MEMBUAT WEB DENGAN MUDAH MENGGUNAKAN HTML,CSS DAN JAVASCRIPT

BY M.LUTFI BADILA

PENGENALAN WEBSITE

**Apa itu website ?**

Website adalah kumpulan halaman yang berisi suatu informasi tertentu yang kemudain bisa di akses oleh siapapun, dimanapun dan kapanpun dengan mudah melalui koneksi internet. Proses pembuatan website salah satunya dapat dilakukan dengan dengan pemrograman web atau menuliskan sebuah intruksi–intruksi yang di berikan kepada komputer untuk membuat suatu tugas atau fungsi tertentu. Dengan kata lain Pemrograman Web merupakan cara atau Proses untuk menjalankan intruksi ataupun perintah kepada komputer yang terhubung dengan internet yang dapat digunakan untuk membuat tugas maupun fungsi lainnya. Kemudian ketika menjalankan sebuah program tersebut di dalam web dapat melaui web browser misalnya: Opera, mozilla, Chrome, dll.

**a) Jenis - jenis Website**

• Website Statis

Website Statis merupakan web yang isi atau kontennya tidak pernah di perbaharui atau di update dalam durasi waktu tertentu, atau bisa di katakana isinya tidak pernah berubah sama sekali, contoh website statis seperti web company profile suatu instansi atau perusahaan dll. Tujuan web ini yaitu user atau masyarakat dapat melihat tentang profil dari instansi atau perusahaan tersebut. Website statis ini cenderung tidak responsive, hal ini karena ketika akan mengupdate fitur dari webnya harus mengubah scriptnya atau perintah program secara langsung, tidak dapat dilakukan melalui CMS yang tersedia.

• Website Dinamis

Situs web yang dinamis adalah situs web yang konten atau elemen lainnya sering berubah karena pengguna atau pengunjung lain dapat memperbarui informasi situs secara diam-diam. Website dinamis ini biasanya cocok di gunakan untuk web blog, jejaring sosial, toko online dll. Pembuatan website ini membutuhkan waktu yang cukup lama karena menggunakan database untuk menyimpan semua data dari website yang bersangkutan.

• Website Interaktif

Website Interaktif adalah website yang hampir mirip dengan website yang bersifat statis; namun, interaksi di situs web interaktif lebih sering atau sering terjadi karena ditujukan untuk pengguna yang juga akan menggunakannya untuk mengedit konten di situs web tersebut. Tidak seperti website dinamis yang pilihannya hanya share atau tinggalkan komentar. Seorang pengguna atau pengguna juga dapat melakukan tindakan saat mengedit konten di situs web interaktif seperti YouTube.

**b) Development Web**

Dalam development web, terdapat istilah yang dinamakan Client - side serta Server - side Scripting. Dua istilah tersebut digunakan untuk mengelompokkan dimana bahasa atau kode 3 pemrograman tersebut dijalankan, apakah berjalan di sisi client atau di sisi server. Jika diartikan dalam arti lain, Client-side memiliki arti sisi pengguna dan Server-side memiliki arti sisi server atau penyedia. Berikut penjelasan singkat mengenao web client dan web server.

• Client Side Scripting

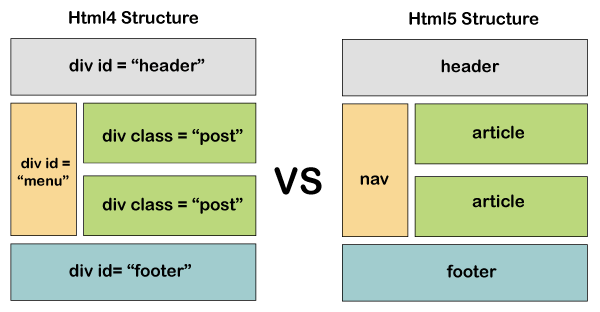
Klien web, sering dikenal sebagai skrip sisi klien, adalah bagian dari bahasa pemrograman web di mana pemrosesan sintaksis terjadi di sisi klien. Dalam proses klien web, browser web berfungsi sebagai klien. Namun ada juga perpustakaan yang tersedia di aplikasi browser web yang dapat digunakan untuk menulis ulang kode atau struktur sintaksis apa pun yang ditemukan di halaman web yang menggunakan skrip sisi klien. Akibatnya, dapat dikatakan bahwa skrip sisi klien mengacu pada aktivitas apa pun yang dilakukan di sisi pengguna atau klien (komputer pengguna). Contoh skrip sisi klien meliputi, antara lain, HTML, JavaScript, CSS, JQUERY, React, Vue, dll.

• Server Side Scripting

Yang dimaksud dengan “server-side scripting” adalah bahasa pemrograman web yang proses pengembangannya dilakukan di sisi server, yaitu web server yang terintegrasi dengan web engine. Tujuan dari mesin pencari online adalah untuk mencarisetiap kemungkinan frase nanti. Mesin web adalah komponen server web yang perlu diinstal di komputer sesegera mungkin. Ada skrip sisi server di situs web dinamis. Dalam skrip sisi server, kecepatan dan efisiensi komputer server adalah faktor penentu seberapa cepat situs web memuat atau beroperasi. Karena pada dasarnya bersifat rahasia, sintaks tertulis di sisi server tidak dapat dilihat di antarmuka klien atau pengguna. PHP, Active Server Pages (ASP), Lasso, Java Server Pages (JSP), dan SSI adalah contoh program skrip sisi server.

**c ) APA ITU HTML ?**

HTML, sering dikenal sebagai Hyper-Text Markup Language, adalah bahasa utama yang digunakan di seluruh dunia oleh situs web. Hampir semua website yang tersedia di internet dibuat dengan menggunakan variasi HTML. Pengembang dapat menambahkan teks, multimedia, dan tautan ke browser yang kompatibel dengan atribut atau elemen lain dengan menggunakan HTML. Dari elemen yang dapat terhubung ke dokumen Anda (hiperteks) hingga elemen yang membuat dokumen Anda lebih interaktif, semuanya terdapat dalam satu file HTML. Dengan kata lain, HTML (Hypertext Markup Language) adalah bahasa yang biasa digunakan untuk menandai dokumen yang akan dipublikasikan secara online. Dalam bahasa sederhana, HTML mengilustrasikan struktur halaman Web, yang terdiri dari serangkaian elemen HTML, untuk menjelaskan kepada browser web cara menampilkan konten yang telah dipilih. HTML memiliki standar yang dibuat oleh W3C (World Wide Web Consortium) pada tahun 1997 dan semakin lama semakin bermanfaat. Dalam HTML, ada tag yang digunakan untuk menunjukkan struktur. Tag dan elemen dapat dilindungi dengan menggunakan karakter < dan >. HTML bukanlah merupakan bahasa pemrograman tapi merupakan bahasa markup.



Perbandingan HTML 4 dan HTML 5

a) Tag dalam HTML

|  |  |  |
| --- | --- | --- |
| **Tag** | **Fungsi** | **Contoh** |
| <p></p> | Untuk membuat paragraf | p.jpg |
| <h1></h1> | Untuk membuat heading 1 | h1-h6.jpg |
| <h2></h2> | Untuk membuat heading 2 |
| <h3></h3> | Untuk membuat heading 3 |
| <h4></h4> | Untuk membuat heading 4 |
| <h5></h5> | Untuk membuat heading 5 |
| <h6></h6> | Untuk membuat heading 6 |
| Comment tag.jpg | Untuk menonaktifkan tag agara tidak dieksekusi | tag komen.jpg |
| <b></b> | Untuk membuat teks tebal | b-dan-strong.jpg |
| <strong></strong> |
|  |  |  |
|  |
|  |  |  |
| <i><i/> | Untuk membuat teks miring | i-dan-em.jpg |
| <em><em/> |
| <u><u/> | Untuk membuat teks underline | u-dan-ins.jpg |
| <ins><ins/> |
| <s><s/> | Untuk membuat teks tercoret | s-dan-del.jpg |
| <del><del/> |
| <sup><sup/> | Untuk membuat pangkat atas | sup.jpg |
| <sub><sub/> | Untuk membuat pangkat bawah | sub.jpg |
| <mark><mark/> | Untuk membuat maker dalam teks (warna kuning sebagai default) | mark.jpg |
| <a href=”isi dengan link”><a/> | Untuk menautkan link ke web tertentu | a.jpg |
| <img src=””><img/> | Untuk menampilkan gambar | img.jpg |
| <ol>  <li>  <li/>  <ol/> | Untuk membuat list berurutan | ol-li.jpg |
| <ul>  <li>  <li/>  <ul/> | Untuk membuat list tidak berurutan | ul-li.jpg |
| <video>  <source src="nama file" type="video/tipe file"> <video/> | Untuk menambahkan video | video.jpg |
| <audio>  <source src="nama file " type="video/tipe file ">  < audio /> | Untuk menambahkan audio | audio.jpg |
| <form>  <p>  <label>username</label>  <input type="text" name="" placeholder="nama">  </p>  <p>  <label>password</label>  <input type="password" name="" placeholder="password"> </p>  </form> | Untuk membuat form | form.jpg |
| Explaining-table.jpg | | |
| <th><th/> | Untuk membuat judul pada header | |
| <th rowspan><th/> | Untuk membuat judul pada header dengan menggabungkan baris | |
| <th colspan><th/> | Untuk membuat judul pada header dengan menggabungkan kolom | |
| <tr><tr/> | Untuk membuat baris | |
| <td><td/> | Untuk membuat sel | |
| <table><table/> | Untuk membuat tabel | |

**CASCADING STYLE SHEETS (CSS)**

Singkatan CSS adalah singkatan dari "Cascading Style Sheets". Sesuai dengan namanya, CSS memiliki pengertian “style sheet language” dalam bahasa pemrograman yang digunakan untuk pengembangan web. CSS adalah bahasa pemrograman yang digunakan untuk membuat website header. CSS menggunakan id dan penanda yang dapat dikenali kelas saat mendesain header situs web. Di bidang pemrograman dan teknologi yang berkembang pesat , CSS tidak lagi hanya digunakan dengan HTM, namun sudah bisa digunakan untuk mengembangkan tampilans aplikasi Android . CSS dijalankan menggunakan id atau kelas yang diambil dari html untuk memeprcantik tampilan .

