Angular setup:

How to install angular cli using terminal:

```
root1@LAPTOP-R268MI6J:~$ sudo npm install -g @angular/cli
[sudo] password for root1:

added 273 packages in 20s

52 packages are looking for funding
  run `npm fund` for details
root1@LAPTOP-R268MI6J:~$ ng version


Angular CLI: 19.2.1
Node: 18.19.1
Package Manager: npm 9.2.0
OS: linux x64

Angular:
...

Package                      Version
---------------------------------------------------------
@angular-devkit/architect    0.1902.1 (cli-only)
@angular-devkit/core         19.2.1 (cli-only)
@angular-devkit/schematics   19.2.1 (cli-only)
@schematics/angular          19.2.1 (cli-only)
```

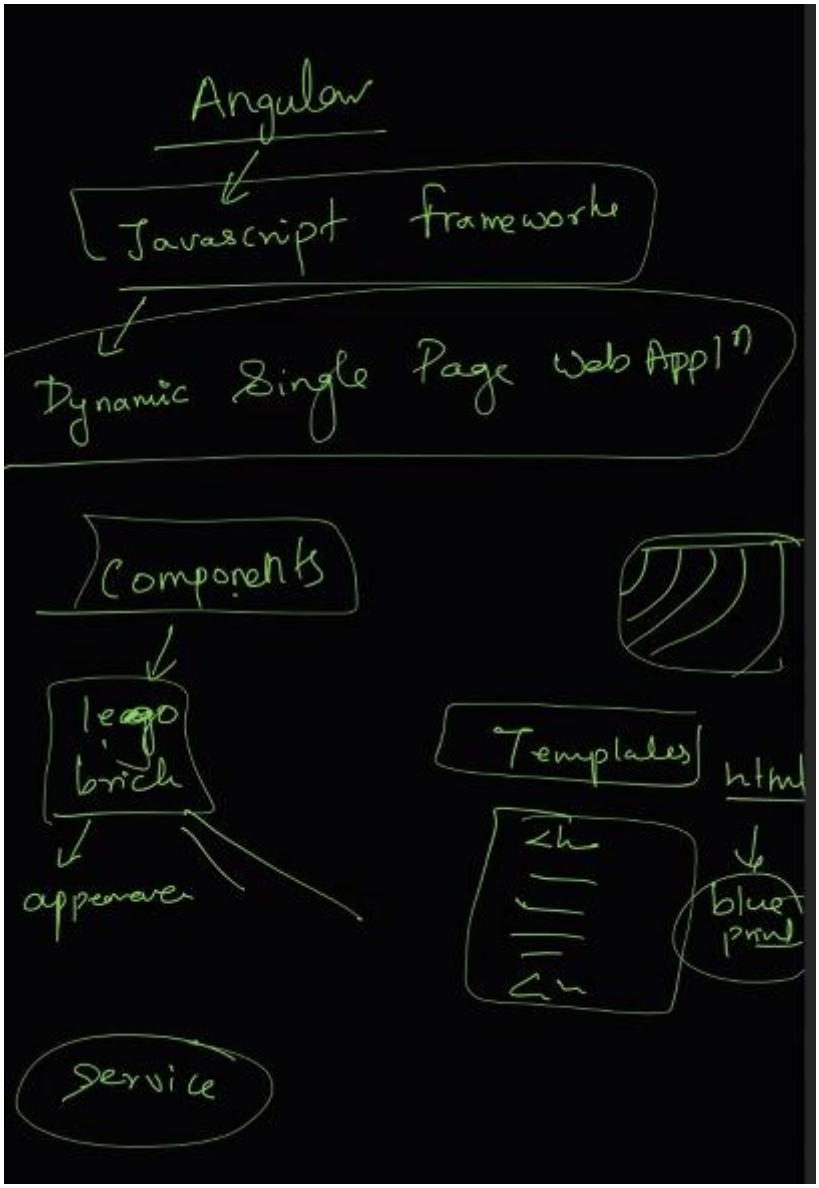
Then we create an folder named hello world:

```
root1@LAPTOP-R268MI6J:~$ ng new first-hello-world
 Would you like to enable autocompletion? This will set up your terminal so pressing TAB while typing Angular CLI commands will show possible options and
autocomplete arguments. (Enabling autocompletion will modify configuration files in your home directory.) Yes
Appended `source <(ng completion script)` to `/home/root1/.bashrc`. Restart your terminal or run the following to autocomplete `ng` commands:

    source <(ng completion script)

Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.dev/cli/analytics.

   Yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

    ng analytics disable --global

Global setting: enabled
Local setting: No local workspace configuration file.
Effective status: enabled
✔Which stylesheet format would you like to use? CSS            [ https://developer.mozilla.org/docs/Web/CSS                    ]
✔Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? Yes
✔Would you like to use the Server Routing and App Engine APIs (Developer Preview) for this server application? Yes
CREATE first-hello-world/README.md (1478 bytes)
CREATE first-hello-world/.editorconfig (314 bytes)
CREATE first-hello-world/.gitignore (587 bytes)
CREATE first-hello-world/angular.json (2789 bytes)
CREATE first-hello-world/package.json (1258 bytes)
CREATE first-hello-world/tsconfig.json (915 bytes)
CREATE first-hello-world/tsconfig.app.json (489 bytes)
CREATE first-hello-world/tsconfig.spec.json (434 bytes)
CREATE first-hello-world/.vscode/extensions.json (130 bytes)
CREATE first-hello-world/.vscode/launch.json (470 bytes)
CREATE first-hello-world/.vscode/tasks.json (938 bytes)
CREATE first-hello-world/src/main.ts (250 bytes)
CREATE first-hello-world/src/index.html (301 bytes)
```

