|  |  |
| --- | --- |
| **Tanggal Asistensi :** | |
| **Tanda Tangan** | **Nilai** |
|  |  |

MODUL 3

*HTML LAYOUT*

Nama: Angelina Febriani Madesen

NIM: F55123084

Kelas : C

1. HTML *LAYOUT*

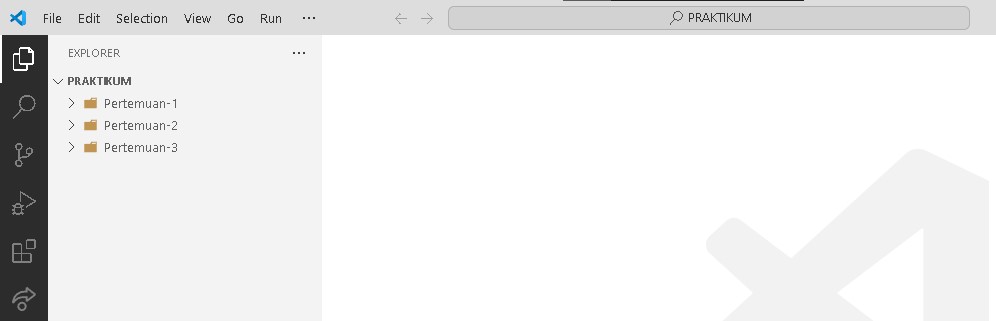
*Layout* pada *HTML* adalah tata letak atau susunan kerangka sebuah halaman *web*. Ini adalah bagian penting dalam desain *web* yang mengatur cara elemen-elemen konten seperti teks, gambar, dan elemen lainnya ditempatkan dan ditampilkan di halaman *web*. *Layout* bertujuan untuk menciptakan tampilan yang terstruktur, baik, dan responsif agar pengguna dapat dengan mudah berinteraksi dengan konten.

Dalam *HTML*, *layout* dapat diatur menggunakan berbagai elemen dan teknik. Beberapa elemen yang umum digunakan dalam pembuatan *layout HTML* termasuk *tag* ‘<div>’ (divisi), *tag* ‘<table>’ (tabel), *tag* ‘<header>’ (bagian atas), *tag* ‘<footer>’ (bagian bawah), dan lainnya. Selain itu, *CSS* (*Cascading Style Sheets*) juga digunakan untuk mengontrol tampilan dan posisi elemen-elemen tersebut.

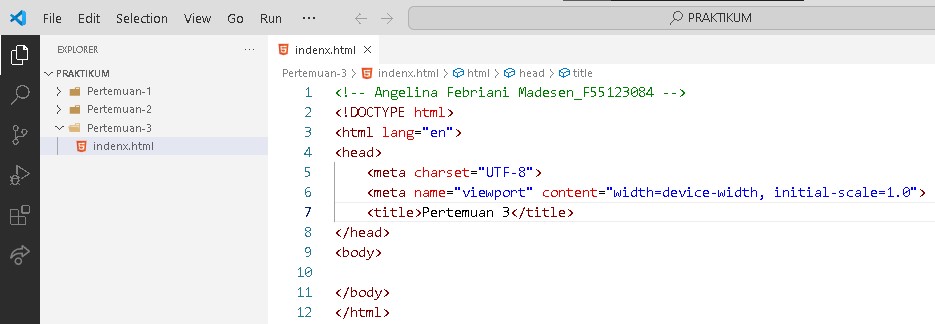
Mengenal teknik dan elemen-elemen tata letak *HTML* sangat penting dalam mengembangkan halaman *web* yang menarik dan berfungsi dengan baik. Dengan memahami prinsip-prinsip dasar *layout* *HTML*, anda dapat merancang tampilan *web* yang sesuai dengan kebutuhan dan preferensi desain anda serta memberikan pengalaman yang baik kepada pengguna.