Fungsi CSS

• Mempercantik variasi tampilan

• Membuat website tampil lebih rapi di berbagai ukuran layar

• Mempercepat Loading Halaman Web

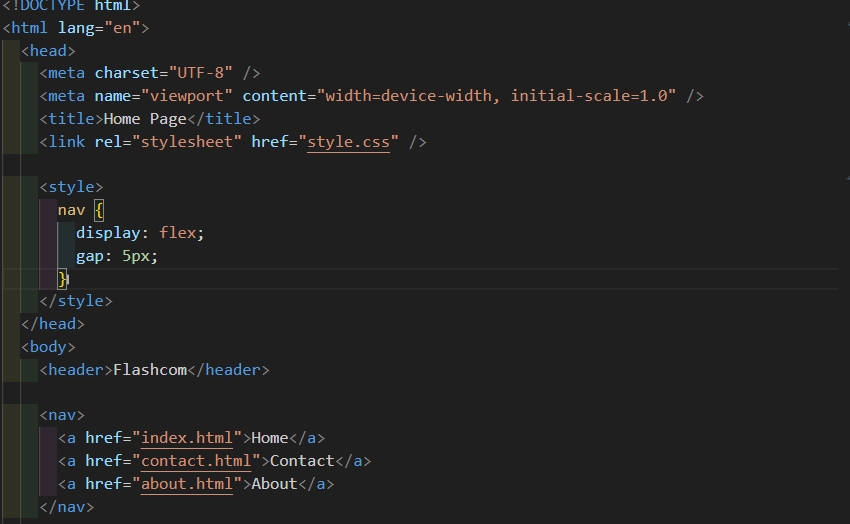
• Memudahkan pengelolaan kode

**Cara Menulis CSS di HTML**

Ada dua cara untuk menambahkan CSS di HTML :

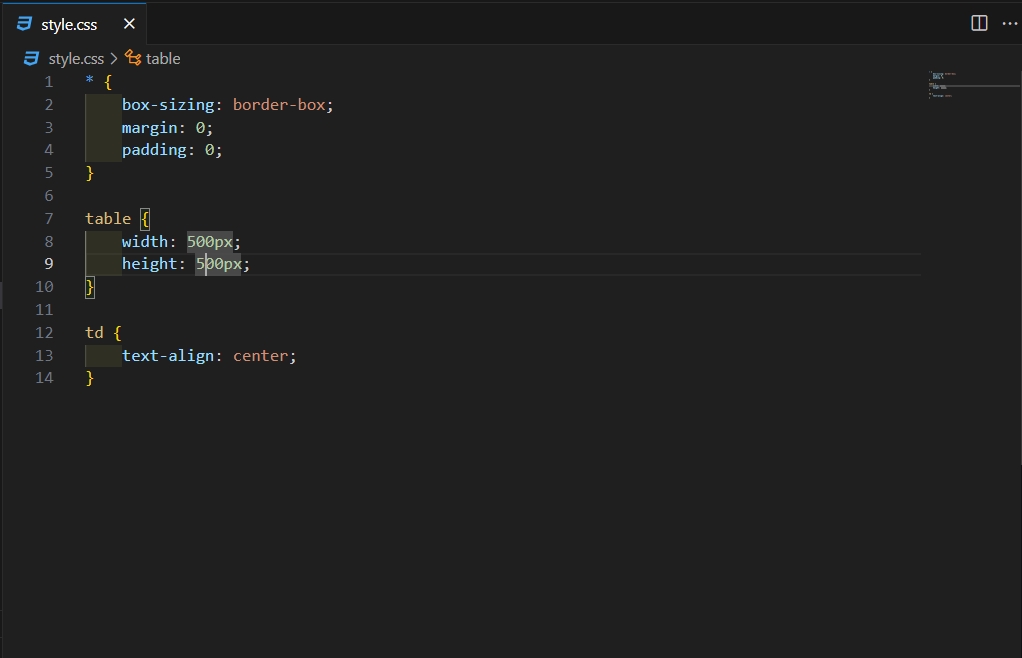
1. Intenal CSS

Yaitu dengan menulis CSS langsung di file HTML. Contohnya sebagai berikut:

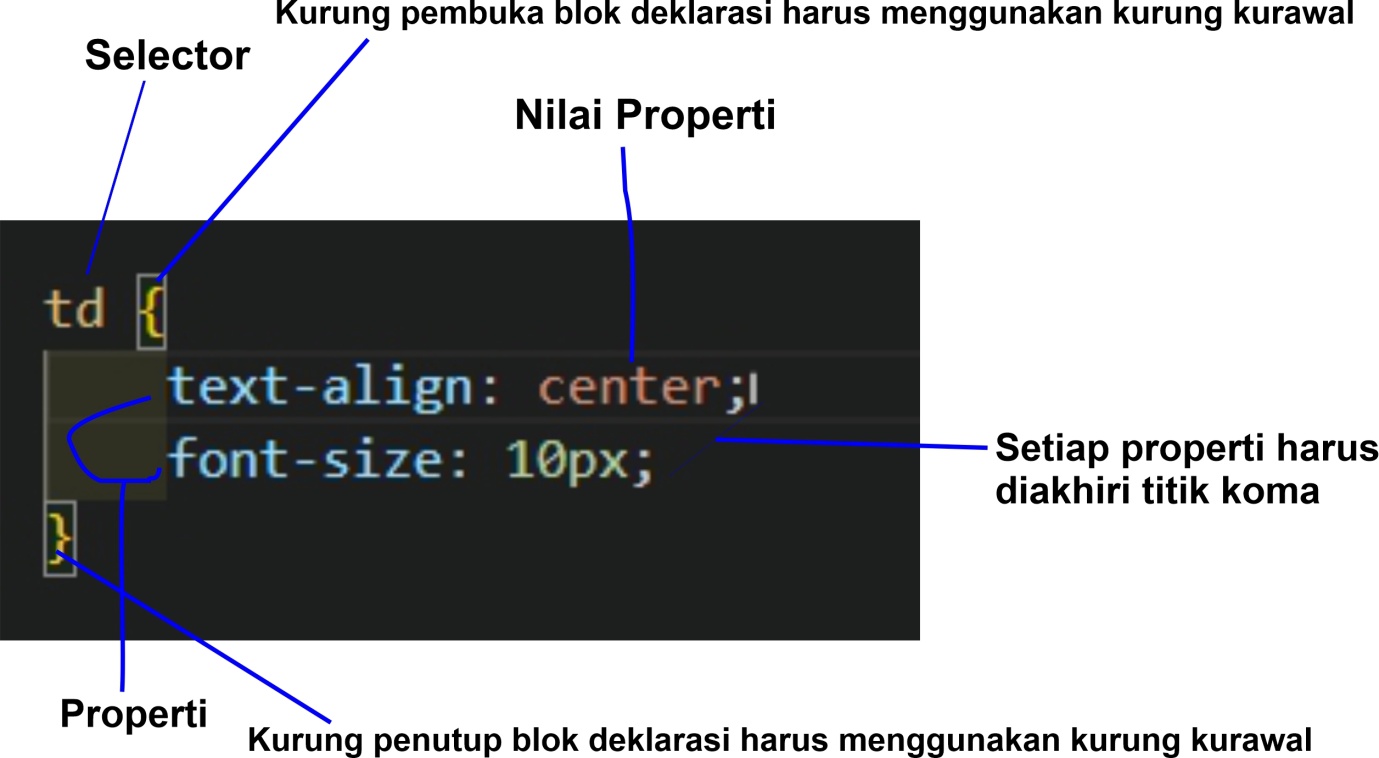


2.Eksternal CSS

Yaitu dengan menulis CSS diluar file HTML yang berarti membuat file tersendiri untuk CSS tersebut dan cara ini lebih sering digunakan. Contohnya sebagai berikut:



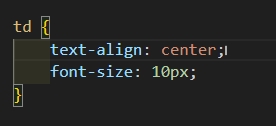
**Tata Cara Untuk Menulis CSS**

****

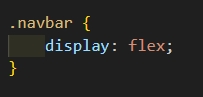
**Selektor dalam CSS**

Ada 6 jenis selektor dalam CSS:

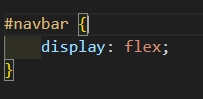
1. Selektor Tag:



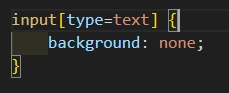
1. Selektor Class:



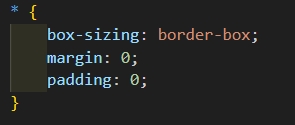
1. Selektor ID:



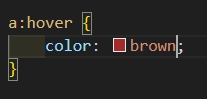
1. Selektor Atribut:



1. Selektor Universal:



6. Selektor Pseudo:



**Merubah Warna teks dan background di CSS**

Kita bisa merubah warna teks denan menggunakan properti **color** lalu tambahkan value nama yang diinginkan dan jika merubah background bisa menggunakan properti **background.**

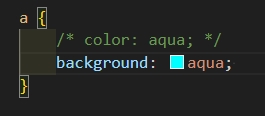
Contoh untuk teks:



Maka akan menjadi:

color.jpg

Contoh untuk background:



Maka akan menjadi:

**bg-text.jpg**

**Text Align di CSS**

Ada beberapa macam text align di css diantaranya:

**left**: untuk rata kiri;

**right:** untuk rata kanan;

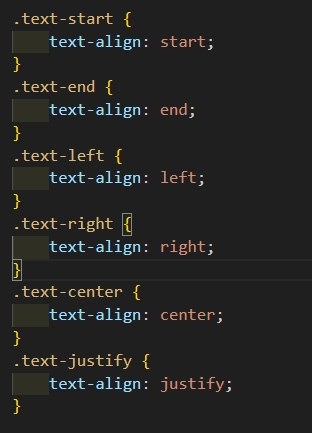
**center:** untuk rata tengah;

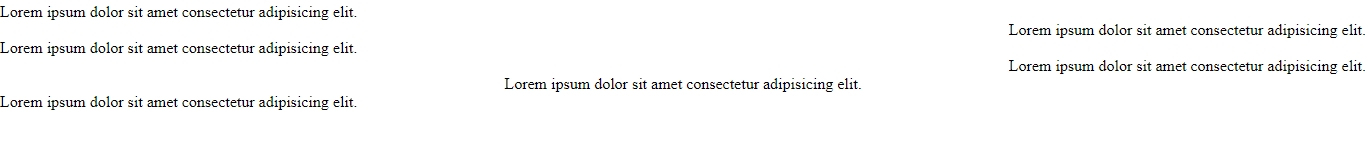
**justify:** untuk rata kiri dan kanan;

**start:** rata pada awal (mengikuti direction);

**end:** rata pada akhir (mengikuti direction);

Contohnya:



Hasilnya:

**Text Decoration di CSS**

Yaitu bisa memberikan dokorasi untuk teks, ada beberapa macam protperti, seperti:

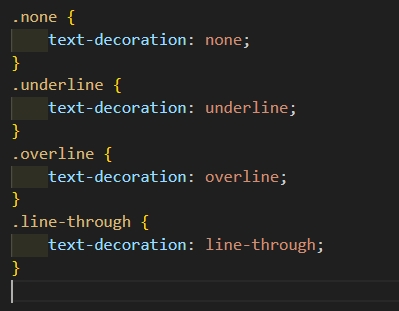
**none** artinya tidak menggunakan dekorasi apapun;

**underline** artinya kita akan memberikan dekorasi garis bawah;

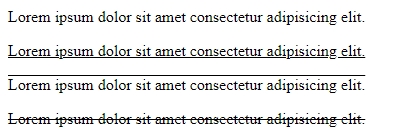
**overline** dekorasi dengan garis di atas teks;

