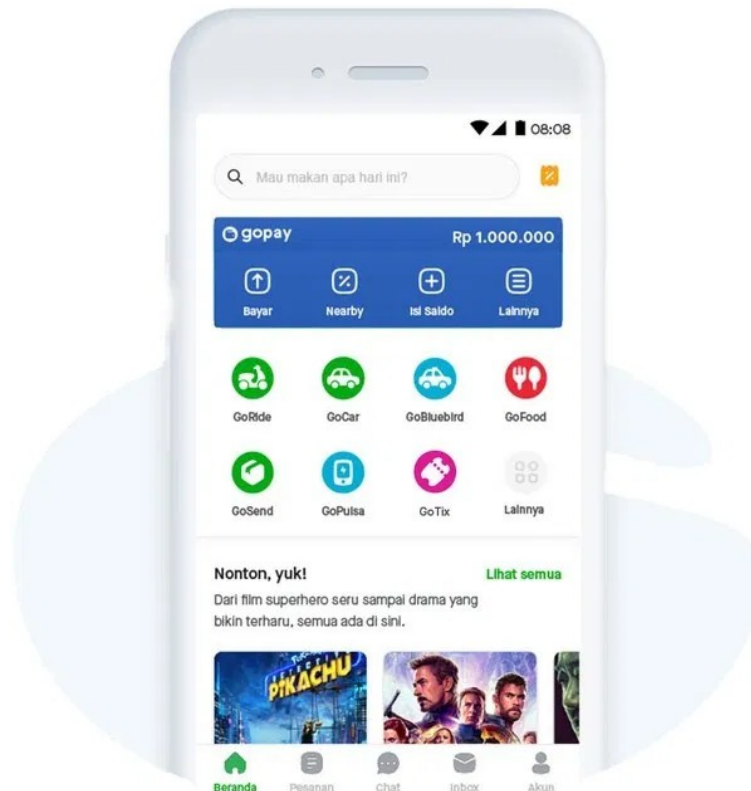


### 1.5.1. Seperti Apa Sih Pekerjaan Frontend Developer ?

Pada prakteknya Frontend Developer akan bekerjasama dengan Desainer ( atau sering disebut sebagai UI/UX Designer ).

Pertama-tama biasanya UI/UX Designer yang bertugas meriset seperti apa tampilan sebuah aplikasi yang paling nyaman dan mudah digunakan oleh user. Hasil riset tersebut mereka tuangkan dalam bentuk desain seperti berikut :

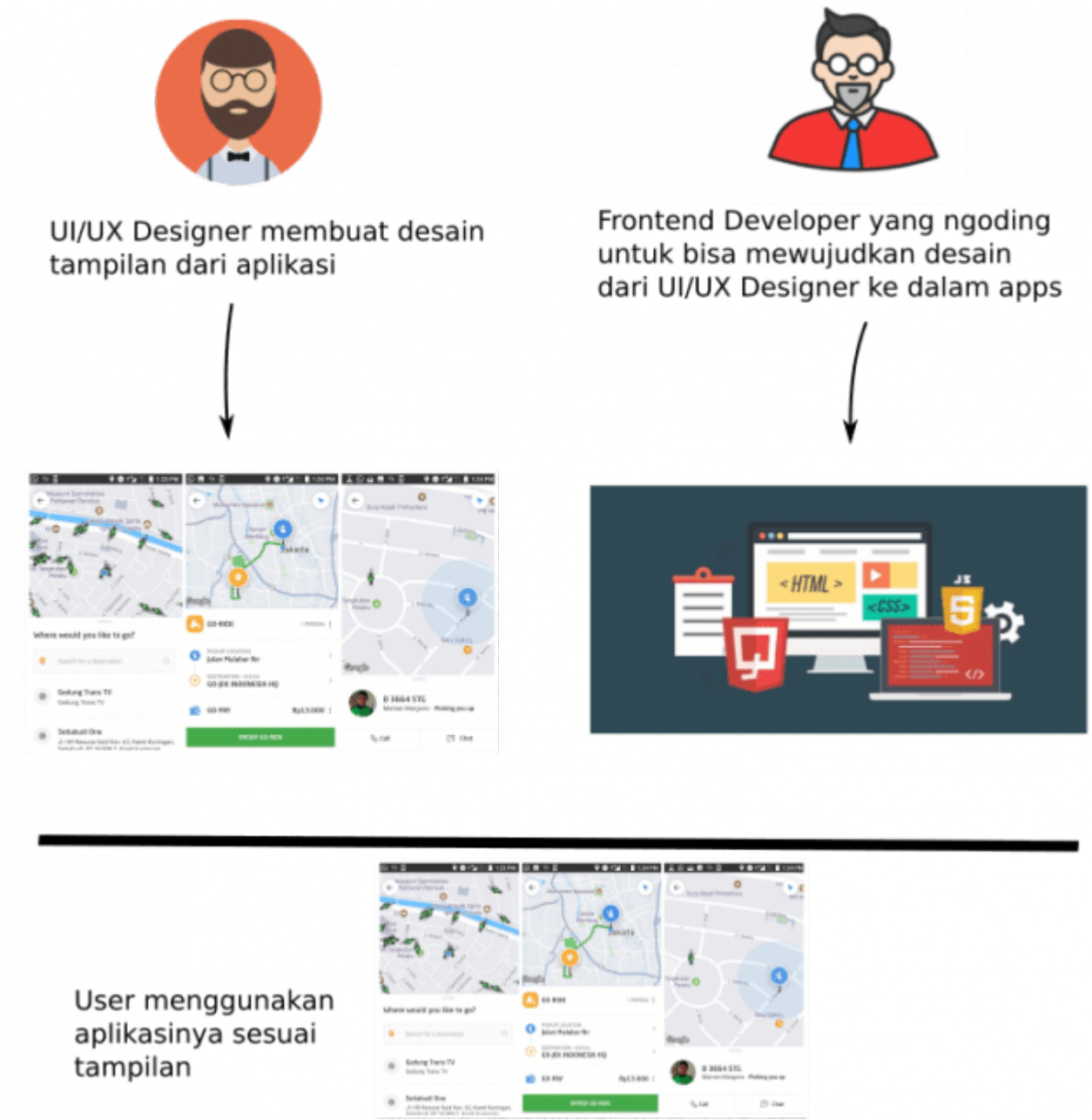




Dari situ tugas Frontend Developer adalah bagaimana mengkonversi desain tersebut menjadi codingan dan dapat benar-benar berfungsi di aplikasinya. UI/UX Designer juga harus sering koordinasi terlebih dahulu dengan Frontend Developer, agar tidak terjadi perbedaan antara desain yang dibuat dengan tampilan real di aplikasi.

Misalnya ketika UI/UX Designer ingin ketika User menekan tombol Daftar, tombol tersebut akan beranimasi seperti melompat-lompat. Atau ketika UI/UX Designer ingin ketika user mengetik chat dan mengirim chat, benar-benar realtime seperti di Whatsapp ( tidak ada loading atau reload halaman sama sekali ).



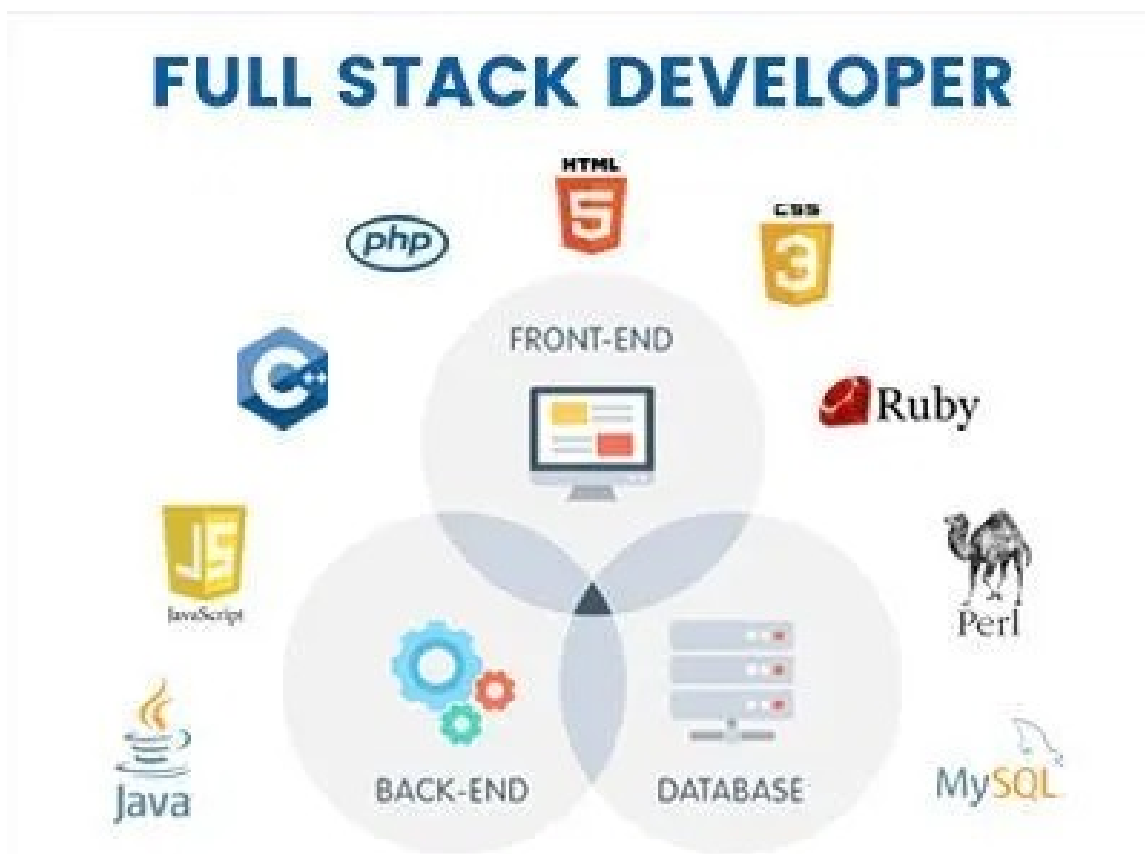


Hal-hal di atas sangat mudah dibuat secara desain dan konsep. Dan tentunya sangat bagus untuk diterapkan. Tetapi menuangkannya kedalam bentuk codingan itu beda cerita. Bisa jadi sangat sulit !

Disinilah tantangan bagi Anda sebagai seorang developer. Anda harus bisa memilih teknologi, teknik, maupun framework apakah yang tepat untuk bisa mewujudkan desain, animasi, dan kebutuhan yang bagus-bagus tersebut.

### 1.5.2. Apa itu Fullstack Developer?

Fullstack Developer adalah 1 orang Programmer gabungan dari Backend dan Frontend Developer. Mereka jago membangun sistem / fungsi dari aplikasi, dan jago pula dalam membuat tampilannya. Intinya hanya dengan 1 orang Fullstack Developer ini harusnya sudah bisa membuat 1 buah working apps. Mau bikin aplikasi / fitur apapun, Fullstack Developer bisa membuatnya sampai benar-benar jadi dan berfungsi.



Inilah yang membuat Fullstack Developer laris manis di industri. Khususnya Startup. Dengan meng-hire Fullstack Developer, maka dapat lebih menghemat biaya tapi tetap bisa membuat aplikasi yang berkualitas dengan cepat.