Pembuatan *layout HTML* juga melibatkan penggunaan teknik CSS yang lebih kompleks, seperti *Flexbox* dan *Grid*, yang memungkinkan penataan elemen secara responsif dan dinamis. *Flexbox* sangat berguna untuk mengatur elemen-elemen dalam satu baris atau kolom, sedangkan *Grid* memungkinkan pengaturan dalam baris dan kolom sekaligus, sehingga lebih fleksibel untuk struktur yang kompleks. Selain itu, *layout* *HTML* responsif dibuat dengan memanfaatkan media *queries* di *CSS*, yang menyesuaikan tampilan elemen berdasarkan ukuran layar perangkat. *Framework* *CSS* seperti *Bootstrap* dan *Tailwind* *CSS* juga menyediakan kelas-kelas siap pakai yang memudahkan dalam pembuatan *layout* yang konsisten di berbagai perangkat. *Layout* yang terstruktur tidak hanya meningkatkan estetika, tetapi juga mempermudah navigasi, meningkatkan aksesibilitas, dan mengoptimalkan pengalaman pengguna. Pemahaman mendalam tentang elemen dan teknik *layout* membantu pengembang *web* menciptakan halaman yang intuitif, profesional, dan mudah diakses.

1. *HTML LAYOUT*
2. *Project*
3. Pada folder utama, buat folder baru dengan nama “Pertemuan-3”.



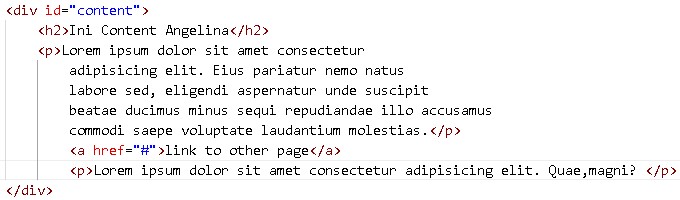
1. Pada folder “Pertemuan-3”, buat *file* dengan nama “index.html” dan tambahkan struktur dasar *HTML* 5 kedalamnya.



1. *Layout*
2. Dalam *tag body*, tambahkan kode berikut untuk membuat *layout* navigasi.



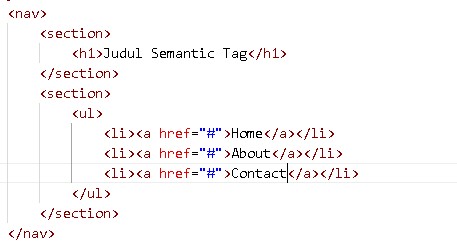
1. Tambahkan kode berikut untuk membuat *layout content.*



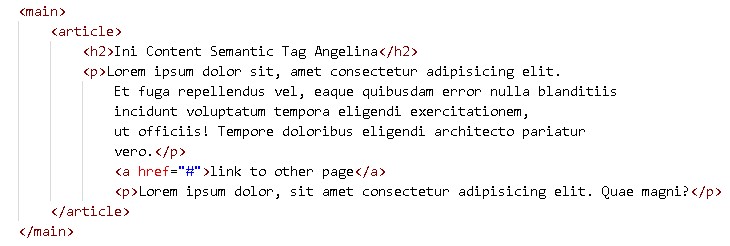
1. Tambahkan kode berikut untuk membuat *layout footer*.



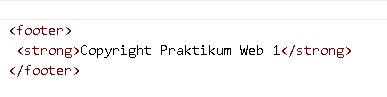
1. *Semantic Layout*
2. Dalam *tag body*, tambahkan kode berikut untuk membuat *layout navbar.*