**line-through** dekorasi garis di tengah atau teks tercoret.

Contohnya:



Hasilnya:



Text Transform di CSS

Yaitu untuk mengubah bentuk teks, ada beberapa value yang bisa digunakan di text transform yaitu:

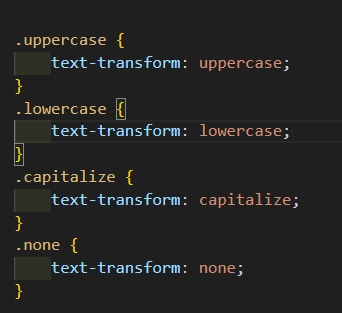
**lowercase** untuk mengubah semua teks menjadi huruf kecil;

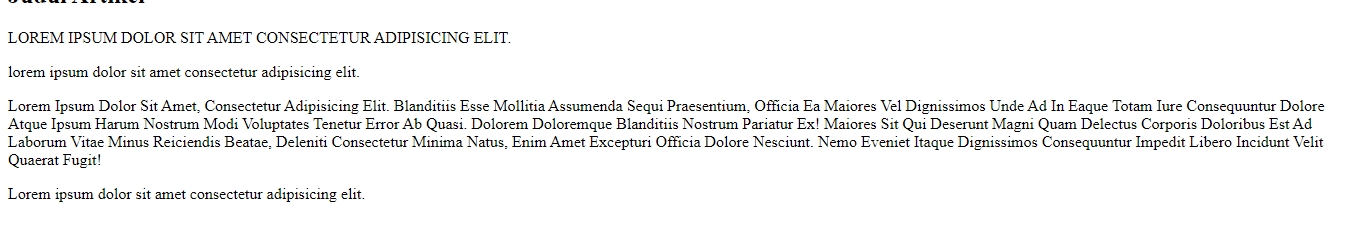
**uppercase** untuk mengubah semua teks menjadi huruf kapital (besar);

**capitalize** untuk mengubah teks pada huruf pertama kata menjadi kapital.

**none** untuk tidak melakukan transform.

Contohnya:



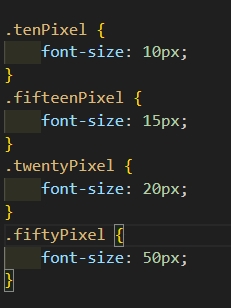
Hasilnya:

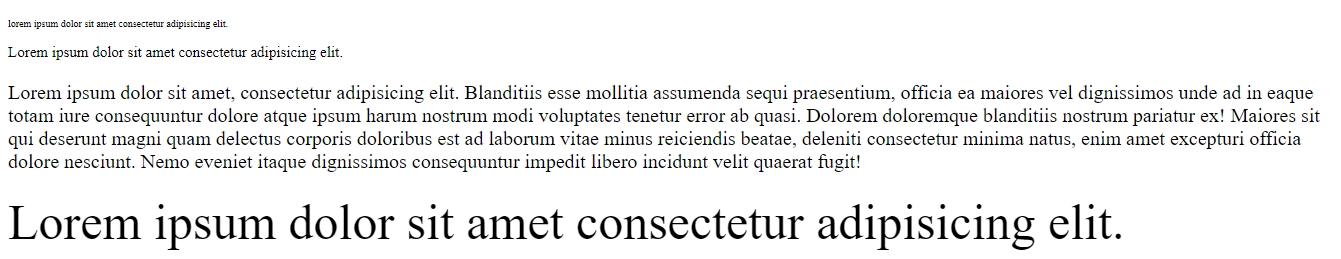
**Mengatur font di CSS**

Yaitu untuk mengatur font pada teks, seperti:

**font-size** untuk mengatur ukuran font;

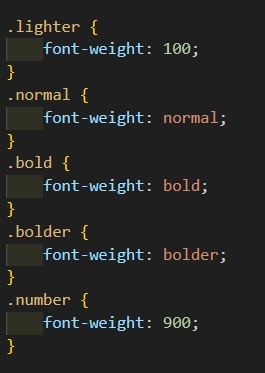
contohnya:



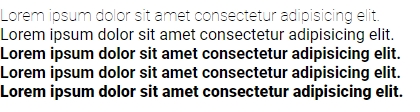
Hasilnya:

**font-weight** untuk mengatur ketebalan font;

contohnya:



Hasilnya:

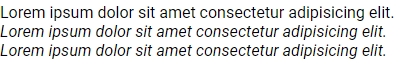


**font-style** untuk mengatur gaya font;

contohnya:



Hasilnya:

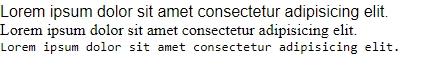


**font-family** untuk mengatur jenis font;

contohnya:

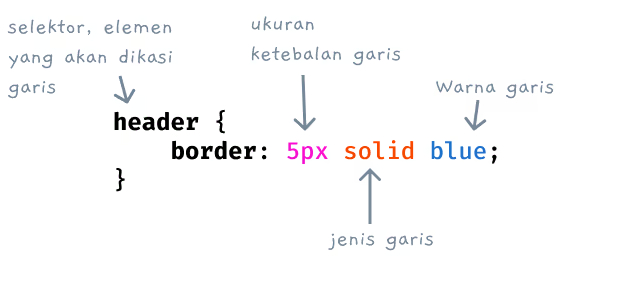


Hasilnya

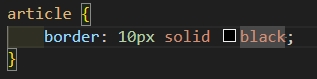
****

**Membuat Garis di CSS**

Kita akan membuat garis beserta mengatur ukuran, warna, mengenal jenis-jenis garis, memberikan garis pada sisi tertentu, membuat Garis yang pojoknya tumpul, membuat Garis dengan Gambar.



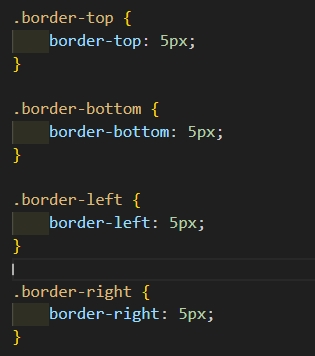
Contoh untuk membuat garis:



Hasilnya:



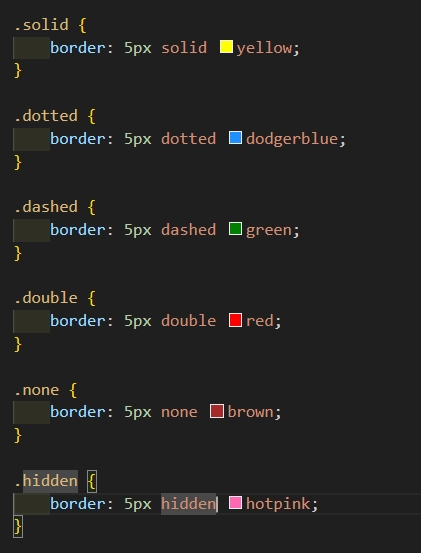
Kita juga bisa memberi garis pada sisi tertentu, contohnya:

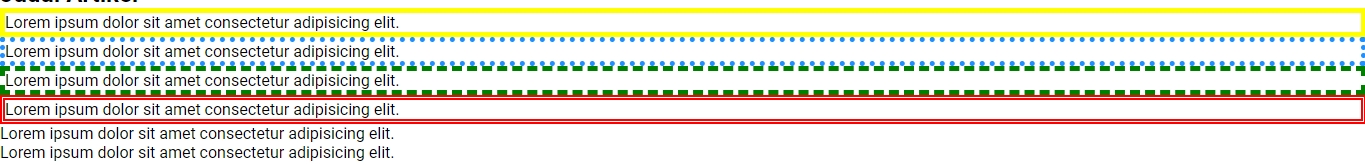


Hasilnya:

Macam-macam garis border

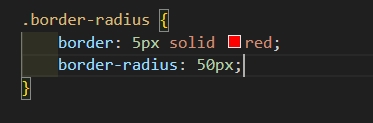
Contohnya:



Hasilnya:

Membuat Pojok Garis Tumpul

Contohnya:



Hasilnya:

result-border-raidus.jpg

**CSS Lanjutan**

**Animasi**

## Apa itu CSS Animation ?

CSS Animation memungkinkan elemen web **berubah secara halus dari satu gaya ke gaya lainnya** tanpa JavaScript.

## 1. ****Properti Utama untuk Animasi****

### keyframes

Digunakan untuk **mendefinisikan langkah-langkah** dalam animasi.

### animation shorthand atau properti lengkap seperti:

* animation-name
* animation-duration
* animation-timing-function
* animation-delay
* animation-iteration-count
* animation-direction
* animation-fill-mode

## 2. Contoh Dasar: Fade In

Html:

<div class="fade-in-box">Halo!</div>

css:

.fade-in-box {

opacity: 0;

animation: fadeIn 2s forwards;

}

@keyframes fadeIn {

to {

opacity: 1;

}

}

## 3. Contoh Bergerak dari Kiri ke Kanan

.move-right {

position: relative;

animation: moveRight 3s ease-in-out infinite;

}

@keyframes moveRight {

0% { left: 0; }

50% { left: 200px; }

100% { left: 0; }

}

## 4. Contoh Hover Animasi: Zoom

<div class="zoom-box">Hover Me</div>

css

SalinEdit

.zoom-box {

transition: transform 0.3s ease;

}

.zoom-box:hover {

transform: scale(1.1);

}

## 5. Contoh Rotation (Putar)

.spin {

animation: spin 2s linear infinite;

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

## 6. Animasi dengan Delay dan Iterasi

.blink {

animation-name: blink;

animation-duration: 1s;

animation-delay: 0.5s;

animation-iteration-count: infinite;

animation-timing-function: ease-in-out;

}

@keyframes blink {

0%, 100% { opacity: 1; }

50% { opacity: 0; }

}

## 7. Menggunakan animation shorthand

.box {

animation: slideDown 0.5s ease-out 0s 1 normal forwards;

}

@keyframes slideDown {

from { transform: translateY(-100px); opacity: 0; }

to { transform: translateY(0); opacity: 1; }

}

Format:

animation: name duration timing-function delay iteration direction fill-mode;

**Layouting**

# 1. Layouting dengan float

### Penjelasan:

float digunakan untuk memposisikan elemen ke kiri atau kanan dalam satu baris. Ini adalah metode **lama** sebelum ada Flexbox dan Grid.

### Contoh: 2 Kolom (Sidebar dan Konten)

html

<div class="container">

<div class="sidebar">Sidebar</div>

<div class="content">Konten Utama</div>

</div>

css

.container {

width: 100%;

overflow: hidden; /\* clear float \*/

}

.sidebar {

float: left;

width: 30%;

background: #ccc;

height: 200px;

}

.content {

float: right;

width: 70%;

background: #eee;

height: 200px;

}

Gunakan clear: both atau overflow: hidden pada kontainer untuk menghindari "collapse".