And when done with it then:

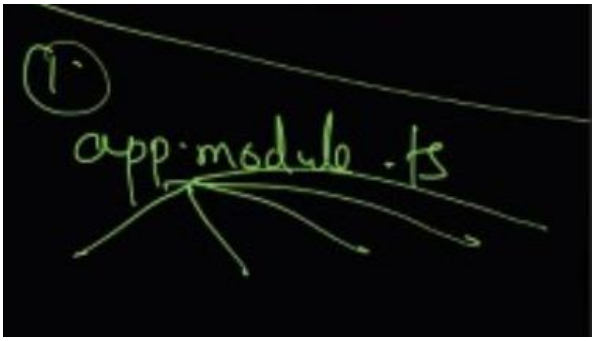
```
root1@LAPTOP-R268MI6J:~$ cd first-hello-world/
root1@LAPTOP-R268MI6J:~/first-hello-world$ cursor .
root1@LAPTOP-R268MI6J:~/first-hello-world$
```

Components of Angular:



- html define how component would look .
- then template refer to blueprint .
- now all js file data fetching and all operational code is stored in service file .

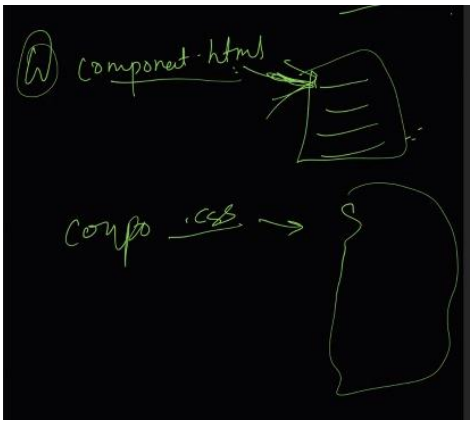
The main file in angular is "app.module.typescript":



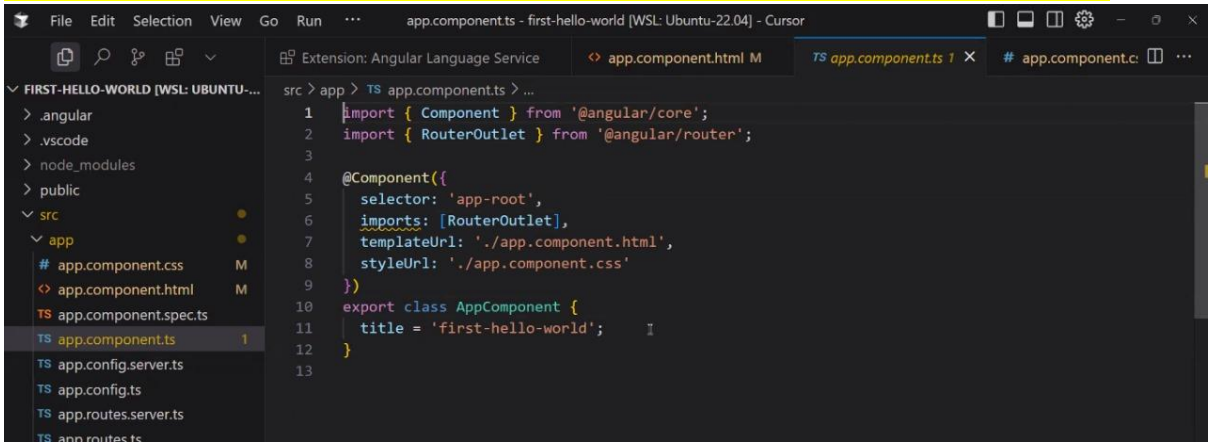
Now second file main container with which all other child containers are connected:

And all the behaviours of components and methods are defined in type script:

if we don't do anything in app.component.ts then by default we use the component.html and component.css which acts as default page:
(otherwise it would always prefer app.component)



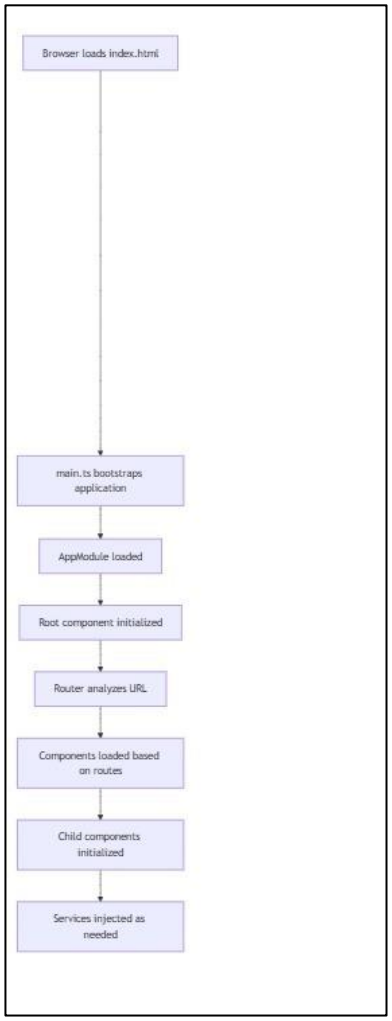
app.component.ts main root file or the main controller of app:



and here we defined app root with the component.html and component.css.

And in main html file we simply write <app-root> to invoke and use html and css of it

THE STUCTURE OF FLOW OF FILES:



Now in app.components.html:

```
Extension: Angular Language Service   app.component.html 4, M   app.component.ts 1   app.compo
src > app > app.component.html > div.container > div.color-change > div.color-controls
Go to component
<div class="container">
  <h1> Second Angular App</h1>
  <p> This is a second Angular App</p>
  <div class="color-change">
    <h2>Color Change</h2>
    <div class="color-box" [style.backgroundColor]="changeColor"></div>
    <div class="color-controls">
    <button (click)="changeColor('red')">Red</button>
    <button (click)="changeColor('blue')">Blue</button>
    <button (click)="changeColor('green')">Green</button>
    </div>
  </div>
</div>
```

And to implement on app.component.ts:

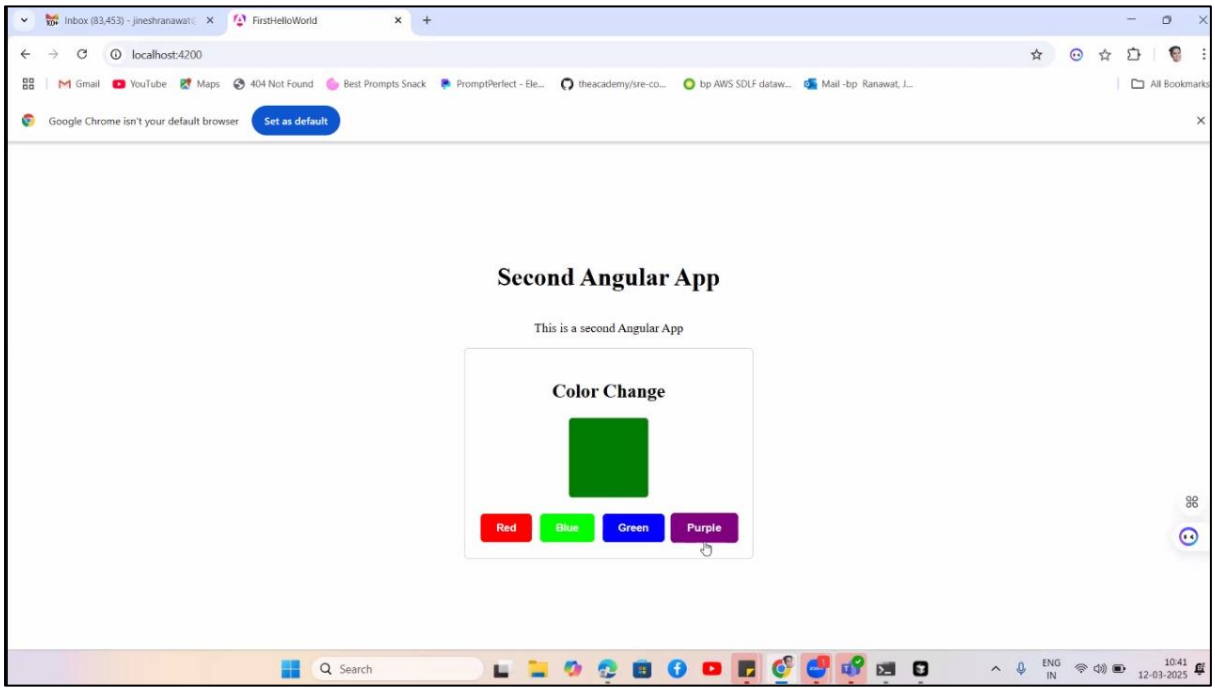
```
Extension: Angular Language Service   app.component.html M   app.component.ts 1
src > app > app.component.ts > AppComponent > changeColor
    imports: [RouterOutlet],
    templateUrl: './app.component.html',
    styleUrl: './app.component.css'
  })
  export class AppComponent {
    title = 'second-app';
    currentColor = '#cccccc';

    changeColor(color: string) {
      this.currentColor = color;
    }
  }