### 1.5.3. Bagaimana Gambaran Kerjanya ?

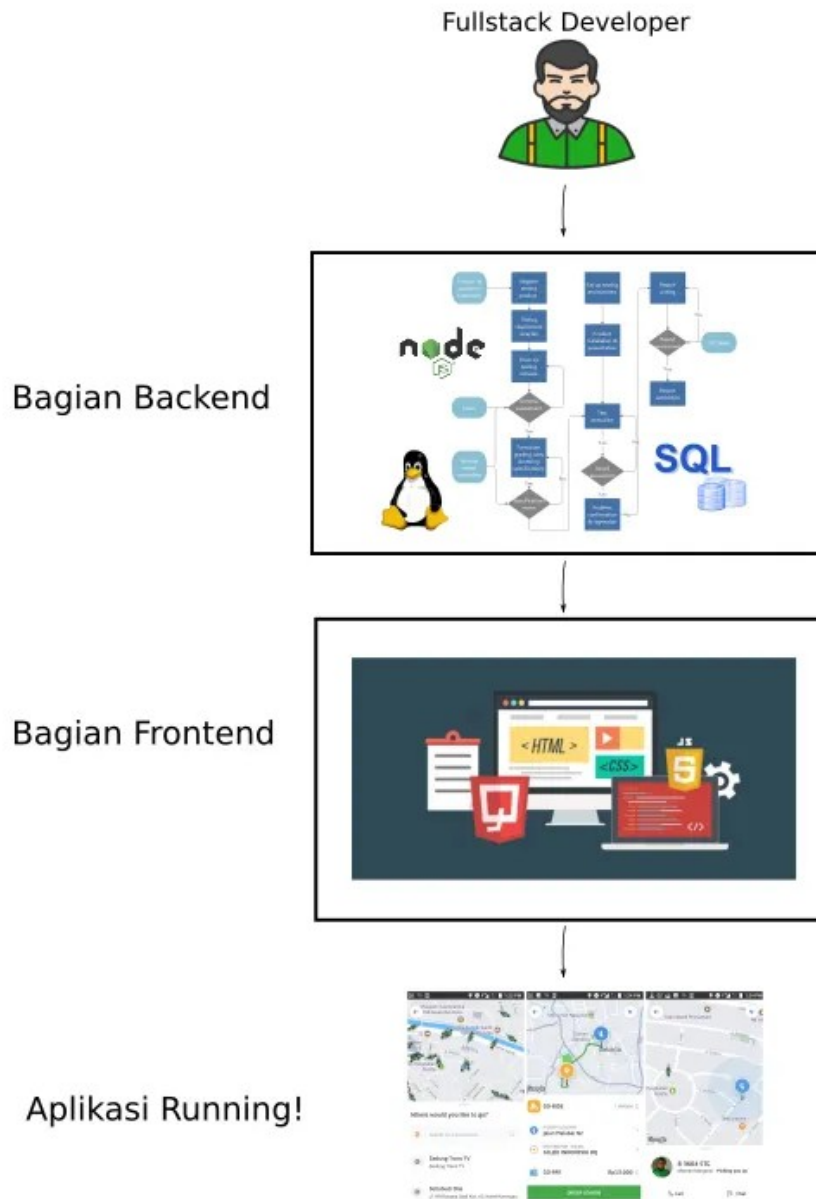
Secara sederhana, gambaran kerja dari Fullstack Developer adalah seperti berikut :



1. Setiap minggunya sudah ditentukan akan ada pembuatan fitur apa. Misalnya pembuatan fitur Login, Registrasi, dan fitur Review Pelanggan. Biasanya Fullstack Developer dalam tim akan dibagi-bagi pekerjaannya berdasarkan fitur. Misalnya 1 orang mengerjakan fitur Login, 1 orang mengerjakan fitur Registrasi, 1 orang mengerjakan fitur Review.
2. Selanjutnya Fullstack Developer akan mengerjakan bagian sistem / fungsinya terlebih dahulu. Membuat alur databasenya seperti apa, alur logikanya seperti apa, dll. Intinya disini mengerjakan pekerjaannya Backend Developer dulu.
3. Setelah sistem dan fungsi sudah jadi, maka Fullstack Developer baru membuat tampilan visual fiturnya sesuai desain yang ada. Memastikan tombolnya sudah muncul, layoutnya sudah sesuai, dan lain-lainnya. Pokoknya sampai jadi tampilan visualnya. Kerjanya Frontend Developer banget.

Tapi di tahap ini Fitur masih belum berfungsi. Visual aplikasinya belum nyambung dengan sistem yang sudah dibuat.

4. Terakhir Fullstack Developer akan memasukkan fungsi / sistem yang telah dibuat ke dalam komponen tampilan visualnya. Misalnya memastikan tombol Login sudah bisa diklik, dan benar berfungsi sesuai yang diinginkan bahwa ketika Username & Password benar, maka pengguna diijinkan masuk.



Gambaran kerja Fullstack Developer

Jika Anda lihat, intinya Fullstack Developer memang mengerjakan 2 jenis pekerjaan dari Backend Developer & Frontend Developer sekaligus.

Mungkin sempat terfikir dalam benak Anda, ***“Wah rugi di saya dong, kerjaan saya untuk 2 orang tapi dibayar hanya buat 1 orang ?”***





Eits, disini Anda harus bedakan **jenis pekerjaan** dengan **beban kerja** ya.

Jenis pekerjaannya memang 2, tapi secara **beban kerja tidak ada bedanya**. Bukan berarti Backend Developer bekerja 8 jam / hari, Frontend Developer bekerja 8 jam / hari, lalu ketika Anda jadi Fullstack Developer harus bekerja 16 jam / hari. Anda tetap bekerja 8 jam / hari kok. Jadi tetap fair.

Lalu kenapa harus jadi Fullstack Developer dibanding fokus menjadi Backend atau Frontend Developer ?

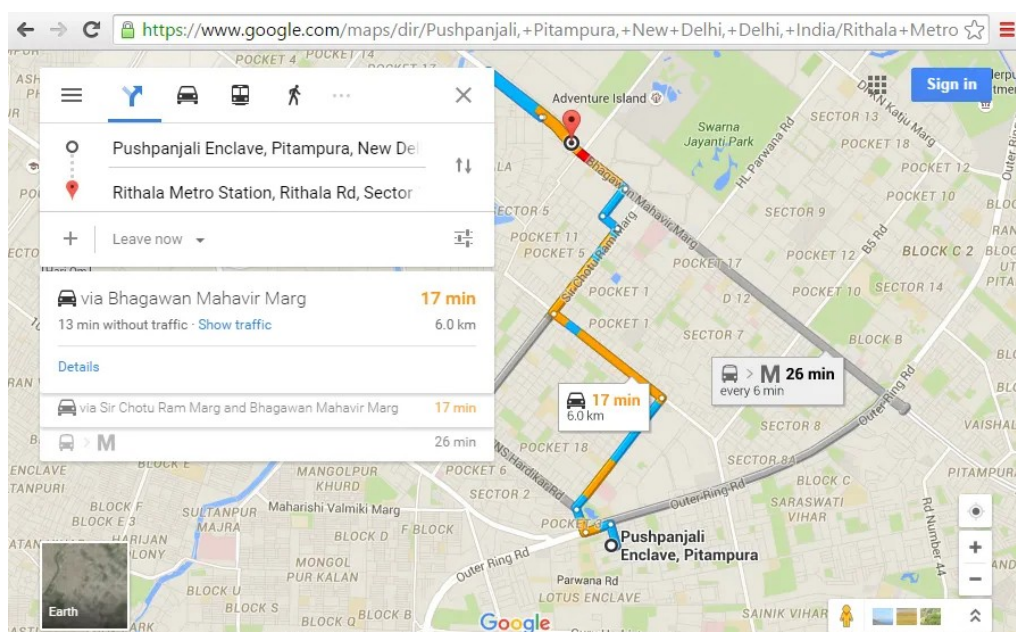
### 1.5.4. Enaknya Menjadi Fullstack Developer

Jika Anda benar-benar seorang pemula baru mau terjun ke dunia Programmer, maka saran saya jauh lebih baik untuk menjadi Fullstack Developer terlebih dahulu. Kenapa ?

#### 1. Paham Konsep Membangun Aplikasi

Syarat utama seseorang bisa jago sesuatu adalah paham betul apa yang dikerjakan. Dan untuk bisa paham apa yang dikerjakan, Anda perlu terjun langsung untuk melihat konsepnya secara keseluruhan.

Ibaratnya jika Anda mau pergi ke suatu tempat yang belum Anda ketahui, jauh lebih mudah jika Anda melihat Google Maps secara zoom out dulu bukan ? Sehingga Anda tau gambaran besar bagaimana kondisi jalannya, arah perjalanannya kemana, dll.



Dengan paham bagaimana membangun dari sisi sistem, sisi tampilan, dan bagaimana cara menggabungkannya hingga menjadi fitur / aplikasi yang berfungsi dengan baik maka Anda akan mempunyai skill fundamental bagus sekali. Dibandingkan dengan rekan Anda yang tidak pernah terjun menjadi Fullstack Developer terlebih dahulu.

## **2. Lebih Mudah Mencari Peluang Kerja**

Ini sudah tidak usah ditanyakan lagi. Lowongan Fullstack Developer jauh lebih banyak dibanding Backend / Frontend Developer. Simpel, karena Fullstack Developer banyak dibutuhkan tidak hanya di perusahaan besar tapi juga di Startup. Sedangkan Backend / Frontend Developer cenderung hanya dibutuhkan di perusahaan besar saja. Jadi Anda akan lebih cepat dapat kerja, lebih cepat memperbanyak pengalaman, lebih cepat untuk nantinya bisa naik karir.

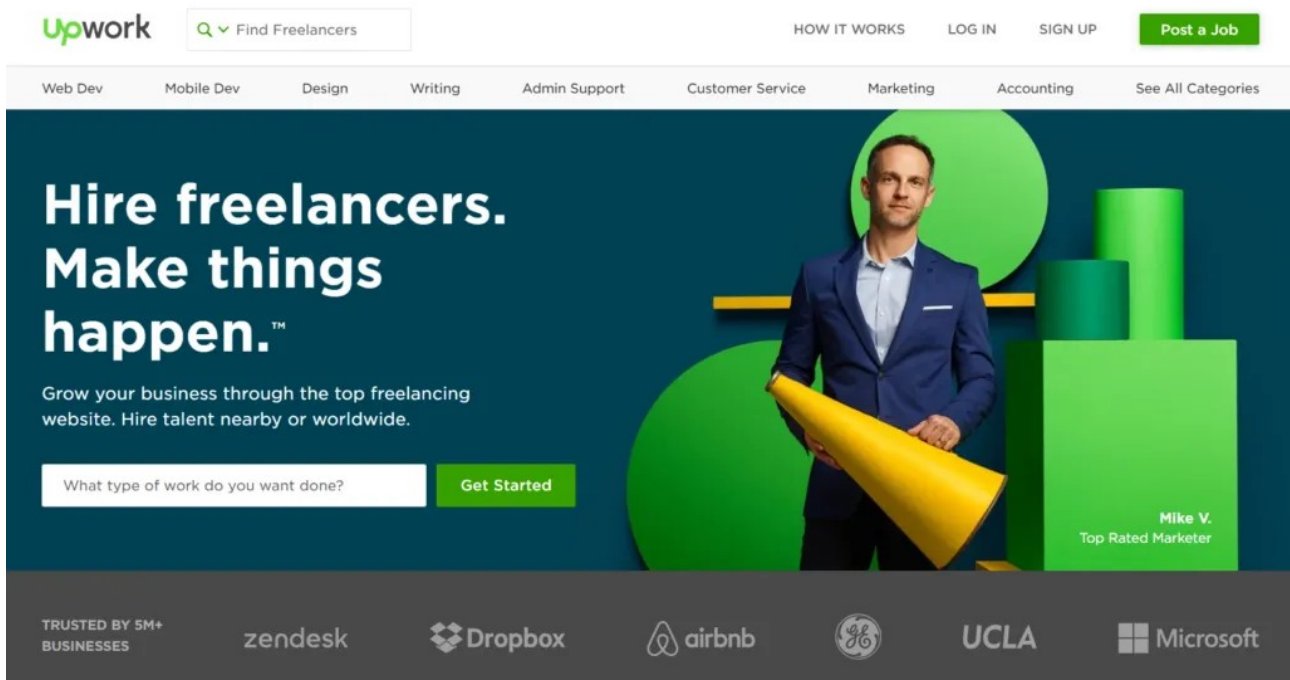
## **3. Gaji Awal Yang Sudah Tinggi**

Ini yang menarik, gaji awal seorang Fullstack Developer biasanya sudah tinggi. Bisa kisaran 6 – 7 juta untuk di Bandung. Dengan gaji awal yang sudah tinggi, maka ketika nanti Anda pindah kantor sudah dipastikan bisa dapat gaji diatasnya.

## **4. Lebih Mudah Mencari Freelance**

Karena Anda bisa membuat sebuah aplikasi sampai jadi, bahkan sampai bisa muncul di Play Store, sudah tentu kesempatan freelance terbuka lebar. Banyak sekali perorangan atau bahkan perusahaan yang minta dibuatkan aplikasi. Saran saya di awal-awal karir Anda perbanyak mengambil proyek freelance seperti ini untuk memperkaya portfolio.

Upwork adalah salah satu situs freelancer bergaji dollar yang bisa Anda coba.



*Fullstack Developer bisa menjadi freelancer di Upwork*

## 5. Jenjang Karir Lebih Banyak Pilihan

Jika Anda sudah cukup kenyang menjadi seorang Fullstack Developer ( biasanya dalam 1 tahun ), maka Anda akan sangat mudah untuk bisa memfokuskan diri ke jalur karir berikutnya. Bisa naik jadi Technical Lead, atau ternyata Anda lebih suka dengan ngoding visual, maka bisa fokuskan ke Frontend Developer. Bisa juga buka jasa Software House sendiri, bisa juga menjadi CTO di startup teman Anda.

## 6. Bisa Memulai Karir di Daerah

Ini khusus untuk Anda yang mungkin saat ini tinggal di luar Jawa, kesempatan untuk berkarir di daerah pun cukup tinggi. Karena tidak sedikit pemerintahan, bank-bank daerah, atau bisnis yang berkembang di daerah yang mulai ingin membuat aplikasi. Pasti mereka membutuhkan Developer.