# 2. Layouting dengan Flexbox

### Penjelasan:

Flexbox memudahkan untuk membuat layout **baris atau kolom** dengan elemen yang fleksibel dan responsif.

### ✅ Contoh: Navbar Horizontal

html

<nav class="nav">

<div>Home</div>

<div>About</div>

<div>Contact</div>

</nav>

css

.nav {

display: flex;

justify-content: space-around;

background: #333;

color: white;

padding: 10px;

}

### Contoh: 3 Kolom Responsive

html

<div class="flex-container">

<div class="box">1</div>

<div class="box">2</div>

<div class="box">3</div>

</div>

css

.flex-container {

display: flex;

gap: 10px;

flex-wrap: wrap;

}

.box {

flex: 1;

min-width: 200px;

background: #ddd;

padding: 20px;

}

Flex digunakan untuk **layout satu dimensi** (baris ATAU kolom).

# 3. Layouting dengan CSS Grid

### Penjelasan:

Grid digunakan untuk membuat **layout dua dimensi** (baris dan kolom), cocok untuk desain kompleks.

### Contoh: 3 Kolom Grid

html

<div class="grid-container">

<div class="grid-item">A</div>

<div class="grid-item">B</div>

<div class="grid-item">C</div>

</div>

css

.grid-container {

display: grid;

grid-template-columns: repeat(3, 1fr); /\* 3 kolom sama rata \*/

gap: 10px;

}

.grid-item {

background: #bbb;

padding: 20px;

text-align: center;

}

### Contoh: Grid Layout Majalah

html

<div class="magazine">

<div class="header">Header</div>

<div class="menu">Menu</div>

<div class="main">Main</div>

<div class="ads">Ads</div>

<div class="footer">Footer</div>

</div>

css

.magazine {

display: grid;

grid-template-areas:

"header header header"

"menu main ads"

"footer footer footer";

grid-template-columns: 1fr 3fr 1fr;

gap: 10px;

}

.header { grid-area: header; background: #ddd; }

.menu { grid-area: menu; background: #aaa; }

.main { grid-area: main; background: #ccc; }

.ads { grid-area: ads; background: #bbb; }

.footer { grid-area: footer; background: #999; }

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabel Perbandingan** | | | |
| **Fitur** | **Float** | **Flexbox** | **Grid** |
| Dimensi | 1D | 1D (baris/kolom) | 2D (baris & kolom) |
| Responsive | Sulit | Mudah | Sangat fleksibel |
| Kontrol posisi | Terbatas | Baik | Sangat baik |
| Cocok untuk | Layout dasar | Navbar, baris item | Galeri |
| Clearfix diperlukan | Ya | Tidak | Tidak |

**Javascript**

**1. Pengenalan JavaScript**

JavaScript adalah bahasa pemrograman yang digunakan terutama untuk membuat halaman web interaktif. Bahasa ini berjalan di sisi klien (browser) dan dapat mengontrol elemen HTML serta CSS.

**Contoh**:

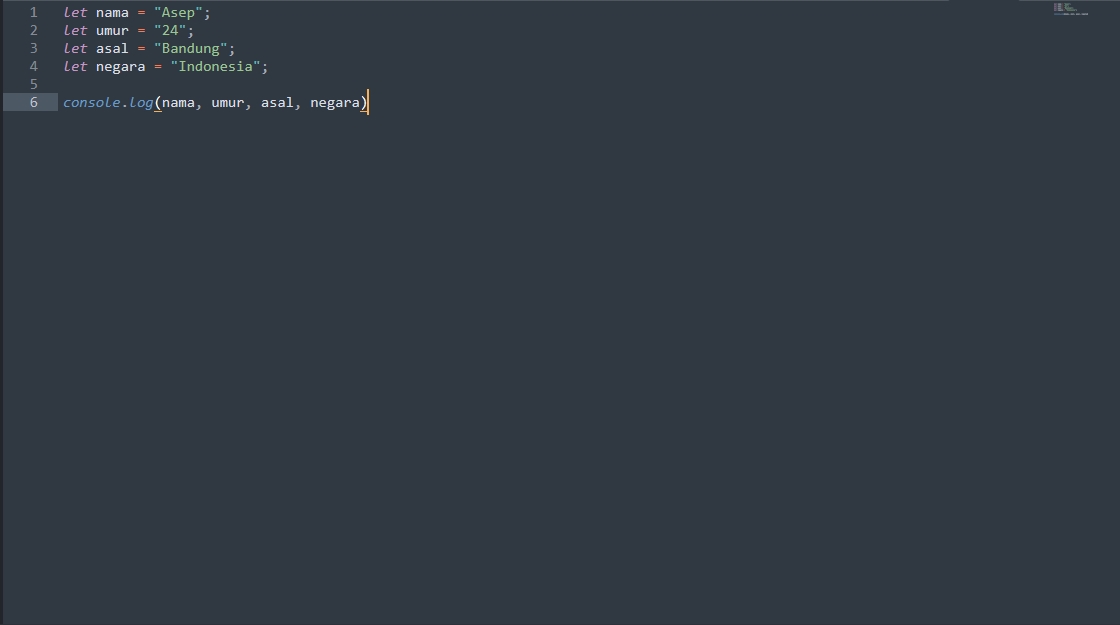


**2. Variabel**

Variabel digunakan untuk menyimpan data. Dalam JavaScript, ada tiga kata kunci untuk mendeklarasikan variabel: var, let, dan const.

* **var**: dapat diubah dan memiliki cakupan global atau fungsi.
* **let**: dapat diubah dan memiliki cakupan blok.
* **const**: tidak dapat diubah (konstan) dan memiliki cakupan blok.

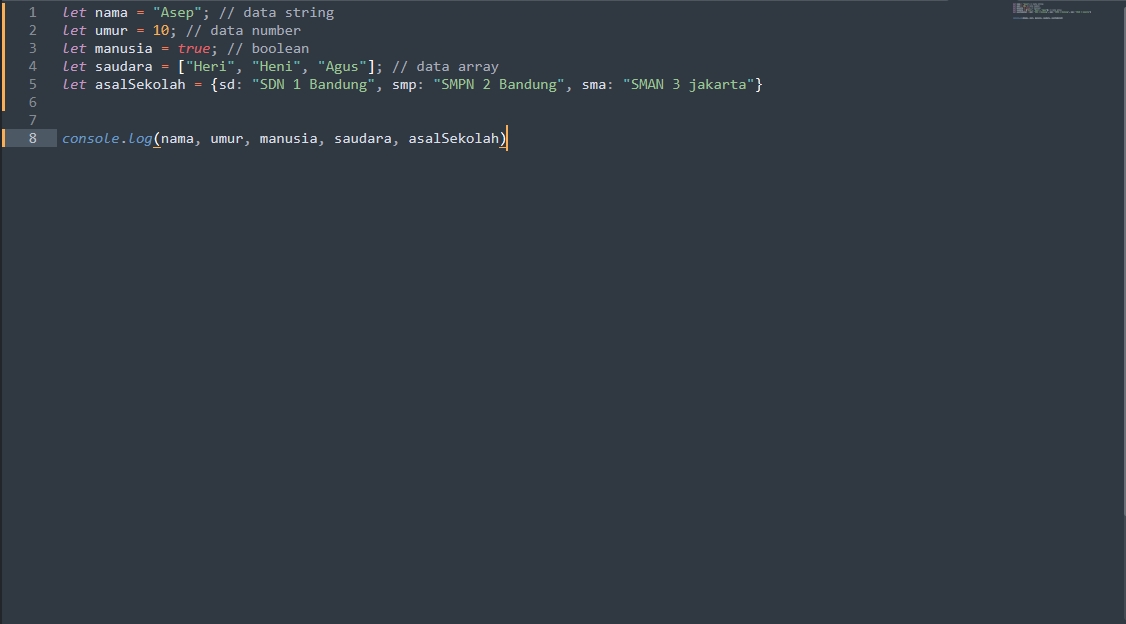
**Contoh:**



### ­­­­ 3. Tipe Data

JavaScript memiliki beberapa tipe data dasar:

* **String**: teks, contoh: "hello"
* **Number**: angka, contoh: 5, 3.14
* **Boolean**: true atau false
* **Object**: struktur data yang kompleks
* **Array**: daftar yang dapat menyimpan banyak nilai

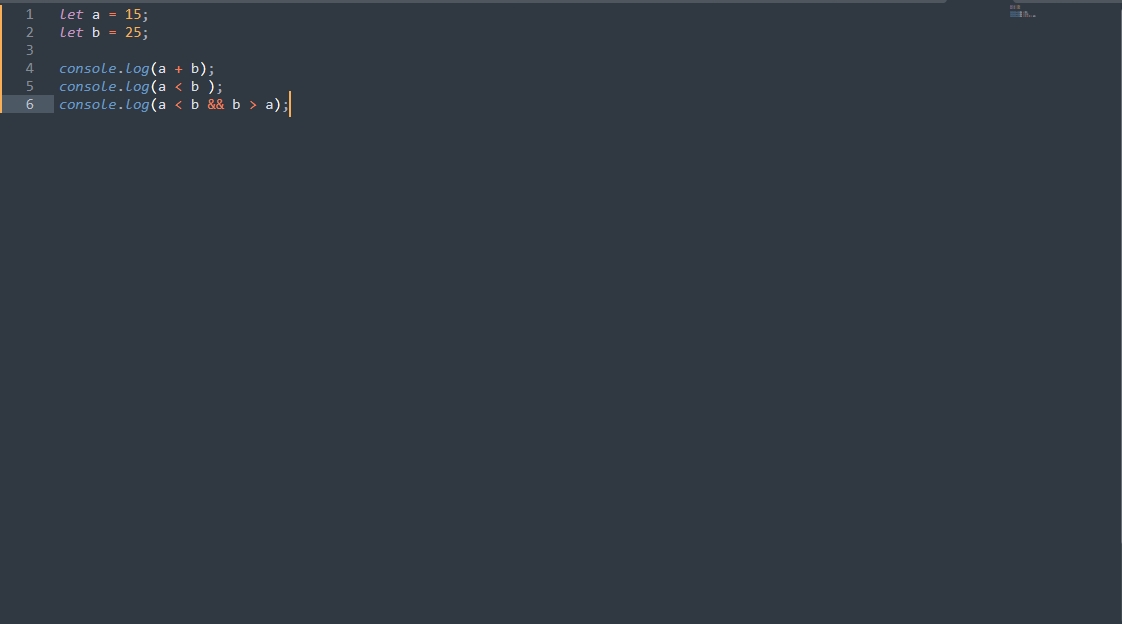
Contoh:

**4. Operator**

JavaScript memiliki berbagai jenis operator, seperti:

* **Operator Aritmatika** (+, -, \*, /, %)
* **Operator Perbandingan** (==, ===, !=, !==, >, <, >=, <=)
* **Operator Logika** (&&, ||, !)