1. Tambahkan kode berikut untuk membuat *layout content.*



1. Tambahkan kode berikut untuk membuat *layout footer*.



1. *HTML LAYOUT*
2. *Layout*



Gambar di atas merupakan hasil dari implementasi program *Layout html*, yang dimana menghasilkan halaman *web* sederhana yang terdiri dari beberapa bagian utama: ‘navbar’, ‘content’, dan ‘footer’. Bagian ‘navbar’ berisi judul utama ‘Judul’ serta daftar navigasi dengan tiga tautan, yaitu ‘Home’, ‘About’, dan ‘Contact’, yang masing-masing mengarah ke ‘#’ sebagai contoh tautan *placeholder* tanpa halaman tujuan spesifik. Selanjutnya, bagian ‘content’ menyajikan subjudul ‘Ini Content Angelina Febriani Madesen\_F55123084’ dan diikuti oleh dua paragraf teks ‘Lorem ipsum’, yang berfungsi sebagai *placeholder* untuk konten teks yang lebih panjang. Selain paragraf, terdapat sebuah tautan tambahan yang bertuliskan ‘link to other page’ yang, seperti tautan di ‘navbar’, juga mengarah ke ‘#’ dan berfungsi sebagai contoh tanpa tujuan tertentu. Bagian ‘footer’ pada halaman ini memuat teks ‘Copyright Praktikum Web 1’ yang ditebalkan menggunakan *tag* ‘<strong>’, memberikan penekanan pada teks tersebut agar lebih menonjol. Kode ini tidak menyertakan elemen gaya atau skrip tambahan, sehingga tampilan halaman sangat mendasar dengan struktur *HTML* dasar saja, menghasilkan *layout* statis tanpa desain visual yang mendetail. Halaman ini memberikan kerangka sederhana yang dapat dikembangkan lebih lanjut dengan menambahkan elemen *CSS* untuk memperindah tampilan.

1. *Semantic Layout*



Gambar di atas merupakan hasil dari implementasi program *Semantic Layout html*, yang dimana menampilkan sebuah halaman *web* sederhana yang menggunakan elemen-elemen semantik *HTML5* seperti ‘<nav>’, ‘<main>’, ‘<section>’, ‘<article>’, dan ‘<footer>’ untuk memperjelas struktur konten. Di dalam elemen ‘<nav>’, terdapat dua ‘<section>’ yang berfungsi untuk menampilkan judul ‘Judul Semantic Tag’ serta daftar navigasi dengan tiga tautan: ‘Home’, ‘About’, dan ‘Contact’, yang semuanya mengarah ke ‘#’ sebagai *placeholder* tanpa tujuan tertentu. Bagian konten utama dalam elemen ‘<main>’ berisi ‘<article>’ yang menampilkan subjudul ‘Ini Content Semantic Tag Angelina’ diikuti dengan dua paragraf teks *Lorem ipsum* sebagai teks contoh. Ada juga tautan tambahan bertuliskan ‘link to other page’, yang seperti tautan sebelumnya, hanya mengarah ke ‘#’ tanpa halaman spesifik. Bagian bawah halaman menggunakan ‘<footer>’ untuk menampilkan teks ‘Copyright Praktikum Web 1’ yang diberi penekanan tebal dengan *tag* ‘<strong>’. Dengan menggunakan elemen semantik, struktur halaman menjadi lebih mudah dibaca dan diidentifikasi, baik oleh pembaca manusia maupun mesin pencari, meskipun tampilannya tetap sederhana tanpa elemen *CSS* atau *JavaScript* tambahan untuk *styling* atau fungsionalitas interaktif. Desain ini berfokus pada penggunaan markup yang bermakna dan memberikan dasar yang lebih baik untuk pengembangan lebih lanjut di aspek visual atau fungsionalnya.

1. *HTML LAYOUT*

*Layout HTML* adalah pengaturan elemen-elemen dalam halaman *web* untuk menciptakan tampilan yang terstruktur, responsif, dan mudah dinavigasi. Teknik *CSS* seperti *Flexbox*, *Grid*, dan media *queries* memungkinkan penataan elemen secara dinamis, sementara *framework* seperti *Bootstrap* dan *Tailwind* *CSS* mempermudah pembuatan *layout* yang konsisten. Memahami elemen dan teknik *layout* penting untuk merancang halaman *web* yang menarik, fungsional, dan memberikan pengalaman pengguna yang optimal.