## 1.6. Why JavaScript?

Sudah sejauh ini Anda mulai berminat menjadi seorang Fullstack Developer. Lalu bagaimana agar bisa menjadi Fullstack Developer yang dicari-cari perusahaan ? Jawabannya satu, yaitu kuasai bahasa pemrograman dan framework yang saat ini dan ke depannya masih banyak dipakai di Industri.

Di Indonesia sendiri bahasa yang paling banyak digunakan adalah Javascript, dan Framework yang paling banyak digunakan adalah NodeJS + ReactJS.



Kenapa Javascript dan kenapa NodeJS + ReactJS ? Kita akan bahas dari 2 sisi.

### 1.6.1. Alasan Dari Sisi Bisnis

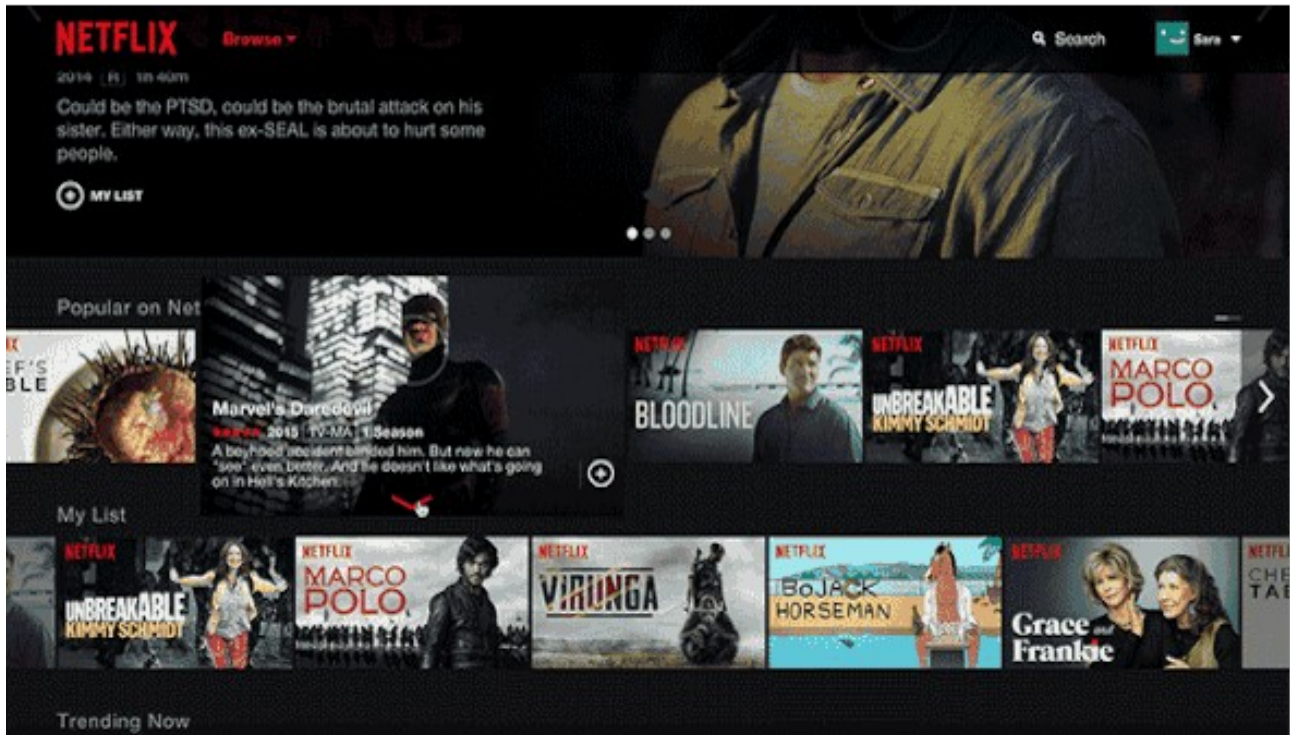
Jawabannya satu, karena dengan JavaScript menggunakan framework NodeJS + ReactJS, Fullstack Developer dapat membuat aplikasi yang real-time, cepat, ringan, handal dan tampilannya sangat modern.

Jika Anda perhatikan, semua aplikasi-aplikasi masa kini baik itu aplikasi di web maupun aplikasi Android pasti sangat ringan digunakan, sangat cepat, dan bisa melakukan berbagai interaksi secara realtime. Selain itu sangat handal walaupun di akses oleh banyak pengguna sekaligus.





Salah satu contoh sederhananya coba lihat aplikasi Netflix berikut ini yang sangat real-time dan canggih :



Coba bandingkan dengan aplikasi Internet Banking BCA berikut ini dimana ketika kita klik maka akan merefresh halaman :



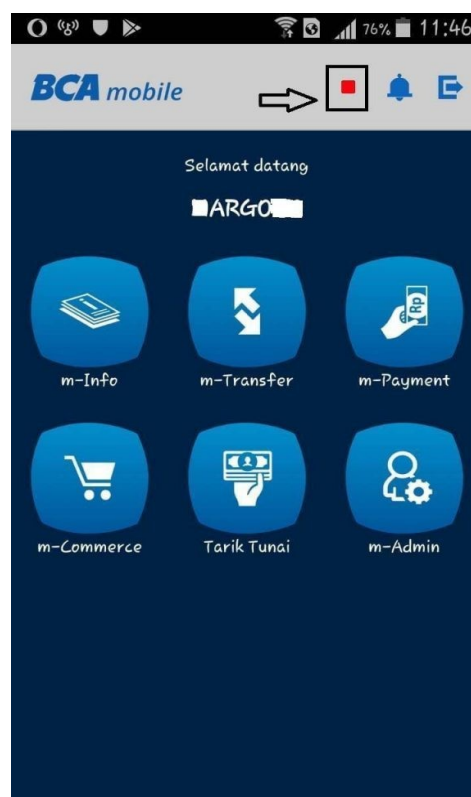

Aplikasi Netflix tersebut juga sangatlah handal & jarang down walaupun di akses jutaan penggunanya setiap hari di seluruh dunia. Bandingkan dengan aplikasi internet banking BCA yang kadang-kadang masih error dan down.

Dengan membuat aplikasi super keren, cepat, canggih, realtime, dan handal maka pengguna akan jauh lebih senang. Pengguna yang senang tentu akan berimbas pada meningkatnya kesuksesan perusahaan.

Apakah benar efeknya bisa sebesar itu ? Hanya karena aplikasi modern dan bagus bisa membawa kesuksesan pada perusahaan ? Ya. Memang sebesar itu.

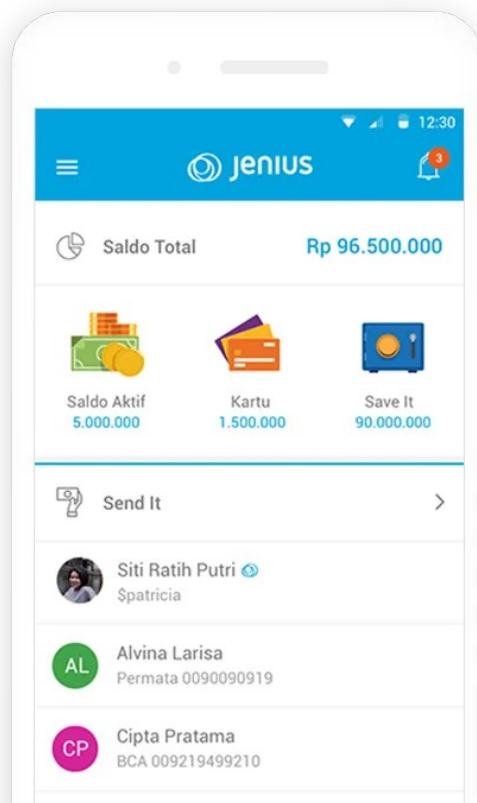
Salah satu contoh paling terkenal adalah ketika BTPN ( Bank yang saya pun hampir tidak tahu ) tiba-tiba meledak di kalangan milenial karena merilis Jenius, dimana tampilan aplikasinya sangat modern dan bagus. Jauh mengungguli aplikasi internet banking dan mobile banking dari semua bank besar di Indonesia.

Coba saja bandingkan tampilan aplikasi berikut :



Dengan tampilan aplikasi Jenius Mobile berikut :





Kira-kira mana yang menurut Anda yang lebih akan disukai pengguna ?

**NB : seluruh analogi dan contoh diatas sengaja dibuat oversimplified hanya untuk kebutuhan edukasi saja. Tidak ada maksud tertentu, juga kenyataannya banyak faktor lain yang mempengaruhi.**

### 1.6.2. Alasan Dari Sisi Teknis

Faktor-faktor utamanya adalah sebagai berikut :

#### 1. Mudah Dipelajari

Struktur bahasa pemrograman JavaScript terkenal sangat mudah dipahami oleh pemula sekalipun. Sehingga ini menjadi nilai plus untuk Anda yang berasal dari background non IT bisa cepat mempelajarinya.



Jika Anda rajin ngulik terus, sangat mungkin bisa menguasainya hanya dalam hitungan 3-4 bulan.

## 2. Performa Aplikasi Lebih Baik

Beberapa kemampuan NodeJS seperti Asynchronous dan Event Driven non Blocking I/O membuat performanya lebih cepat dan ringan dibanding misalnya bahasa pemrograman PHP.

Contoh sederhananya misal jika di PHP klik suatu tombol butuh 2 detik agar bisa tampil, di NodeJS mungkin bisa hanya 0.5 detik atau 1 detik saja. Perbedaan kecil ini akan sangat signifikan jika kita berbicara performa aplikasi yang di akses oleh jutaan orang per detiknya.

Inilah mengapa perusahaan-perusahaan sangat menyukai Javascript, khususnya NodeJS + ReactJS. Gambar diatas menunjukkan bahwa NodeJS jauh lebih stabil dibandingkan PHP yang makin lambat performanya jika jumlah request / akses semakin tinggi.

## 3. Dukungan Komunitas Tinggi

Javascript merupakan bahasa pemrograman nomor satu menurut data Github. Artinya begitu banyak developer di seluruh dunia yang saling berkontribusi. Sehingga jika Anda mengalami kesulitan dan butuh tempat bertanya, cukup mudah menemukannya.

Dengan berbagai manfaat diatas lah kenapa saat ini Fullstack Developer Javascript masih yang paling dicari-cari di seluruh dunia, khususnya Indonesia.



## 1.7. Skill yang Harus Dimiliki Seorang Fullstack Developer

### 1. Matangkan Attitude Anda

Attitude berada di peringkat 1. Karena jika bagian ini Anda tidak bisa penuhi, maka seluruh poin berikutnya tidak akan mungkin Anda kuasai. Anda harus punya kemauan keras untuk mau terus belajar, semangat, dan tidak malu untuk bertanya. Ini adalah kuncinya. **No Pain, No Gain.**