Contoh :



### 5. Kondisional

Pernyataan kondisional memungkinkan kita menjalankan kode berdasarkan kondisi tertentu. Pernyataan kondisional yang umum adalah if, else if, dan else.

**Contoh**:

**6. Perulangan (Looping)**

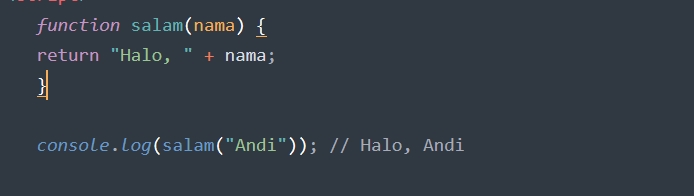
JavaScript menyediakan beberapa jenis perulangan:

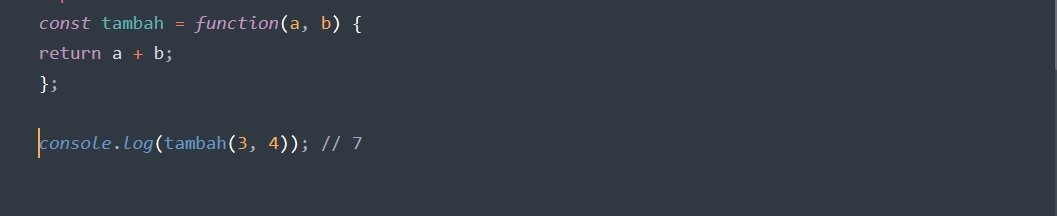
* **for** loop ketika jumlah iterasi sudah diketahui.
* **while** loop jika kondisi lebih fleksibel atau jumlah iterasi tidak diketahui.
* **do...while** loop ketika Anda ingin menjalankan kode setidaknya sekali.
* **for...of** loop untuk mengakses elemen dari array atau iterable.
* **for...in** loop untuk mengakses properti dari objek, namun hindari pada array karena bisa memberikan hasil yang tidak terduga.



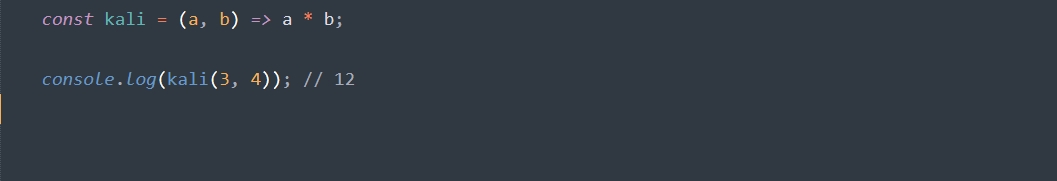
#### 7. **Fungsi (Function)**

Fungsi adalah blok kode yang dapat dipanggil berulang kali untuk melakukan tugas tertentu.

* **Fungsi Deklaratif:**
* **Fungsi Ekspresi:**

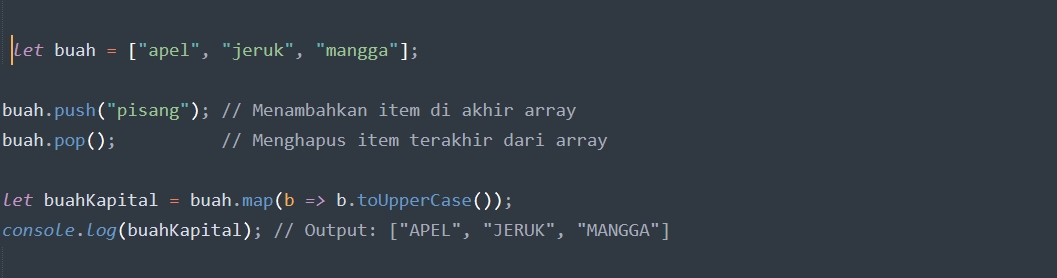
****

* **Arrow Function**



### 8. Array dan Method Array

Array adalah struktur data untuk menyimpan banyak nilai dalam satu variabel. Ada beberapa metode untuk mengolah array, seperti push, pop, map, filter, dan reduce.



### Ada beberapa metode dalam array:

### .length

### Untuk meghitung jumlah elemen di dalam array.berikut contohnya:

### Screenshot_37.jpg

### .push()

### Untuk menambahkan elemen baru di akhir array dan mengembalikan panjang array setelah penambahan.

### Screenshot_38.jpg

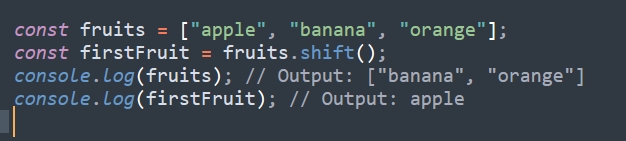
### .pop()

### Menghapus elemen terakhir dan mengembalikan elemen yang dihapus

### Screenshot_39.jpg

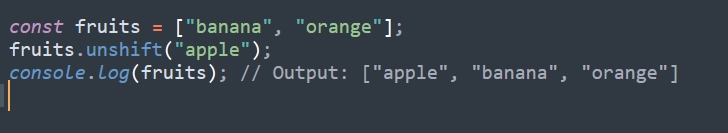
1. .shift()

Menghapus elemen pertama dari array dan mengembalikan elemen yang dihapus.



1. .unshift()

Menambahkan elemen baru ke awal array dan mengembalikan panjang array setelah penambahan.



### .splice()

### Menambahkan, menghapus, atau mengganti elemen di posisi tertentu dalam array.

### Screenshot_42.jpg

### .

### slice()

### Mengembalikan bagian dari array sebagai array baru tanpa mengubah array asli.

### Screenshot_43.jpg

### .concat()

### Menggabungkan dua atau lebih array dan mengembalikan array baru.

### Screenshot_44.jpg

### .indexOf() dan .lastIndexOf()

### Mencari indeks pertama atau terakhir dari elemen tertentu dalam array

### Screenshot_45.jpg

### .includes()

### Memeriksa apakah array mengandung elemen tertentu, mengembalikan true atau false.

### Screenshot_46.jpg

### .forEach()

### Menjalankan fungsi tertentu untuk setiap elemen dalam array.

### Screenshot_47.jpg

### .map()

### Membuat array baru dengan hasil dari pemanggilan fungsi tertentu pada setiap elemen dalam array.

### Screenshot_48.jpg

### .filter()

### Membuat array baru yang berisi semua elemen yang lulus uji dalam fungsi yang diberikan.

### Screenshot_49.jpg

### .reduce()

### Mengakumulasi nilai dari array ke dalam satu nilai berdasarkan fungsi yang diberikan.

### Screenshot_50.jpg

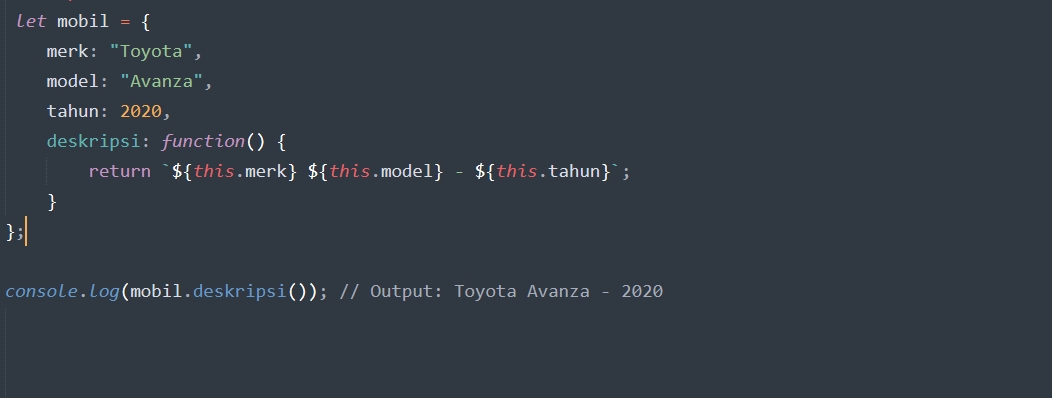
### .sort()

### Mengurutkan elemen array sesuai urutan leksikografis atau sesuai dengan fungsi pembanding.

### Screenshot_52.jpg

### 9. Objek

Objek adalah koleksi pasangan kunci-nilai. Objek sering digunakan untuk merepresentasikan entitas dengan properti tertentu.



### 10. JavaScript DOM (Document Object Model)

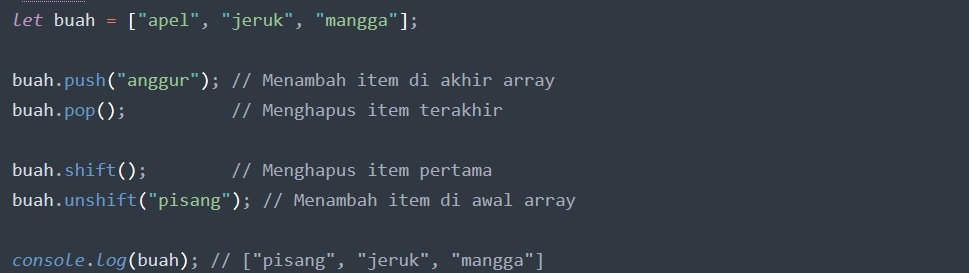
DOM adalah antarmuka untuk mengakses dan memanipulasi elemen HTML melalui JavaScript.



#### 11. **Array dan Metode Array**

Array adalah tipe data yang menyimpan banyak nilai dalam satu variabel.

**Contoh Metode Array:**



#### 12. **Fetch API**

fetch() digunakan untuk melakukan permintaan HTTP.



## ****DOM Manipulation with JavaScript****

DOM (**Document Object Model**) adalahsuatu cara untuk memanipulasi html melalu javascript agar bisa mengakses dan mengubah html kosong menjadi html biasa.

#### 1. ****Menyeleksi Element****

#### DOMJS1.png

#### 2. ****Mengubah Text dan HTML****

#### DOMJS2.png

#### 3. ****Mengubah atau Menambah CSS****

#### DOMJS3.png

#### 4. ****Mengubah Atribut****

#### DOMJS4.png

#### 5. ****Membuat dan Memasukkan Element HTML****

#### DOMJS5.png

#### 6. ****Event Handling****

#### DOMJS6.png

Belajar Backend

Apa itu Backend ?

Backend, dalam konteks pengembangan aplikasi (seperti website atau aplikasi mobile), adalah bagian sisi server yang tidak terlihat oleh pengguna. Ini adalah tulang punggung yang memproses data, menjalankan logika bisnis, dan mengelola basis data, semuanya di balik layar. Backend bekerja sama dengan frontend (bagian yang dilihat dan digunakan pengguna) untuk memastikan aplikasi berfungsi dengan baik.