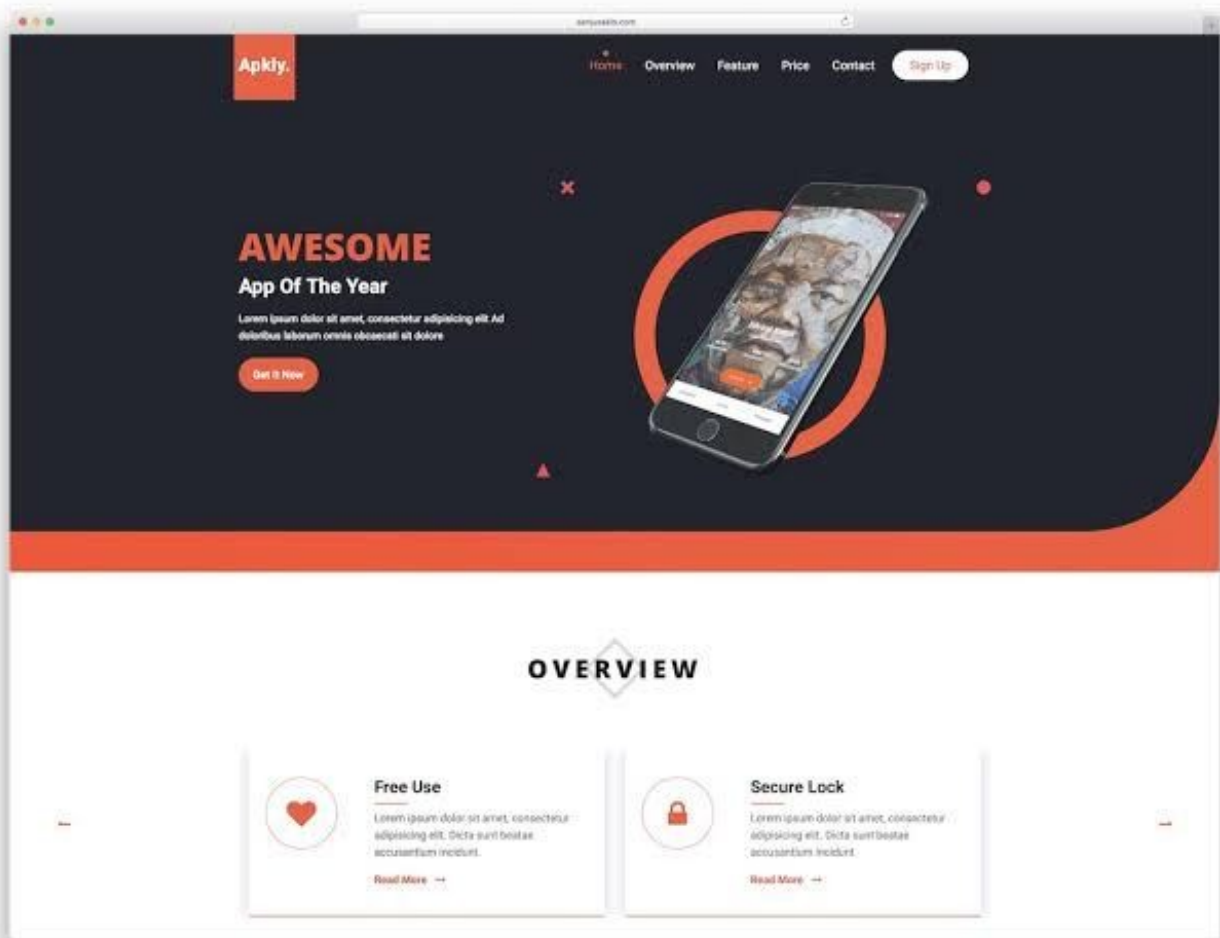
### 2. Dasar Programming dan Algoritma

Coba pahami dan banyak praktek terlebih dahulu berbagai konsep dasar pemrograman dan algoritma seperti If Else, Loop, While, Array. Ini melatih logika berpikir paling fundamental sebagai seorang Developer. Jika Anda sudah cukup paham konsep ini maka baru Anda bisa lanjut ke pematangan skill berikutnya.

### 3. Fundamental Frontend & Javascript ( HTML, CSS, ES6, JS )

Berikutnya Anda bisa mendalami bagaimana membangun tampilan-tampilan web. Membuat layout, membuat tabel, membuat tombol, menentukan warna, posisi, dll. Lalu Anda mulai bisa mempelajari dari sisi fundamental Javascript itu sendiri. Ini sangat penting, karena jika fundamental Javascript Anda kurang, maka akan kesulitan saat belajar NodeJS dan ReactJS nya nantinya.

Hasil akhirnya bisa Anda buat 1 buah halaman web static semacam ini :



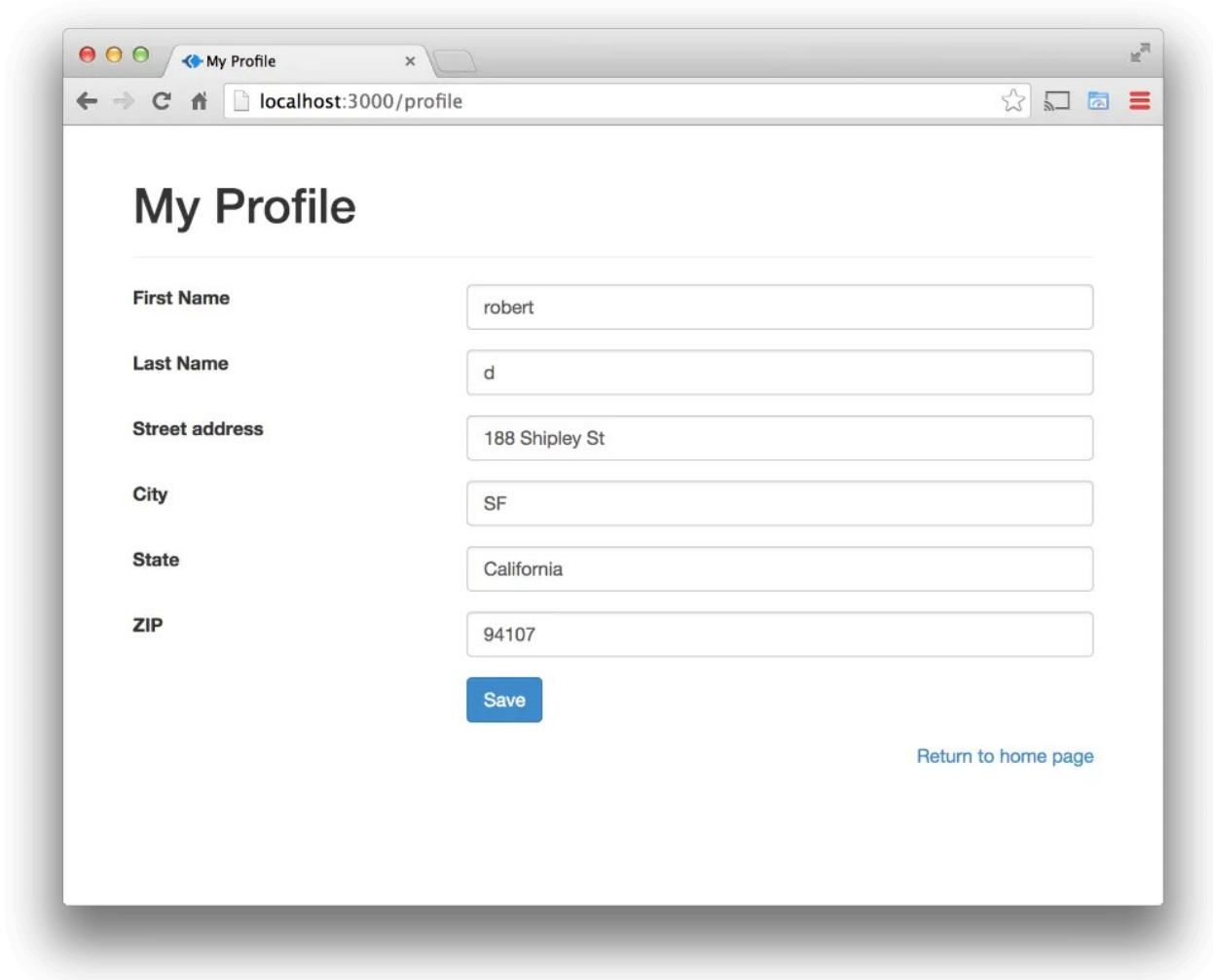
*Latihan awal Fullstack Developer membuat web sederhana*

#### 4. Fundamental Backend ( NodeJS, OOP & Database )

Berikutnya Anda bisa masuk untuk sisi Backendnya. Belajar bagaimana membuat sistem dari landing page yang sudah Anda buat sebelumnya. Disini Anda bisa mulai belajar menggunakan NodeJS dan Database seperti PostgreSQL.

Hasil akhirnya mungkin Anda bisa buat 1 buah aplikasi sederhana yang bisa menyimpan data seperti aplikasi absensi atau aplikasi kalkulator.





The screenshot shows a web browser window with the title 'My Profile' and the address bar displaying 'localhost:3000/profile'. The page content includes a heading 'My Profile' followed by a form with the following fields and values:

Field Label	Value
First Name	robert
Last Name	d
Street address	188 Shipley St
City	SF
State	California
ZIP	94107

Below the form is a blue 'Save' button. At the bottom right of the form area is a link that says 'Return to home page'.

## 5. Dalami Framework NodeJS & ReactJS

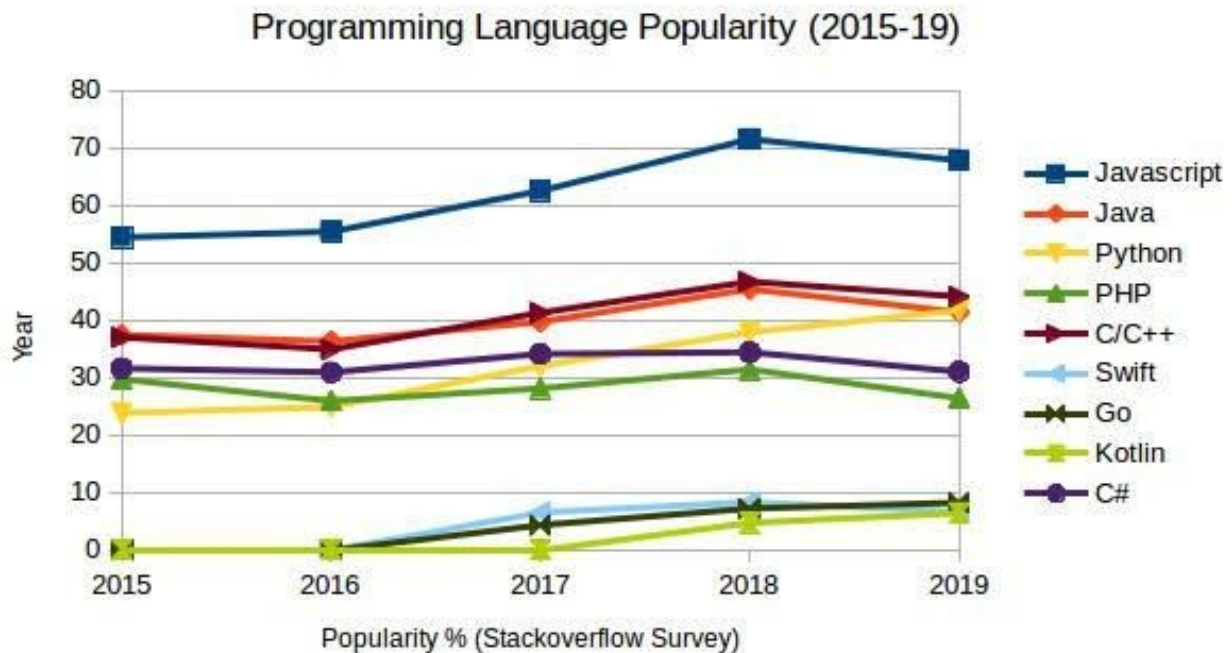
Terakhir baru Anda fokus membangun berbagai project real menggunakan Framework NodeJS & ReactJS. This is the fun part ! Semakin Anda sering praktek dan berhasil membuat macam-macam project, maka semakin jago Anda.

Kesimpulannya, jangan pernah kuatir walaupun Anda tidak punya background IT. Anda juga pasti bisa untuk menjadi seorang Fullstack Developer JS :)