* **Server**

Server adalah sebuah sistem komputer yang menyediakan layanan atau sumber daya kepada komputer lain (klien) dalam sebuah jaringan. Server dirancang untuk menangani permintaan dari banyak klien secara bersamaan dan beroperasi sepanjang waktu. Dalam konteks internet, server digunakan untuk menyimpan dan mengelola data, serta menyediakan layanan seperti akses website, email, dan lainnya.

**Fungsi Utama:**

Server bertindak sebagai pusat dalam jaringan, menyediakan akses ke berbagai sumber daya seperti data, aplikasi, dan layanan.

**Perangkat Keras dan Lunak:**

Server dapat berupa perangkat keras (hardware) khusus atau perangkat lunak (software) yang berjalan di komputer.

Contohnya:

* Hardware

Ini adalah bagian fisik dari server, seperti CPU, RAM, hard drive, motherboard, dan komponen lainnya. Hardware ini menyediakan infrastruktur fisik untuk menjalankan perangkat lunak server dan menyediakan layanan.

* Software

Ini adalah program atau aplikasi yang berjalan di atas hardware server. Perangkat lunak server, seperti sistem operasi jaringan (misalnya, Linux atau Windows Server) dan aplikasi server (misalnya, web server seperti Apache atau Nginx), mengelola dan menyediakan layanan ke klien dalam jaringan.

**Contoh Penggunaan:**

Server Web: Menyediakan halaman web kepada pengguna melalui browser.

Server Database: Menyimpan dan mengelola data dalam format terstruktur.

**Cara Kerja:**

Server menerima permintaan dari klien, memproses permintaan tersebut, dan mengirimkan hasilnya kembali kepada klien.

**Perbedaan dengan Komputer Client:**

Server dirancang untuk menangani beban kerja yang lebih berat dan bekerja terus-menerus, sementara komputer client digunakan untuk tugas-tugas sehari-hari.

Jadi, server adalah tulang punggung dari berbagai layanan yang kita gunakan sehari-hari, mulai dari mengakses website hingga menggunakan aplikasi online.

* **Hosting**

Hosting adalah layanan yang menyediakan tempat penyimpanan dan akses online untuk data website, seperti file, gambar, video, dan database. Ini memungkinkan website dapat diakses oleh pengguna lain melalui internet. Jadi, hosting bisa diibaratkan seperti tanah tempat bangunan rumah (website) berdiri.

Berikut adalah penjelasan lebih lanjut mengenai hosting:

**Fungsi Hosting:**

**Penyimpanan Data Website:**

Hosting menyediakan ruang penyimpanan untuk semua file yang dibutuhkan website, termasuk kode, gambar, video, dan teks.

**Akses Online:**

Hosting memastikan website dapat diakses oleh pengguna dari seluruh dunia melalui internet.

**Pengelolaan Website:**

Hosting menyediakan alat dan fitur untuk mengelola website, seperti kontrol panel untuk mengelola file, basis data, dan akun email.

**Keamanan Data:**

Hosting yang baik menyediakan keamanan data untuk melindungi website dari ancaman online.

**Jenis-jenis Hosting:**

**Shared Hosting:**

Beberapa website berbagi sumber daya server yang sama, cocok untuk website dengan trafik rendah.

**VPS Hosting:**

Sumber daya server dibagi menjadi beberapa bagian virtual, memberikan lebih banyak kontrol dan fleksibilitas daripada shared hosting.

**Dedicated Hosting:**

Pengguna menyewa seluruh server, memberikan kontrol penuh dan performa tinggi.

**Cloud Hosting:**

Website disimpan di beberapa server virtual yang terhubung, menawarkan skalabilitas dan keandalan yang tinggi.

**Cara Kerja Hosting:**

Penyewaan Layanan: Pengguna membeli layanan hosting dari penyedia hosting.

Penyimpanan File: File website diunggah ke server hosting menggunakan FTP atau panel kontrol.

Akses Website: Saat pengguna mengakses website, browser akan mengirimkan permintaan ke server hosting untuk menampilkan konten website.

Perbedaan Hosting dan Domain:

Hosting: Tempat penyimpanan website.

Domain: Alamat website (misalnya, www.contoh.com) yang mengarahkan pengguna ke server hosting.

Jadi, hosting dan domain bekerja sama untuk membuat website dapat diakses dan dioperasikan secara online.

* **Database**

Database adalah kumpulan data terstruktur yang disimpan secara elektronik dalam sistem komputer. Data ini diorganisasi sedemikian rupa sehingga mudah diakses, dikelola, dimodifikasi, dan diperbarui. Secara sederhana, bayangkan database sebagai sebuah lemari arsip digital yang teratur, memudahkan kita untuk mencari dan menemukan informasi yang kita butuhkan.

Fungsi utama database:

Penyimpanan Data:

Database berfungsi sebagai tempat penyimpanan data yang terpusat dan terorganisir.

Pengelolaan Data:

Database memungkinkan pengguna untuk mengelola data, termasuk menambah, menghapus, dan memperbarui informasi.

Akses Data Cepat:

Database dirancang untuk memungkinkan akses data yang cepat dan efisien, seringkali melalui bahasa query khusus seperti SQL.

Keamanan Data:

Database dapat dilengkapi dengan fitur keamanan untuk melindungi data dari akses yang tidak sah.

Contoh Penggunaan Database:

Sistem Informasi Akademik: Menyimpan data mahasiswa, mata kuliah, nilai, dan informasi akademik lainnya.

Sistem Informasi Kepegawaian: Menyimpan data karyawan, gaji, absensi, dan informasi personal lainnya.

Sistem Informasi Kesehatan: Menyimpan data pasien, riwayat penyakit, dan informasi medis lainnya.

Sistem Informasi Penjualan: Menyimpan data produk, pelanggan, transaksi, dan informasi penjualan lainnya.

Jenis-jenis Database:

Ada berbagai jenis database, yang masing-masing memiliki karakteristik dan kegunaan yang berbeda. Beberapa jenis database yang umum adalah: database relasional, database NoSQL, database grafik, dan database berorientasi objek.

Berikut adalah penjelasan lebih rinci tentang jenis-jenis database tersebut:

1. Database Relasional:

Karakteristik:

Data disimpan dalam tabel dengan baris dan kolom. Setiap tabel mewakili entitas tertentu, dan hubungan antar tabel didefinisikan melalui kunci asing.

Contoh:

MySQL, PostgreSQL, Microsoft SQL Server, Oracle.

Kelebihan:

Struktur data yang terorganisir dengan baik, konsistensi data yang kuat (ACID), dan bahasa query yang terstandarisasi (SQL).

Kekurangan:

Skalabilitas vertikal (meningkatkan kapasitas server) lebih mudah daripada skalabilitas horizontal (menambahkan lebih banyak server), dan kompleksitas dalam menangani data yang tidak terstruktur.

2. Database NoSQL:

Karakteristik:

Tidak menggunakan skema tabel yang kaku seperti database relasional. Cocok untuk data yang tidak terstruktur atau semi-terstruktur.

Contoh:

MongoDB (berbasis dokumen), Redis (key-value), Cassandra (kolom).

Kelebihan:

Skalabilitas horizontal yang mudah, fleksibilitas dalam struktur data, dan performa yang baik untuk aplikasi yang membutuhkan kecepatan tinggi.

Kekurangan:

Konsistensi data yang umumnya akhirnya konsisten (BASE), dan bahasa query yang bervariasi tergantung jenis NoSQL.

3. Database Grafik:

Karakteristik:

Data disimpan dalam simpul (node) dan hubungan antar simpul (edge). Cocok untuk aplikasi yang memerlukan analisis hubungan data yang kompleks.

Contoh:

Neo4j, Datastax Enterprise Graph.

Kelebihan:

Analisis hubungan data yang efisien, kemampuan untuk menemukan pola tersembunyi, dan fleksibilitas dalam struktur data.

Kekurangan:

Kurang umum dibandingkan database relasional dan NoSQL, dan memerlukan pemahaman yang mendalam tentang teori graf.

4. Database Berorientasi Objek:

Karakteristik:

Data disimpan sebagai objek, yang merupakan gabungan dari data dan fungsi (metode) yang beroperasi pada data tersebut.

Contoh:

ObjectDB, db4o.

Kelebihan:

Pemetaan data yang lebih alami ke objek dalam bahasa pemrograman, dan kode yang lebih bersih dan mudah dipelihara.

Kekurangan:

Kurang umum digunakan, dan mungkin memerlukan lebih banyak sumber daya untuk pengelolaan dan optimasi.

5. Jenis Database Berdasarkan Fungsi:

Operational Database:

Digunakan untuk memproses transaksi dan data secara real-time, seperti dalam sistem e-commerce atau perbankan.

Data Warehouse:

Digunakan untuk menyimpan data dari berbagai sumber untuk keperluan analisis dan pelaporan.

Distributed Database:

Menyimpan data di berbagai lokasi fisik tetapi tetap terhubung sebagai satu kesatuan.

6. Jenis Database Berdasarkan Arsitektur:

Database Terpusat: Semua data disimpan di satu lokasi.

Database Terdistribusi: Data tersebar di beberapa lokasi.

End-User Database: Database yang digunakan oleh pengguna akhir untuk aplikasi tertentu.

Database Komersial: Database yang dikembangkan dan dijual oleh perusahaan komersial.

Pemilihan jenis database yang tepat sangat bergantung pada kebutuhan aplikasi dan jenis data yang akan disimpan.

**GraphQL**

GraphQL adalah bahasa kueri untuk API dan runtime untuk mengeksekusi kueri tersebut. Secara sederhana, GraphQL memungkinkan klien untuk meminta data yang spesifik dan dibutuhkan saja, berbeda dengan REST yang mungkin mengembalikan data lebih banyak daripada yang diperlukan. GraphQL juga dikenal karena fleksibilitas dan efisiensinya dalam pengambilan data, terutama dalam aplikasi yang kompleks.

Berikut adalah beberapa poin penting tentang GraphQL:

Fleksibel:

Klien dapat menentukan dengan tepat data apa yang mereka butuhkan dalam satu permintaan, menghindari pengambilan data yang berlebihan (over-fetching).

Efisien:

Karena hanya data yang diminta yang dikirimkan, GraphQL dapat mengurangi beban jaringan dan waktu pemrosesan.

Pencarian Data Deklaratif:

GraphQL menggunakan skema untuk mendefinisikan jenis data yang dapat dikueri dan hubungan antar jenis, memberikan struktur yang jelas untuk pengambilan data.

Resolvers:

Setiap bidang dalam skema memiliki resolver yang bertanggung jawab untuk mengambil data yang sesuai.

Mutasi:

GraphQL mendukung operasi mutasi untuk membuat, memperbarui, atau menghapus data di server.

Alternatif untuk REST:

GraphQL sering dianggap sebagai alternatif yang lebih efisien untuk REST API, terutama dalam aplikasi modern yang kompleks.

Singkatnya, GraphQL adalah alat yang kuat untuk membangun dan berinteraksi dengan API, menawarkan pendekatan yang lebih fleksibel dan efisien untuk pengambilan data dibandingkan dengan pendekatan tradisional seperti REST.