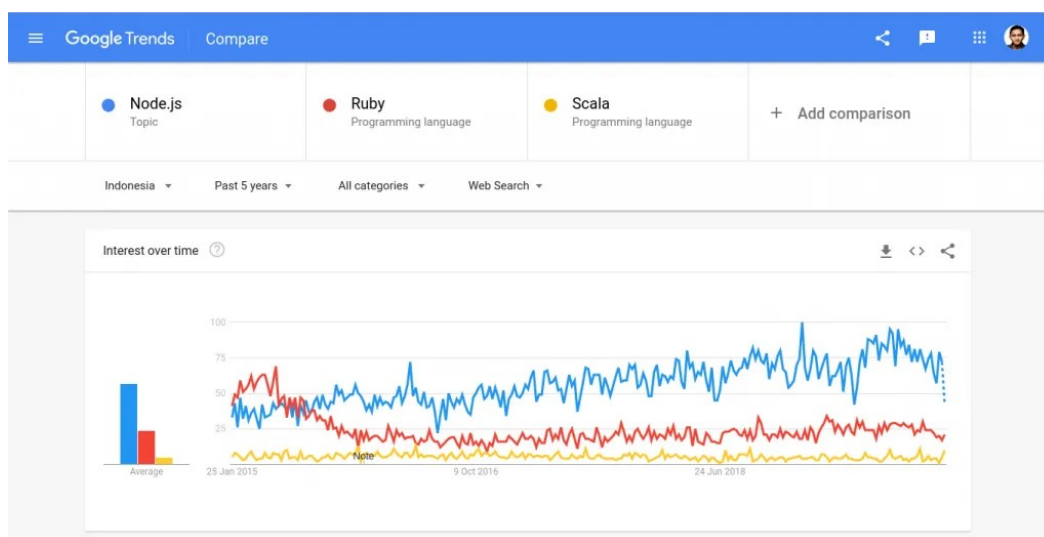


## 1.8. Peluang Karir Fullstack Developer JS

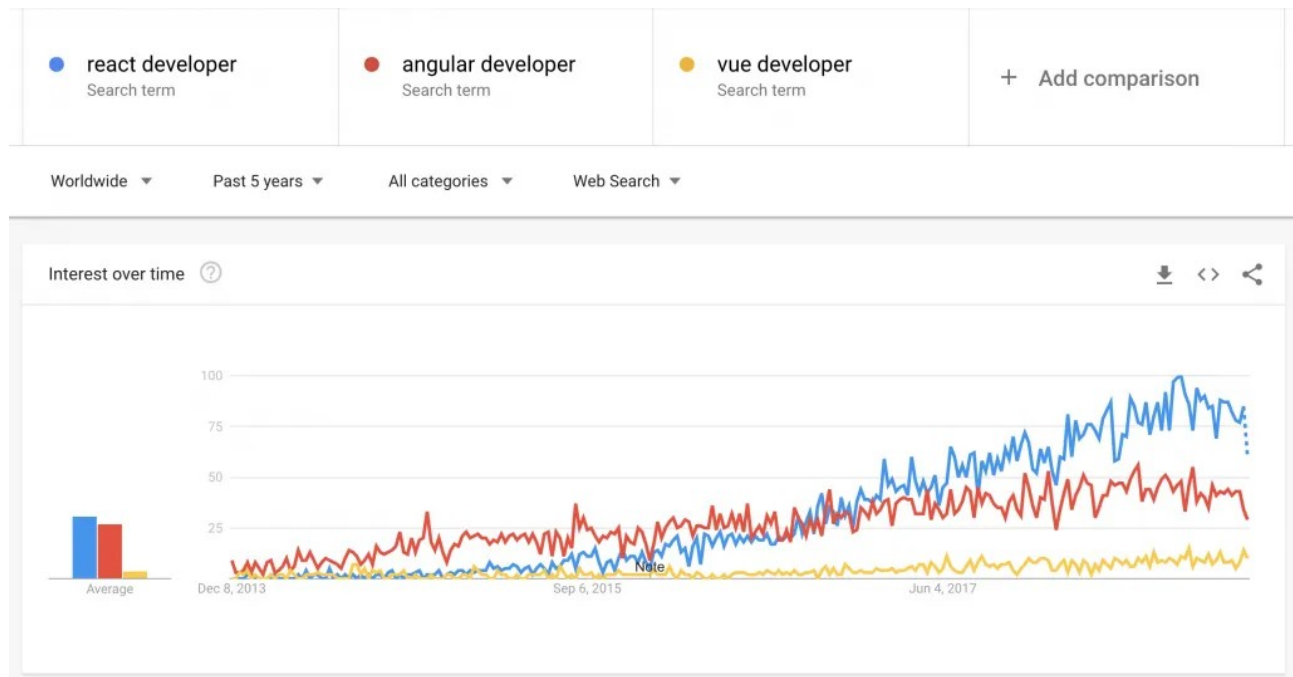
Peluang karir untuk menjadi seorang Fullstack Developer Javascript masih sangatlah besar. Salah satu survey yang diadakan oleh situs tanya jawab terbesar di dunia, Stackoverflow pun menunjukkan Javascript masih berada di peringkat 1 top skill Developer yang paling dicari :



Khusus untuk framework NodeJS juga masih menunjukkan tren Framework paling banyak digunakan dibanding Ruby dan Scala misalnya :



Bagaimana dengan React ? Saingan terberatnya adalah Angular dan Vue. Ternyata menurut tren juga React Developer masih menunjukkan tren positif dan berada di posisi 1 dibanding Angular maupun Vue.



Sebagai validasi apakah data-data diatas valid, cobalah main ke situs lowongan kerja seperti jobstreet.com atau indeed.com. Lowongan Fullstack Developer pasti dipenuhi requirement harus menguasai NodeJS dan ReactJS.

Lalu karir kerja apa saja sih yang bisa Anda tekuni sebagai Fullstack Developer Javascript ? Berikut diantaranya :

### 1. Menjadi Fullstack Developer

Ini sudah pasti. Anda bisa menjadi seorang Fullstack Developer baik di corporate, di startup, maupun di Software House jasa pembuatan aplikasi.

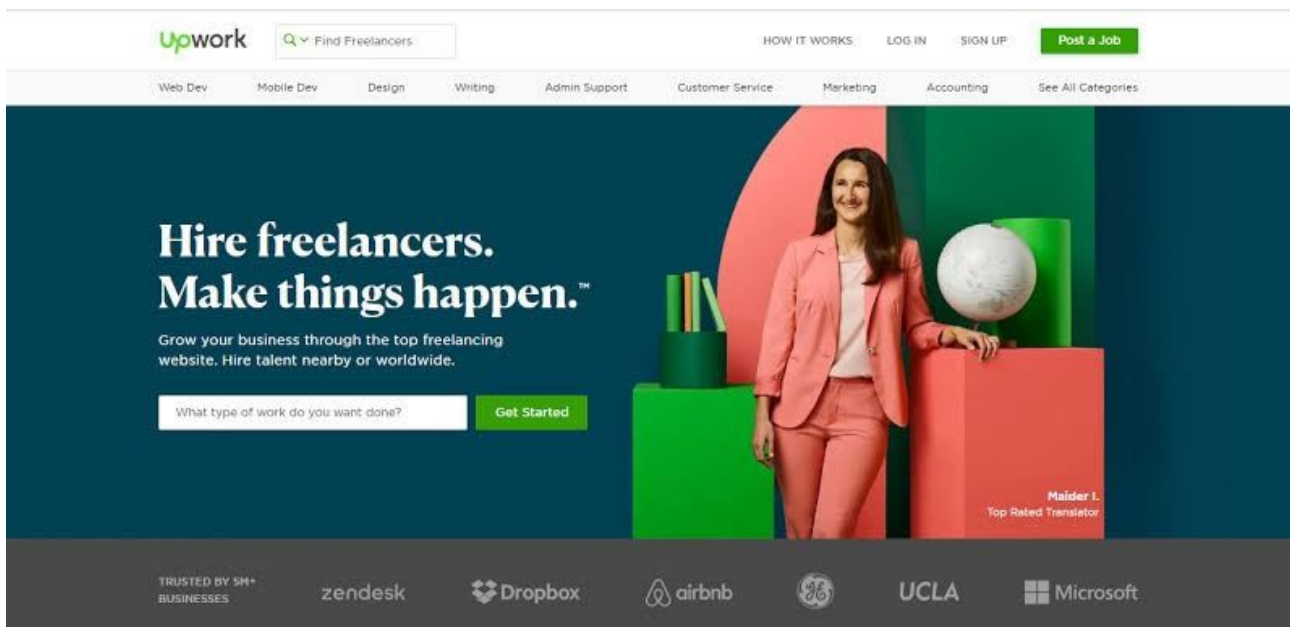
### 2. Menjadi Frontend / Backend



Anda pun bisa coba memilih lowongan karir yang lebih spesifik ke Frontend atau Backend Developer. Tentunya Anda juga perlu lebih mendalami ke salah satu frameworknya, apakah Anda lebih mendalami ke ReactJS ( dari sisi Frontend ) atau ke NodeJS ( dari sisi Backend ).

### 3. Menjadi Freelancer Developer

Sangat banyak kesempatan untuk membantu membuatkan aplikasi untuk klien-klien perusahaan / perorangan baik itu di Indonesia maupun di luar negeri. Cobalah untuk main-main ke situs seperti Upwork untuk mencoba mencari kesempatan menjadi Freelancer bergaji Dollar.



### 4. Menjadi CTO di Startup

Anda punya ide mau bikin aplikasi sendiri yang menjadi the next Facebook atau the next Gojek ? Maka Anda sangat bisa untuk membuatnya sendiri. Atau jika Anda kesulitan bagaimana menjalankan Startup dari sisi bisnisnya, cobalah cari partner. Jika Anda bisa coding sebagai seorang Fullstack, kemampuan Anda sangat dicari-cari. Biasanya akan banyak teman yang mencoba menawarkan Anda untuk membangun sebuah startup.



Pilihan karir manapun yang Anda pilih, tentunya sama baiknya. Peluangnya sama besarnya. Tinggal Anda sesuaikan dengan visi hidup Anda masing-masing.

## 1.9. How to be a Good Developer

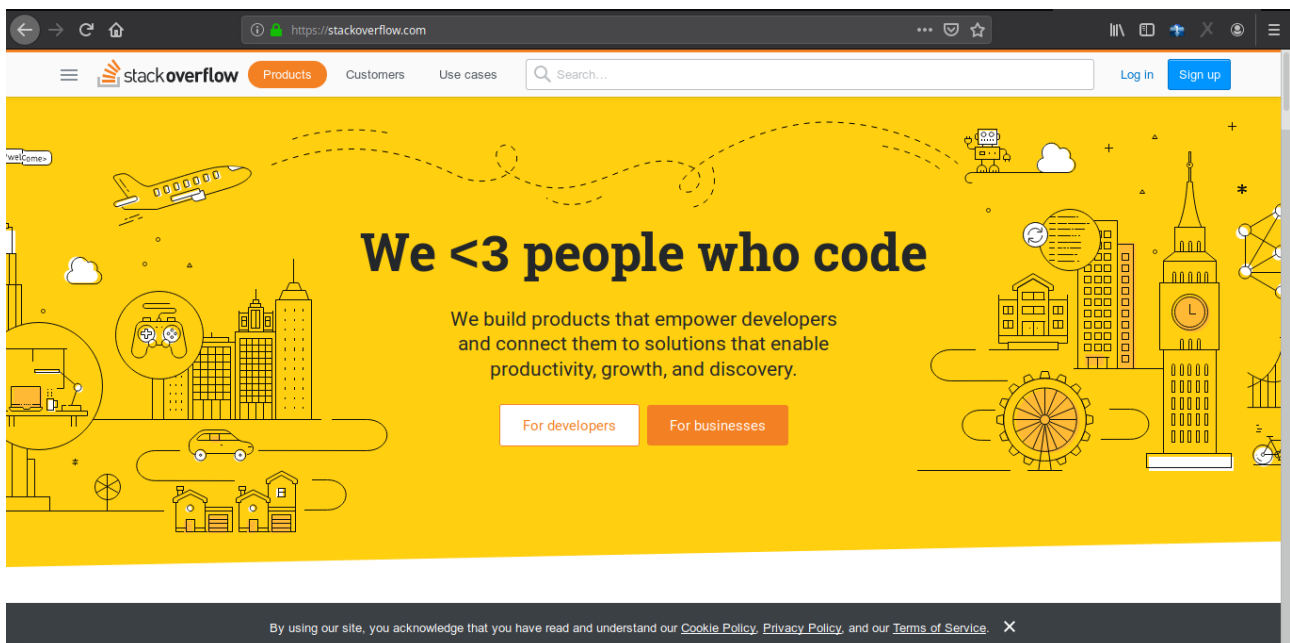
Berikut yang perlu kamu asah ketika ingin menjadi Developer yang handal:

1. Sabar, kesabaran dalam membangun dan melakukan troubleshooting sangat dibutuhkan oleh engineer. Pengambilan keputusan sangatlah penting saat terjadi sebuah masalah dan sabar adalah kunci agar kita dapat berfikir jernih untuk mengambil keputusan.
2. Visionary, dapat melihat apa yang akan terjadi selanjutnya. Bukan seperti cennyang, namun visionari disini adalah rancangan bagaimana pekerjaan kita selanjutnya dan apa dampaknya.
3. Detail, jangan menyepelekan hal-hal kecil. Perhatikan detail dari setiap pekerjaan, pastikan semua sudah sesuai perancangan.
4. Berfikir jangka panjang, fikirkan sistem berjalan di kemudian hari. Apakah sistem yang kita bangun masih sanggup melayani pengguna? Jika tidak bagaimana solusinya.
5. Haus akan pengetahuan, terus belajar dan mencari masalah.
6. Jadilah pemalas!, pemalasan bisa saja menyelesaikan hal rumit dengan cara yang lebih efisien dan cepat. Mengapa? Karena mereka malas mengerjakannya!
7. Dapat berkomunikasi dengan baik, sampaikan apa yang ada difikiran kamu, terima apa pendapat orang lain, berdiskusi bila ada yang tidak sesuai.
8. Dapat bekerja tim! Jangan bekerja sendiri, banyak kepala lebih baik dari pada satu, namun ingat! Jangan egois.
9. Time management, pastikan pekerjaan beres sebelum deadline. Atur waktu mu agar semua pekerjaan dapat terselesaikan.
10. Good Programming Skill, perhatikan apa saja yang perlu kalian kuasai.

11. Mengetahui Kapan Harus Optimalisasi, berikan alasan ketika melakukan optimalisasi agar tidak membuang waktu untuk optimalisasi hal yang tidak perlu di optimalkan.
12. Good Testing Skill, sebelum di luncurkan pastikan website tidak memiliki bug. Skill testing diperlukan untuk memastikan hal website bug free!

### 1.9.1. Cara Bertanya yang Baik

Di sebuah literatur mengatakan bahwa **keberhasilan, kepintaran, dan kesuksesan kita itu bukan ditentukan dari cara kita mencari informasi, melainkan dari cara kita mengajukan pertanyaan**. Ada benarnya, karena permasalahan-permasalahan dalam hidup seringkali tidak ada jawabannya di buku atau internet. Misalnya kita menghadapi error yang sudah kita cari di internet tidak ketemu jawaban yang memuaskan dan tidak bisa menyelesaikan masalah kita, mau tidak mau kita mesti tanya ke orang lain yang kita anggap pakar di bidangnya. Maka dari itu, kita harus selalu berpikir panjang bagaimana caranya mengajukan pertanyaan. Terutama kepada orang yang sibuk, orang yang tidak punya banyak waktu untuk melayani orang lain supaya mereka mau menjawab pertanyaan kita.



*Halaman awal situs Stackoverflow, tempat para engineer saling tanya jawab*

Jadi, bagaimana cara bertanya yang benar?





Sebelumnya, kita harus tahu dulu contoh-contoh pertanyaan yang salah dan membuat orang malas untuk membalas/menjawab pertanyaan Anda.

- gan, error pas \_\_\_\_\_. itu kenapa ya?
- ada yang pernah nyoba \_\_\_\_\_?
- ada yang paham \_\_\_\_\_, ada yang mau saya tanyakan.
- ada yang tau ini kenapa? [screenshot]
- ada yang bisa bantu ane buat \_\_\_\_\_

Dan jika Anda bertanya seperti ini di Stackoverflow, barangkali tidak akan ada yang mau jawab, bahkan bisa di-downvote dan ujung-ujungnya bisa di bully.

Kalau kita lihat dari pertanyaan-pertanyaan itu dapat kita ambil poin-poin karakter pertanyaan yang kemungkinan dibalasnya kecil, yaitu:

1. Pertanyaan yang tidak berbobot.
2. Pertanyaan yang kekurangan informasi.
3. Pertanyaan yang retoris (tidak memerlukan jawaban), alias “udah tau nanya”.
4. Meminta bantuan yang menguras waktu.
5. Pertanyaan yang tidak jelas apa maksud pertanyaannya.

Jadi, supaya pertanyaan kita dibalas dengan jawaban yang kita butuhkan, maka kita perlu mencegah bertanya dengan karakter pertanyaan seperti diatas, atau dengan solusi-solusi berikut:

1. **Hindari bertanya kalau anda belum melakukan apapun.** Tujuan bertanya adalah untuk mendapatkan jawaban atas kesulitan yang kita hadapi, bukan untuk membuat orang lain berpikir keras menggantikan anda. Anda justru akan dicap malas.
2. **Berikan informasi yang spesifik.** Orang yang anda tanyakan harus tahu persis kondisi anda, terutama apa saja yang sudah anda lakukan dan apa yang menyebabkan permasalahan tersebut terjadi. Jangan sampai ada celah informasi yang menyebabkan orang yang ditanyakan berasumsi tentang kondisi anda.



3. **Jangan meminta bantuan yang menyita waktu.** Semua orang punya kesibukannya masing-masing. Ketika bertanya, asumsikan orang yang anda tanyai tidak punya waktu sama sekali. Pikirkan bagaimana caranya supaya orang yang tidak punya waktu ini jadi bersedia menjawab pertanyaan anda.
4. **Cari dulu jawabannya sendiri.** Jangan tanyakan pertanyaan yang sudah ada banyak jawabannya di internet, apalagi di depan mata. Cari dulu, gunakan Google. Kalau memang pertanyaan anda sama sekali tidak ada jawabannya di mana pun, baru ditanyakan.
5. **Singkat, padat, jelas.** Hilangkan semua informasi yang tidak perlu disertakan dalam pertanyaan (termasuk formalitas), buat supaya pertanyaan anda jadi sesingkat mungkin. Pastikan juga pertanyaannya bisa jelas dipahami.
6. Kalau anda butuh jawaban dari pertanyaan **yang lebih rumit**, atau kalau anda butuh bantuan mereka, anda harus **jalin hubungan positif dengan mereka**. Apalagi kalau orangnya super sibuk.

Nah, dari keenam poin diatas, berikut adalah contoh format pertanyaan yang baik dan benar.

Selamat pagi/siang/malam teman-teman.

saya sudah mengikuti tutorial ini, namum mendapat kendala di \_\_\_\_\_,  
dengan pesan error \_\_\_\_\_.

Berikut ini konfigurasi yang saya tulis: \_\_\_\_\_ (link gist.github.com).

dan beberapa log: \_\_\_\_\_ (link gist.github.com).

Screenshot tampilannya atau errornya seperti ini:

\_\_\_\_\_ (link image/bisa juga diupload)

Tindakan yang sudah saya lakukan:



1. Saya sudah mengubah ini \_\_\_\_ menjadi \_\_\_\_  
2. Saya baru update \_\_\_\_ ke versi terbaru  
  
Versi \_\_\_\_ yang saya gunakan saat ini adalah \_\_\_\_.  
Sekian pertanyaan saya, terima kasih

### 1.9.2. Asking and Googling

Jika pada bagian sebelumnya kita telah mempelajari bagaimana cara bertanya yang baik (khususnya di forum tanya jawab), kali ini kita akan belajar cara menggunakan situs mesin pencari (search engine) Google yang baik dan benar.



*Mesin Pencari Google*

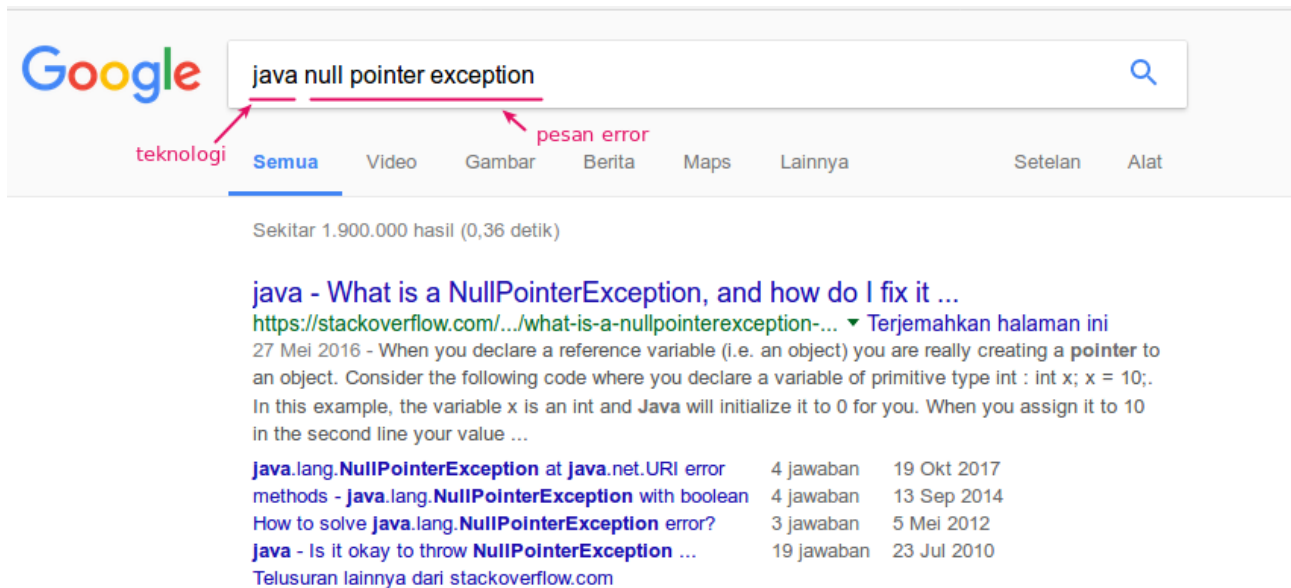
Lho, mau pakai Google aja kok pake belajar lagi? Nah ini yang membedakan mental kita sebagai engineer dengan orang awam. Seorang engineer yang professional pun terkadang masih perlu menggunakan Google untuk mencari jalan keluar dari permasalahan yang dihadapinya saat melakukan pekerjaannya. Sebagai seorang engineer, berikut adalah tips agar apa yang sedang Anda cari dapat ditemukan dengan mudah.

1. Cari **menggunakan bahasa Inggris**.
2. Cari dengan **kata kunci yang detail, lengkap, dan jelas**. Contohnya teknologi yang digunakan, pesan error yang ditemui, versi yang digunakan, dll.
3. Cari dan **baca manual guide** dari aplikasi yang Anda gunakan terlebih dahulu.



4. Ambil referensi dari situs-situs yang cukup terpercaya, seperti forum Reddit, Stackoverflow, Tecmint, DigitalOcean, dll. Apabila tidak menemukan, baru kita bisa coba cari dari blog-blog yang beredar.

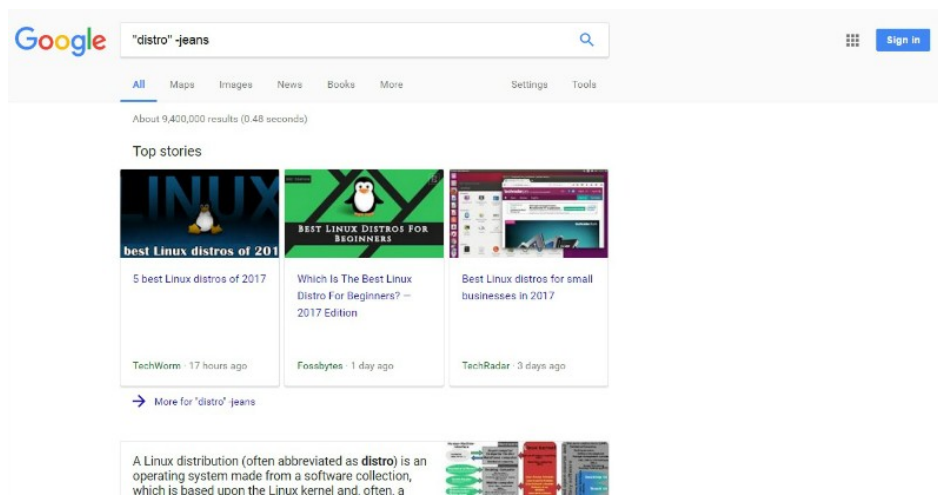
Contoh cara Googling yang benar:



*Melakukan pencarian di mesin pencari Google*

Selain itu, Anda juga bisa menggunakan karakter-karakter untuk memudahkan pencarian Anda, diantaranya:

1. Menggunakan tanda minus (-)



*Penggunaan Tanda Minus*



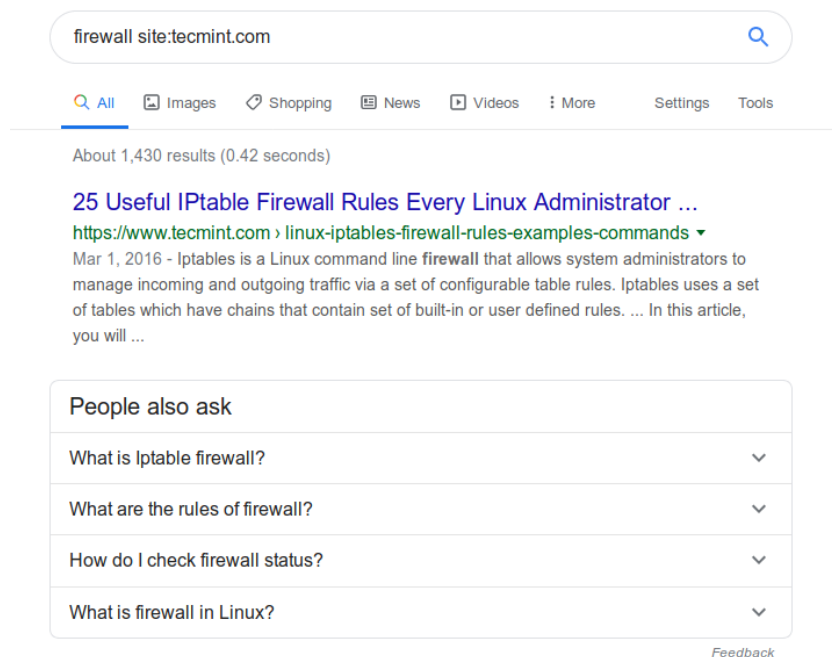
Banyak nama yang sama untuk berbagai informasi berbeda. Agar kita mendapatkan informasi sesuai dengan bidang yang kita cari, kita bisa menyisipkan tanda minus (-) di belakang frase atau kata yang dicari. Tanda minus itu diikuti dengan bidang yang kita cari. Semisal fashion, sport, dan sebagainya.

2. Gunakan tanda petik ("...") untuk mencari kata yang benar-benar diinginkan.

Pemakaian tanda kutip berguna untuk mencari kalimat dengan kata-kata dan urutan yang sama sesuai dengan yang kita ketikkan. Contohnya saja kita mencari kalimat : "Ilmu komunikasi menurut Dedy Mulyana", maka hasil pencarian yang akan keluar hanyalah yang berisi kalimat dengan urutan yang sama seperti yang kita ketikkan.