**Graph database**

jenis basis data yang dirancang untuk menyimpan dan mengelola data yang saling berhubungan dalam bentuk grafik. Grafik ini terdiri dari simpul (node) yang mewakili entitas dan hubungan (edge) yang mewakili hubungan antar entitas. Dengan kata lain, graph database unggul dalam memodelkan dan menganalisis data yang memiliki hubungan kompleks, seperti jaringan sosial, sistem rekomendasi, atau jaringan transportasi.

Perbedaan dengan Basis Data Relasional:

Struktur Data:

Basis data relasional menyimpan data dalam tabel dengan baris dan kolom, sedangkan graph database menggunakan grafik dengan node dan edge.

Hubungan Data:

Dalam basis data relasional, hubungan antar data diwakili oleh kunci asing, yang bisa menyebabkan performa lambat saat melakukan query yang melibatkan banyak hubungan. Di sisi lain, graph database menyimpan hubungan secara langsung dalam edge, sehingga query yang melibatkan hubungan bisa dilakukan lebih cepat.

Fleksibilitas:

Graph database lebih fleksibel dalam menangani perubahan skema karena Anda dapat dengan mudah menambahkan node dan edge baru tanpa harus mengubah struktur keseluruhan.

Kelebihan Graph Database:

Representasi yang Alami:

Memungkinkan pemodelan data yang lebih alami dan intuitif, mirip dengan cara manusia memahami hubungan.

Performa Tinggi:

Mampu melakukan query kompleks pada data yang terhubung dengan cepat, terutama pada aplikasi yang membutuhkan navigasi grafik.

Fleksibilitas Skema:

Mudah untuk dimodifikasi dan diperluas dengan menambahkan node dan edge baru.

Analisis Jejaring:

Sangat cocok untuk menganalisis hubungan dalam jejaring sosial, menemukan pola dan wawasan tersembunyi.

Contoh Penggunaan:

Jejaring Sosial:

Menganalisis koneksi antar pengguna, menemukan teman bersama, memahami pengaruh sosial.

Sistem Rekomendasi:

Memberikan rekomendasi produk atau konten yang dipersonalisasi berdasarkan preferensi pengguna dan hubungan antar produk.

Analisis Keuangan:

Mendeteksi penipuan dengan menganalisis hubungan antara transaksi, akun, dan pihak terkait.

Sistem Transportasi:

Merencanakan rute optimal, menganalisis pola lalu lintas, dan mengoptimalkan manajemen jaringan transportasi.

Contoh Graph Database Populer:

Neo4j, Amazon Neptune, ArangoDB.

Pernahkah Anda mendengar istilah API (Application Programming Interface)? Bukan hanya banyak digunakan pada transaksi bisnis, nyatanya API adalah essential thing yang dapat mempermudah para developers untuk membuat dan mengembangkan aplikasi.

Bahkan, API juga kerap kali digunakan di berbagai media sosial. Untuk Anda yang masih bingung dengan apa itu API dan bagaimana cara kerjanya, mari kita cari tahu bersama-sama lewat artikel di bawah ini!

Apa Itu API?

Memahami apa itu API ternyata tidak sulit, kok. Bayangkan saja API adalah restoran mewah yang menyediakan berbagai macam menu makanan. Anda tinggal memilih, mana santapan yang ingin dipesan.

Setelah itu, mereka akan membuatkannya untuk Anda tanpa perlu Anda tahu bagaimana tahapan pembuatannya. Selain itu, API juga bisa bekerja layaknya sebuah restoran yang mampu menghidangkan makanan dengan bahan-bahan dari Anda.

Membaca pengertian mudah tentang API tersebut, tentu kini Anda sudah paham bahwa API adalah platform yang memudahkan developer untuk membuat atau mengembangkan aplikasi, bukan? Agar lebih memahami cara kerja platform yang satu ini, Anda harus tahu bahwa seringkali kita membutuhkan dua aplikasi untuk saling terhubung, API bisa mempermudah hal tersebut untuk terwujud.

Cara Kerja API

Kini saatnya Anda memahami cara kerja API supaya makin mengerti bagaimana pentingnya tools yang satu ini di era digital. Proses API menerima perintah dan memberikan respons itu terbagi menjadi empat langkah.

Pertama, Anda menyampaikan request ke aplikasi yang telah menggunakan perangkat API. Kemudian, perintah tersebut akan diteruskan ke server yang berkaitan.

Setelah itu, API akan mendapatkan informasi atau data-data yang diperlukan pada server tersebut. Terakhir, informasi akan diteruskan oleh API ke aplikasi yang Anda gunakan tadi.

Sebagai contoh, mari kita bayangkan saat membuka aplikasi Traveloka. Anda mungkin akan membuka aplikasi ini ketika ingin memesan tiket kereta api.

Ketika Anda berusaha memperoleh jadwal keberangkatan kereta api pada hari dan jam tertentu, Traveloka akan mengakses alamat API. Kemudian, API ini nantinya akan menghubungi server Kereta Api Indonesia.

Data yang sudah diperoleh akan diteruskan ke API. Lalu, API akan menyalurkannya ke Traveloka sehingga detail keberangkatan berikut nama kereta dan jamnya akan tampil di aplikasi.

Baca juga: Website Ecommerce Indonesia dengan Halaman Checkout Terbaik

4 Jenis API yang Paling Umum

Pada dasarnya, ada empat jenis API yang dibedakan berdasarkan hak aksesnya. Apa sajakah itu? Anda bisa cari tahu selengkapnya pada poin-poin di bawah ini!

1. Public API

Jenis API yang pertama adalah Public API. Sesuai namanya, jenis API yang satu ini bisa diakses oleh siapapun dengan mudah untuk berbagai lintas platform.

Namun, kelemahannya tentu berada pada proses autentikasinya yang rendah. Sehingga, data yang bisa dibagikan mungkin juga terbatas pada hal-hal tertentu.

Public API dapat Anda manfaatkan hanya dengan mendaftar atau langsung menggunakannya di aplikasi. Salah satu contoh Open API yang mudah Anda temui adalah Google Maps atau data Covid di Indonesia.

2. Private API

Berbeda dengan poin sebelumnya, Private API adalah tools yang dibuat untuk keperluan internal sehingga tidak dapat diakses oleh sembarang orang.

Kelebihan dari jenis API yang satu ini tentunya berada pada tingkat keamanan yang lebih terjamin. Sementara itu, tujuan pembuatan Private API biasanya digunakan untuk mengembangkan suatu jenis aplikasi.

Contohnya, API dari bagian back end akan mengakses data ke bagian front end sehingga dapat makin mengembangkan hal-hal yang sudah dimiliki oleh perusahaan tersebut.

3. Composite API

Untuk Anda yang ingin menghemat waktu dan tenaga, Composite API adalah pilihan tepat. Jenis API ini memungkinkan penyimpanan data dan informasi dari berbagai server dalam satu tempat saja. Sehingga, saat Anda membutuhkan data-data tersebut, proses pengambilannya akan lebih cepat dan tidak memakan banyak effort. Sebab, kompleksitas yang umum terjadi dalam sistem dapat diminimalkan.

4. Partner API

Sedikit berbeda dengan Public API atau Open API, Partner API bisa digunakan oleh umum yang memiliki izin atau lisensi khusus. Biasanya, jenis API yang seperti ini hanya dipakai oleh orang-orang yang berhubungan langsung dengan bisnis tersebut. Dengan sistem yang lebih tertutup, tentunya keamanan data di Partner API lebih terjamin dibandingkan dengan Open API. Selain itu, pemilik API yang menggunakan sistem ini pun dapat lebih mudah mengontrol penggunaan resources mereka yang dibuka untuk orang-orang tertentu.

6 Jenis Arsitektur API

Proses penyimpanan dan pengiriman data tentu berbeda pada masing-masing API. Hal ini bergantung dari jenis arsitektur yang digunakan. Nah, supaya Anda bisa tahu lebih detail, simak enam jenis arsitektur API yang tersedia di pasaran!

1. REST

REST merupakan salah satu arsitektur API yang paling populer saat ini. API dengan arsitektur REST menggunakan metode HTTP, misalnya GET, POST, PUT, dan DELETE untuk berkomunikasi dengan server. Data dikirimkan dalam format yang umum, seperti JSON atau XML. Arsitektur REST didasarkan pada prinsip-prinsip seperti statelessness, antarmuka yang seragam (uniform interface), dan pemanfaatan cache.

Kelebihan REST terletak pada kesederhanaan implementasi dan skalabilitasnya yang baik. Di sisi lain, kekurangan REST meliputi kurangnya standar yang konsisten dalam pengembangan API, kurangnya fleksibilitas dalam mengatur permintaan khusus, dan masalah skalabilitas saat aplikasi berkembang besar.

2. SOAP

SOAP (Simple Object Access Protocol) adalah protokol yang lebih kompleks daripada poin sebelumnya, yakni REST. SOAP menggunakan format XML untuk pertukaran data dan mendefinisikan struktur pesan yang konsisten. Dalam SOAP, operasi API dijelaskan dalam Web Services Description Language (WSDL).

Kelebihan SOAP adalah dukungan yang kuat terhadap keamanan, pengiriman pesan yang andal melalui perbedaan protokol, dan dukungan terhadap transaksi yang kompleks. Namun, implementasi SOAP cenderung lebih kompleks dan memiliki overhead yang lebih tinggi dibandingkan dengan REST.

3. GraphQL

GraphQL adalah bahasa query yang memungkinkan pengguna untuk mengambil data dengan presisi yang tinggi. Dengan GraphQL, pengguna dapat meminta hanya data yang diperlukan dan menghindari over-fetching atau under-fetching. Selain itu, GraphQL juga memungkinkan pengguna untuk melakukan multiple request dalam satu kali permintaan.

Kelebihan GraphQL adalah fleksibilitasnya untuk mengatur dan mengambil data, sehingga memungkinkan pengembang merancang antarmuka API yang lebih efisien dan adaptif. Kekurangan GraphQL antara lain pembelajaran kurva yang curam bagi pengembang baru, masih ada kecenderungan over-fetching atau under-fetching jika tidak dirancang dengan baik, dan keterbatasan dalam menghadapi query yang kompleks.

4. Apache Kafka

Apache Kafka adalah platform streaming yang digunakan untuk membangun sistem yang dapat mengirim dan menerima data secara real-time. Arsitektur API yang satu ini memungkinkan komunikasi yang tidak sinkron antar-aplikasi. Kafka sering digunakan dalam sistem yang membutuhkan streaming data, seperti pemrosesan log, sistem notifikasi, dan analisis data.

Kelebihan Apache Kafka adalah skalabilitas dan daya tahannya yang tinggi dan memiliki dukungan tinggi untuk pengolahan data secara real-time. Namun, arsitektur API satu ini memerlukan proses konfigurasi di awal yang cukup rumit. Jadi, Anda perlu tenaga khusus untuk mengoperasikannya agar berjalan dengan optimal.