3. Cari menggunakan nama website

Di mesin pencarian Google, kita dapat melakukan pencarian secara spesifik dengan menambahkan atribut "site:<alamat website>"

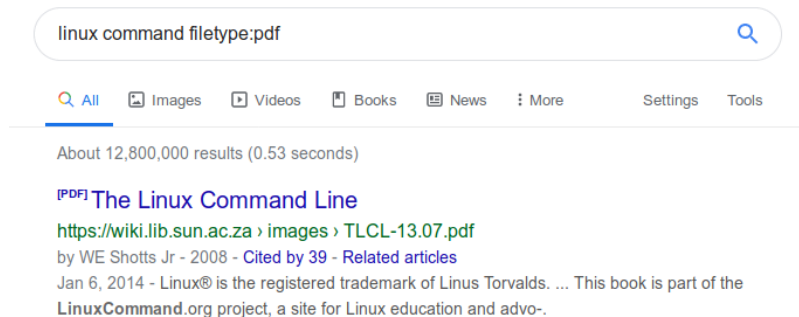


*Mencari suatu topik pada alamat website tertentu*

4. Mencari menggunakan jenis ekstensi file



Anda juga bisa mencari literatur dengan tipe ekstensi file tertentu, yaitu dengan cara menambahkan atribut “filetype:<jenis ekstensi file>”.



*Mencari dengan jenis ekstensi file*

### 1.9.3. Cara Belajar yang Baik

Selain cara bertanya dan cara melakukan pencarian di mesin pencari Google, hal ini yang paling perlu kita bahas; **CARA BELAJAR**. Kalau berbiscara tentang cara belajar, tentunya setiap orang pasti memiliki cara dan metode belajar yang berbeda, ada yang lebih senang dengan melihat (visual), mendengar (auditoris), kinestetik, atau kombinasi beberapa panca indra. Pada bagian ini akan dibahas beberapa mindset yang perlu kita tanamkan dalam cara belajar kita, khususnya di bidang IT.

#### 1. Banyak Membaca

Harry S. Truman berkata, “**not all readers are leaders, but all leaders are readers.**” Sebuah ungkapan luar biasa yang selalu menjadi cambuk untuk kita untuk terus membaca. Untuk menjadi seseorang yang ingin secara professional berkarir di bidang IT (atau bidang apapun), tidak ada yang instant, semuanya perlu proses. Dan proses paling dasar untuk belajar sesuatu adalah dengan **membaca**. Di internet, sekarang sudah banyak bahan bacaan apabila ingin mendalami web development, seperti petanikode.com, dll.

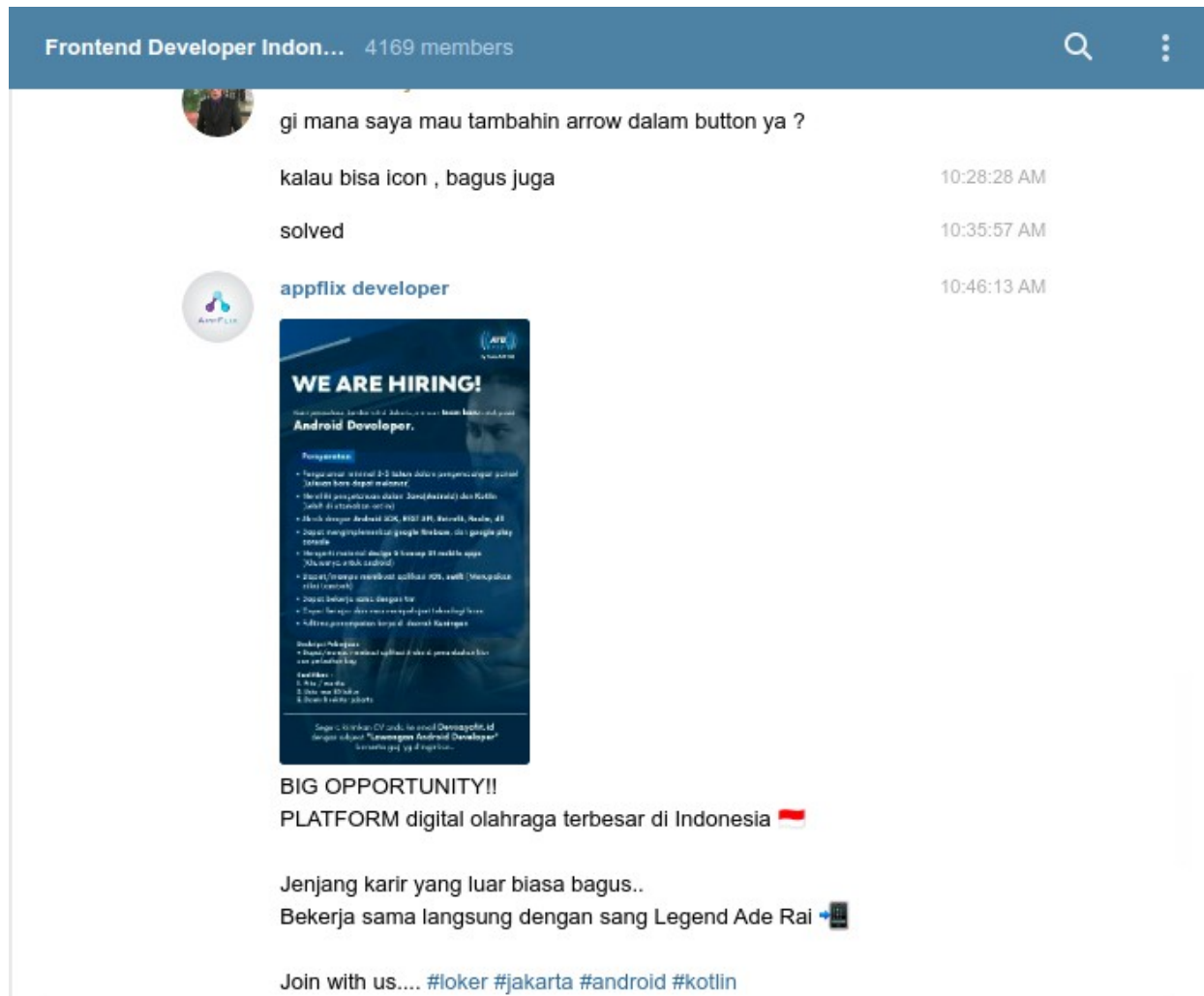




## 2. Aktif di Forum

Saat ini sudah banyak forum-forum yang menyediakan layanan tanya jawab IT dan perkembangan-perkembangannya. Sebut saja forum-forum di Telegram, Facebook, dll.





*Salah satu forum Front End Developer di Telegram*

### 3. Jangan Malas Ngulik!

Hal ini biasanya muncul ketika sudah jenuh dan sudah tersibukkan dengan suatu aktivitas lainnya. Mindset inilah yang harusnya kita hilangkan jauh-jauh apabila kita ingin professional di suatu bidang. Seperti yang kita tahu bahwa dunia developing sangat luas. Apalagi dengan adanya perkembangan teknologi, pastinya akan membuat lebih banyak threat yang mungkin perlu kita pelajari dan “ulik” lagi.





## 1.10. Membangun Mindset Seorang Programmer

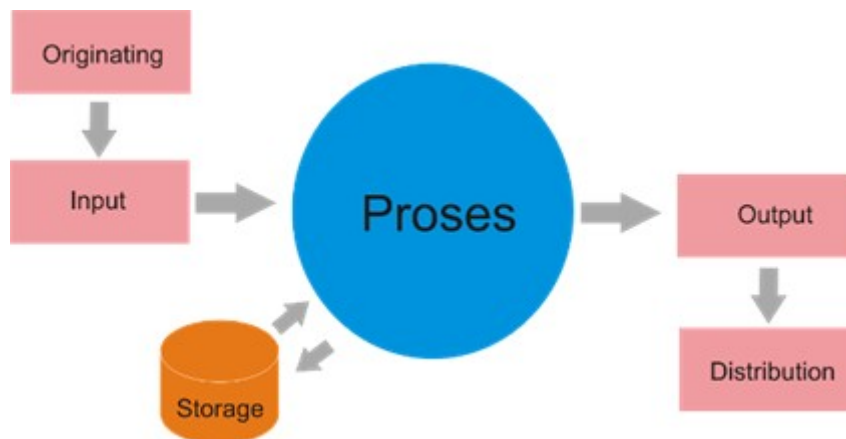
Programming adalah sebuah ilmu yang mempelajari cara membuat sebuah program melalui instruksi-instruksi yang terstruktur dan sistematis (Algoritma). Sebagian orang mungkin banyak yang tidak suka dengan programming karena ilmu ini terkesan sulit untuk dipelajari. Padahal pada kenyataannya tidak seperti itu.

Seperti yang sudah disebutkan diatas, bahwa pola pikir seorang programmer akan cukup berbeda dengan orang awam. Bedanya, dalam proses membuat code, seorang programmer harus memperhatikan langkah-langkah detail dan logika yang akan dibangun. Berikut akan dijelaskan mengenai mindset algoritma yang harus setiap programmer kuasai.

Pada dasarnya konsep dasar pemrograman komputer terdiri dari **Input, proses, dan output**. Namun kini konsep tersebut dikembangkan lagi menjadi **Originating > Input > Proses > Output > Distribution**.

- a. Originating merupakan pengumpulan data yang biasanya berupa pencatatan data sebelum proses input.
- b. Input merupakan proses memasukan data ke dalam komputer menggunakan perangkat input (mouse, keyboard atau lainnya).
- c. Setelah data di inputkan maka akan diproses menggunakan perangkat processing yang biasanya terdiri dari menghitung, membandingkan, mengurutkan, mengelompokkan, dan mencari perangkat penyimpanan (storage).
- d. Data yang sudah diproses akan ditampilkan berupa informasi melalui perangkat output (speaker, monitor, atau lainnya).
- e. Sedangkan distribution adalah proses menyebarkan informasi kepada pihak-pihak tertentu.
- f. Ada satu lagi komponen penting yaitu Storage. Storage adalah tahapan yang merekam hasil pengolahan data. Dan nantinya digunakan untuk proses input selanjutnya.





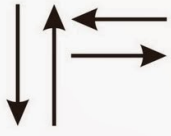






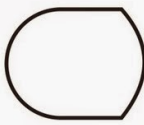
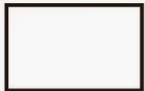


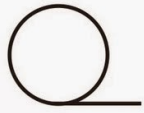
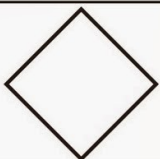


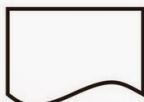
*Diagram Proses Pemrograman Komputer*

## 1.11. Pengenalan Flowchart dan Algoritma Pemrograman

**FlowChart** adalah adalah suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program.

Dalam perancangan flowchart sebenarnya tidak ada rumus atau patokan yang bersifat mutlak (pasti). Hal ini didasari oleh flowchart (bagan alir) adalah sebuah gambaran dari hasil pemikiran dalam menganalisa suatu permasalahan dalam komputer. Karena setiap analisa akan menghasilkan hasil yang bervariasi antara satu dan lainnya. Kendati begitu secara garis besar setiap perancangan flowchart selalu terdiri dari tiga bagian, yaitu input, proses dan output. Seperti konsep pemrograman yang sudah kita pelajari sebelumnya.

Simbol-simbol yang biasa digunakan untuk membuat flowchart :

	<b>Flow Direction symbol</b> Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga connecting line.		<b>Simbol Manual Input</b> Simbol untuk pemasukan data secara manual on-line keyboard
	<b>Terminator Symbol</b> Yaitu simbol untuk permulaan (start) atau akhir (stop) dari suatu kegiatan		<b>Simbol Preparation</b> Simbol untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam storage.
	<b>Connector Symbol</b> Yaitu simbol untuk keluar - masuk atau penyambungan proses dalam lembar / halaman yang sama.		<b>Simbol Predefine Proses</b> Simbol untuk pelaksanaan suatu bagian (sub-program)/prosedure
	<b>Connector Symbol</b> Yaitu simbol untuk keluar - masuk atau penyambungan proses pada lembar / halaman yang berbeda.		<b>Simbol Display</b> Simbol yang menyatakan peralatan output yang digunakan yaitu layar, plotter, printer dan sebagainya.
	<b>Processing Symbol</b> Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer		<b>Simbol disk and On-line Storage</b> Simbol yang menyatakan input yang berasal dari disk atau disimpan ke disk.
	<b>Simbol Manual Operation</b> Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh computer		<b>Simbol magnetik tape Unit</b> Simbol yang menyatakan input berasal dari pita magnetik atau output disimpan ke pita magnetik.
	<b>Simbol Decision</b> Simbol pemilihan proses berdasarkan kondisi yang ada.		<b>Simbol Punch Card</b> Simbol yang menyatakan bahwa input berasal dari kartu atau output ditulis ke kartu
	<b>Simbol Input-Output</b> Simbol yang menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya		<b>Simbol Dokumen</b> Simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas.