5. Async API

Async API (Asynchronous API) adalah arsitektur API yang dirancang untuk komunikasi asinkron (tidak sinkron). Dalam Async API, pesan dikirimkan secara tidak langsung melalui antrian pesan atau sistem penengah lainnya. Arsitektur ini cocok untuk sistem yang membutuhkan pertukaran pesan tanpa memerlukan respons langsung.

Kelebihan Async API adalah kemampuannya untuk mengatasi beban tinggi dan meningkatkan responsivitas aplikasi, serta memungkinkan integrasi yang lebih longgar antar-aplikasi. Kekurangan Async API meliputi kompleksitas konfigurasi, koordinasi transaksi yang rumit, pengelolaan kesalahan yang kompleks, dan pemantauan serta debugging yang lebih sulit.

6. RPC

RPC adalah arsitektur API yang memungkinkan aplikasi untuk memanggil fungsi atau prosedur yang berjalan pada server jarak jauh. RPC menggunakan protokol dan mekanisme komunikasi tertentu, seperti XML-RPC atau JSON-RPC.

Kelebihan RPC termasuk performa yang cepat, implementasi yang sederhana, dan dukungan untuk berbagai bahasa pemrograman. Meski begitu, bukan berarti arsitektur API ini tidak memiliki kelemahan.

Kekurangan RPC antara lain ketergantungan yang tinggi terhadap protokol yang dipakai serta kompleksitas dari segi manajemen. Sehingga, arsitektur API RPC ini akan lebih tepat jika digunakan oleh pelaku bisnis skala menengah cenderung besar.

Baca juga: 7 Keuntungan Memiliki Sistem Pembayaran di Toko Online

Contoh API yang Populer

API digunakan dalam berbagai aplikasi yang sering kita gunakan sehari-hari. Berikut adalah beberapa contoh penggunaan API yang populer.

Google Search

API Google Search memungkinkan pengembang untuk mengintegrasikan hasil pencarian Google ke dalam aplikasi mereka. Sehingga, hasil pencarian yang muncul di aplikasi pengguna akan didasarkan pada hal-hal yang dicari di mesin pencarian Google.

Contohnya, dengan menggunakan API Google Search, pengembang aplikasi e-commerce dapat membuat fitur pencarian untuk memudahkan pengguna mencari produk. Hasil pencarian produk dan rekomendasi yang ditampilkan pun biasanya berdasarkan hal-hal yang dicari oleh pengguna di platform tersebut atau lainnya yang masih bersinggungan.

Weather Widget

Sekarang ini sudah banyak situs web hingga aplikasi yang menyediakan prakiraan cuaca. Untuk mewujudkan hal itu, tentunya situs web dan aplikasi tersebut menggunakan API cuaca sehingga informasi cuaca terkini bisa disuguhkan kepada pengguna mereka.

Dengan menggunakan API cuaca, mereka dapat mengambil data cuaca dari sumber terpercaya dan menampilkannya dalam tampilan yang mudah dibaca. Contohnya, aplikasi cuaca pada ponsel pintar biasanya mengambil data cuaca dari API cuaca dan menampilkan informasi seperti suhu, kelembapan, hingga prakiraan cuaca.

Log in dengan akun media sosial

Banyak aplikasi atau situs web yang memungkinkan pengguna untuk masuk menggunakan akun media sosial, seperti Facebook atau Google. Anda pastinya sudah tidak asing dengan peristiwa ini, bukan? Misalnya, Anda lebih memilih log in ke Twitter menggunakan akun Google daripada membuat akun baru.

Penggunaan API pada media sosial ditujukan untuk mengautentikasi pengguna dan mengambil informasi dasar, seperti nama dan gambar profil. Dengan begitu, aplikasi dapat mempersingkat proses pendaftaran atau sign in, sehingga pengguna dapat segera berbagi konten dengan mudah dan praktis.

Bot Twitter

Bot Twitter menggunakan API Twitter untuk mengirim dan menerima tweet, mengikuti pengguna, atau melakukan tindakan lainnya di platform Twitter secara otomatis. API Twitter memungkinkan pengembang untuk mengakses fitur-fitur Twitter dari aplikasi mereka sendiri.

Contoh penggunaan API di aplikasi Twitter adalah sebuah bot Twitter yang dapat diprogram untuk merespons otomatis terhadap mention. Anda bisa mencoba hal ini dengan membalas tweet orang dengan menyebut akun bot Twitter. Nantinya, mereka akan memberikan jawaban otomatis, berkaitan dengan tweet yang Anda balas. Selain itu, bot Twitter juga dapat dipakai untuk mencatat data tertentu berdasarkan kata kunci yang ditentukan.

Pemesanan tiket dan hotel

Seperti yang sudah dijelaskan sebelumnya, layanan OTA (Online Travel Agent) seperti Traveloka atau Agoda tentunya menggunakan API dari maskapai penerbangan, perusahaan kereta api, dan hotel supaya pengguna aplikasi dapat melakukan pemesanan langsung tanpa perlu menginstal aplikasi dari berbagai akomodasi tersebut.

Dengan menggunakan API, pengguna dapat melihat ketersediaan tiket atau kamar hotel secara real-time, membandingkan harga, dan melakukan pemesanan dengan mudah. Kata lainnya adalah API ini menghubungkan layanan OTA dengan penyedia jasa transportasi atau akomodasi dan menyampaikannya langsung ke pengguna.

Perbedaan API dan Web Service

Meskipun sering digunakan secara bergantian, terdapat perbedaan antara API dan web service. Berikut adalah beberapa perbedaan utama yang dapat Anda temukan.

Protokol yang didukung

Salah satu perbedaan antara API dan web service adalah protokol yang didukung. API dapat menggunakan berbagai protokol komunikasi, seperti HTTP, REST, SOAP, GraphQL, dan sebagainya. Protokol yang digunakan bergantung pada kebutuhan dan tujuan pengembangan aplikasi.

Di sisi lain, web service umumnya menggunakan protokol SOAP (Simple Object Access Protocol) untuk pertukaran pesan. Protokol SOAP didesain untuk memungkinkan komunikasi antara aplikasi yang menggunakan teknologi yang berbeda dan mendefinisikan format pesan dalam XML.

Metode komunikasi

API cenderung memiliki antarmuka yang lebih sederhana dan terfokus pada pengiriman data yang ringkas. API memungkinkan aplikasi untuk berkomunikasi dan bertukar data dengan sistem lain melalui permintaan dan respons. Metode komunikasi yang umum digunakan dalam API adalah metode HTTP, seperti GET, POST, PUT, dan DELETE. Dalam API, data dikirimkan dalam format yang umum, seperti JSON atau XML.

Di sisi lain, web service dapat menangani pertukaran pesan yang lebih kompleks dan menyediakan fungsionalitas yang lebih luas. Web service dapat memfasilitasi komunikasi antara aplikasi yang menggunakan berbagai teknologi dan bahasa pemrograman. Web service juga seringkali menggunakan protokol SOAP dan mendefinisikan operasi melalui WSDL (Web Services Description Language).

Respons yang diberikan

Salah satu perbedaan lainnya antara API dan web service adalah respons yang diberikan. API umumnya memberikan respons dalam format yang lebih sederhana, misalnya JSON atau XML. Data dikirimkan dengan struktur yang lebih terbaca oleh manusia dan mudah diintegrasikan ke dalam aplikasi lain. Respons API dapat berupa data yang diminta oleh aplikasi pengguna atau pesan yang mengindikasikan keberhasilan atau kegagalan operasi.

Di sisi lain, web service, terutama yang menggunakan protokol SOAP, seringkali menyediakan respons yang lebih terstruktur dengan pesan dalam format XML. Respons web service umumnya mencakup elemen metadata dan struktur yang didefinisikan secara khusus sesuai dengan standar yang diikuti.

Di dunia yang semakin terhubung seperti sekarang ini, API adalah elemen penting untuk mengintegrasikan berbagai aplikasi dan sistem. Dengan memahami pengertian, jenis-jenis, cara kerja, dan contoh penggunaan API, Anda dapat memanfaatkannya untuk mengembangkan aplikasi yang lebih kuat dan terhubung.

**Framework vs Library**

Perbedaan utama antara framework dan library terletak pada tingkat kontrol dan fleksibilitas yang diberikan kepada pengembang. Framework menyediakan kerangka kerja yang sudah jadi dengan aturan dan struktur yang harus diikuti, sementara library adalah kumpulan fungsi yang dapat dipanggil sesuai kebutuhan, memberikan lebih banyak kontrol kepada pengembang.

Framework

Definisi:

Framework adalah kerangka kerja yang menyediakan struktur dan aturan untuk membangun aplikasi. Ini mengatur alur kerja dan bagaimana kode harus ditulis.

Kontrol:

Framework memiliki inversi kontrol, artinya framework yang mengontrol alur aplikasi, bukan pengembang.

Contoh:

Framework PHP seperti Laravel, framework JavaScript seperti Angular dan React, framework CSS seperti Bootstrap.

Penggunaan:

Cocok untuk proyek besar yang membutuhkan struktur yang kuat dan konsisten.

Kelebihan:

Mempercepat pengembangan, memastikan konsistensi, dan mempermudah pemeliharaan.

Kekurangan:

Kurang fleksibel, bisa membatasi kreativitas pengembang.

Library

Definisi:

Library adalah kumpulan fungsi dan kode siap pakai yang dapat dipanggil oleh pengembang untuk melakukan tugas-tugas tertentu.

Kontrol:

Pengembang memiliki kendali penuh atas bagaimana dan kapan memanggil fungsi dalam library.

Contoh:

Library JavaScript seperti jQuery, React, dan Three.js, library matematika, library untuk manipulasi string.

Penggunaan:

Cocok untuk tugas-tugas spesifik dan proyek yang membutuhkan fleksibilitas.

Kelebihan:

Fleksibel, mudah dipelajari, dan modular.

Kekurangan:

Membutuhkan lebih banyak usaha untuk membangun struktur aplikasi dan membutuhkan pemahaman lebih dalam tentang cara kerja library.

|  |  |  |
| --- | --- | --- |
| **Fitur** | **Library** | **Framework** |
| Kontrol | Pengembang | Framework |
| Struktur | Bebas | Disediakan |
| Fleksibilitas | Tinggi | Terbatas |
| Penggunaan | Tugas spesifik | Proyek besar |
| Inversi Kontrol | Tidak ada | Ada |

Kesikesimpulan:

Pemilihan antara framework dan library bergantung pada kebutuhan proyek. Jika membutuhkan struktur yang kuat dan konsisten, framework adalah pilihan yang baik. Jika membutuhkan fleksibilitas dan kontrol penuh, library adalah pilihan yang lebih tepat. Kadang-kadang, library dapat digunakan bersama dengan framework untuk melengkapi fungsionalitasnya.