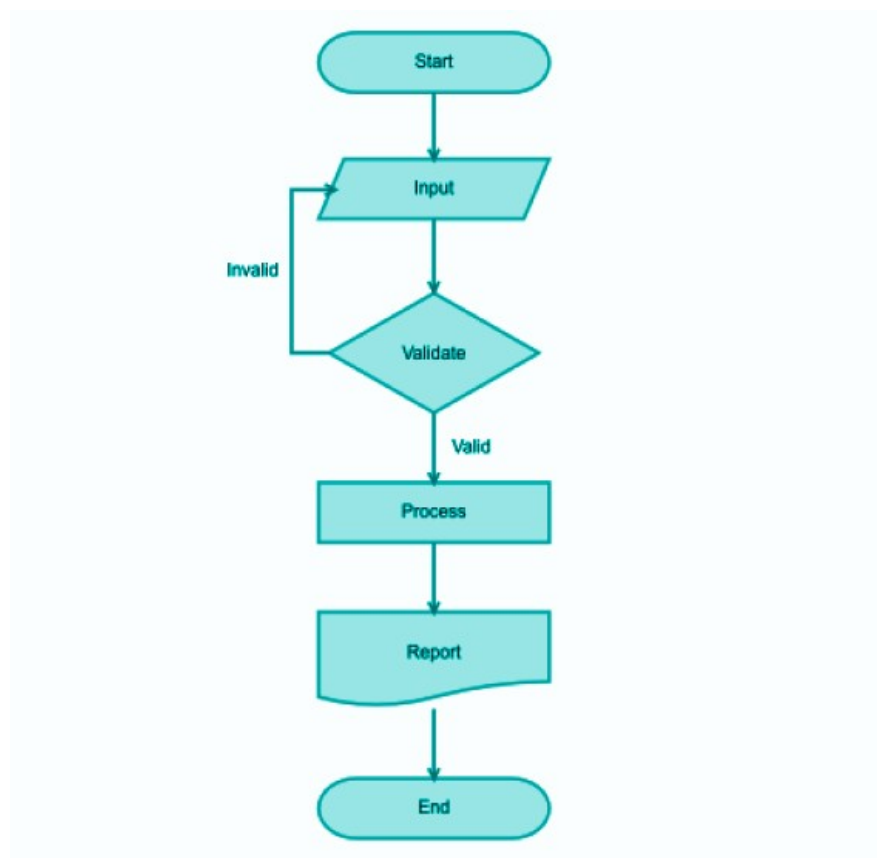
### 1.11.1. Pedoman-Pedoman Dalam Membuat Flowchart

Jika seorang analis dan programmer akan membuat flowchart, ada beberapa petunjuk yang harus diperhatikan, seperti :

1. Flowchart digambarkan dari halaman **atas ke bawah** dan dari **kiri ke kanan**.



2. Aktivitas yang digambarkan harus didefinisikan secara hati-hati dan definisi ini **harus dapat dimengerti** oleh pembacanya.
3. Kapan aktivitas dimulai dan berakhir harus ditentukan secara **jelas**.
4. Setiap langkah dari aktivitas harus diuraikan dengan **menggunakan deskripsi kata kerja**, misalnya “Melakukan Duplikasi Data”.
5. Setiap langkah dari aktivitas harus berada pada **urutan yang benar**.
6. Lingkup dan range dari aktifitas yang sedang digambarkan harus ditelusuri dengan hati-hati. Percabangan-percabangan yang memotong aktivitas yang sedang digambarkan tidak perlu digambarkan pada flowchart yang sama. Simbol konektor harus digunakan dan percabangannya diletakan pada halaman yang terpisah atau hilangkan seluruhnya bila percabangannya tidak berkaitan dengan sistem.
7. Gunakan **simbol-simbol flowchart yang standar**.



*Contoh penerapan dari flowchart*

Note : penerapan flowchart tidak harus sama karna tergantung kebutuhan dari setiap logika pemrograman.

### 1.11.2. Exercise

Buatlah 2 buah flowchart aplikasi sederhana! Anda dapat menggunakan aplikasi pembuat diagram seperti Microsoft Visio, Dia, atau platform online draw.io.

### 1.11.3. Mengenal Algoritma Pemrograman

**Algoritma Pemrograman** merupakan urutan dari langkah-langkah penyelesaian sebuah case atau kasus di tulis dengan bahasa pemrograman oleh developer agar proses dari sebuah program berjalan efektif.



Tujuan pembuatan suatu program tentunya adalah untuk membantu mengurai dan mengatasi suatu masalah. Masalah dapat di atasi tentunya dengan urutan langkah-langkah tertentu. Untuk masalah dengan instansiasi yang relatif kecil kita dapat menemukan solusinya dengan mudah dan cepat. Namun bila masalah yang relatif lebih besar tentunya akan sulit untuk menemukan solusiya dengan cepat tanpa bantuan suatu metode yang menguraikan solusi-solusi dari suatu masalah menjadi langkah-langkah tertentu. Langkah-langkah penyelesaian masalah tersebut yang kemudian disebut sebagai **algoritma**.

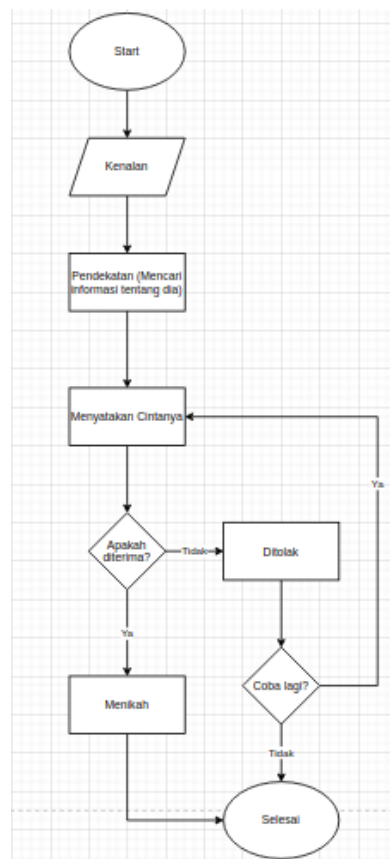
Algoritma dalam kehidupan sehari-hari sering kita temui tanpa sadar. Seperti makan, masak mie instan, dan yang lainnya. Berikut adalah beberapa contoh algoritma yang ada pada kehidupan sehari-hari kita.

#### a. Algoritma Makan dengan Flowchart





## b. Algoritma PDKT dengan Flowchart



Kenapa belajar Algoritma ?

- a. Menggunakan fungsi algoritma bisa digunakan untuk memecahkan program yang rumit.
- b. bisa menjadikan program yang besar menjadi program yang lebih sederhana
- c. memudahkan dalam pembuatan program
- d. meminimalisir penulisan program yang berulang
- e. Program menjadi lebih terstruktur dan rapi sehingga lebih mudah dipahami maupun dikembangkan
- f. Ketika terjadi kesalahan bisa dicari dengan mudah karena dengan fungsi algoritma bisa mendapatkan alur yang jelas

Manfaat belajar Algoritma ?

1. Memperkuat cara berfikir kita agar dapat menyelesaikan suatu masalah
2. Membantu otak agar berfikir jangka panjang
3. Memperkuat analisis ketika pembuatan program
4. Memperluas space berpikir
5. Memperkuat otak kita untuk berpikir panjang

**Pemrograman** itu sendiri adalah proses membuat suatu Program dengan Algoritma + Bahasa Pemrograman. Sebuah Software hanya bisa terwujud jika dibuat dengan Algoritma dan Bahasa Pemrograman.

Langkah-langkah yang dilakukan dalam pemrograman yang umum adalah :

- Definisikan Masalah
- Mengumpulkan kebutuhan (requirement)
- Buat Algoritma dan Struktur Cara Penyelesaian
- Menulis Program dengan bahasa pemrograman





- Testing dan Verifikasi Program
- Implementasi/Instalasi Program
- Dokumentasi Program
- Pemeliharaan Program

Kita bisa memilih Bahasa Pemrograman apa saja yang kita mau dan kuasai. Namun tentu tidak semua bahasa pemrograman cocok dengan permasalahan yang ada dan kita wajib mencari tahu manakah yang cocok dan sesuai dengan kebutuhan kita.

Contoh untuk membuat App Android kita bisa pilih bahasa pemrograman Java, untuk App iOS kita pakai Swift dan Objective-C dsb, untuk Website kita bisa pakai PHP /Javascript /Java/ Ruby dsb, membuat games bisa pakai C++ dsb.

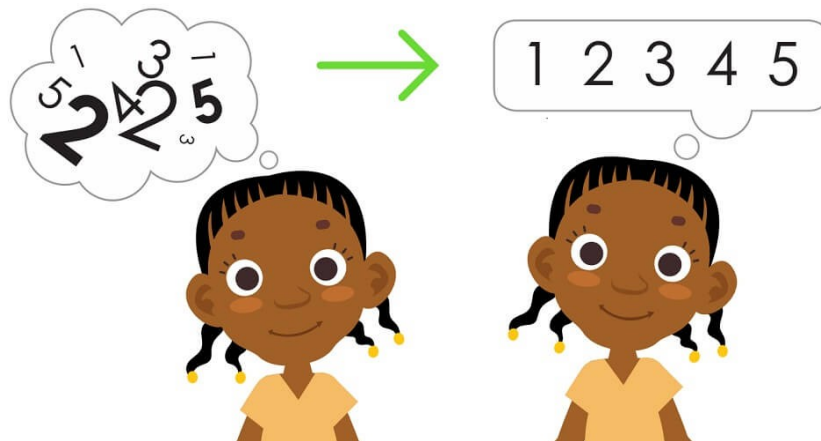
Dalam membangun aplikasi (Software Engineering), metoda yang umum digunakan adalah Waterfall, Agile, Scrum, Extreme Programming, Rapid Application Development Methodology dan Spiral.

## 1.12. Komponen Struktur Algoritma Pemrograman

**Struktur Algoritma** merupakan ketentuan yang biasa dipakai dalam membuat Algoritma ketentuan tersebut dapat berupa runtunan aksi (Sequence), pemilihan aksi (Selection), pengulangan aksi (Loop) atau kombinasi dari ketiganya. Jadi struktur dasar pembangunan algoritma ada tiga, yaitu:

### 1.12.1. Sequence

Sebuah proses yang berisi instruksi yang berurutan. Di dalam sequence setiap instruksi berjalan sesuai dengan urutan penulisannya, jadi instruksi akan berjalan jika instruksi atas nya sudah dijalankan dan setiap perurutan instruksi memiliki maksud kenapa harus dijalankan terlebih dahulu agar instruksi setelahnya bisa di jalankan.



Contoh dalam kehidupan sehari-hari :

Memasak mie harus berurutan stepnya. seperti kita membuka bungkus mie terlebih dahulu baru memasukkan mienya ke dalam air yang sudah mendidih. Jika kita tidak membuka bungkusnya terlebih dahulu artinya tidak sesuai dengan aturan dari program yang di buat.

### 1.12.2. Selection

Perkembangan dari sequence output dari selection ini berupa boolean yang arti nya hanya ada 2 kemungkinan yaitu true and false ( ya dan tidak ). Dan setiap hasil yang di keluarkan harus lah memiliki akhir dan kembali pada tujuan utama ya itu stop, agar program berjalan.

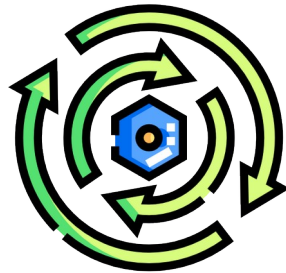


Contoh dalam kehidupan sehari-hari :

Saat kita lapar dan kita hendak memasak mie instan, pertama sebelum kita membuat mie pasti kita akan mengecek dahulu mienya ada atau tidak, nah kalau tidak ada kita beli dulu lalu kita masak dan kalau mienya ada langsung saja kita masak.



### 1.12.3. Loop



Struktur pemograman yang akan dilakukan terus menerus jika suatu kondisi belum memenuhi syarat.

Contoh :

Ketika makan kita akan melakukan kegiatan yang berulang yaitu memasukkan nasi kedalam mulut terus sampai dengan kondisi merasa sudah kenyang maka kita akan berhenti.

### 1.12.4. Exercise

- Buatlah Algoritma Penerimaan Siswa Baru di sebuah Universitas (Jalur Tes Mandiri).
- Buatlah Algoritma Mengitung Luas Segitiga.
- Buatlah Algoritma memasak nasi di rice cooker.