```

And then in app.component.css:

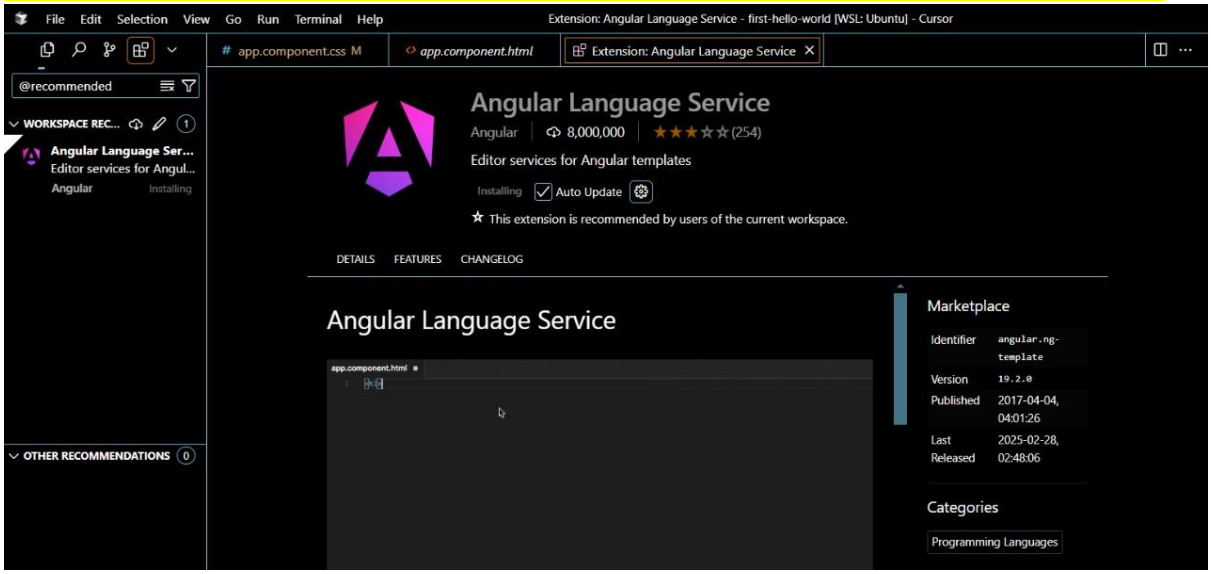
```
ngular Language Service   app.component.html M   app.component.ts 1, M   app.component.css M
src > app > app.component.css > .color-box
.color-change {
}
.color-box {
  width: 100px;
  height: 100px;
  margin: 20px auto;
  border-radius: 5px;
}
.color-controls {
  margin-top: 10px;
  display: flex;
  justify-content: center;
  gap: 10px;
}
button {
```

```
button:nth-child(1) {
  background-color: #ff0000;
  color: white;
}
button:nth-child(2) {
  background-color: #00ff00;
  color: white;
}
button:nth-child(3) {
  background-color: #0000ff;
  color: white;
}
button:nth-child(4) {
  background-color: #800080;
  color: white;
```

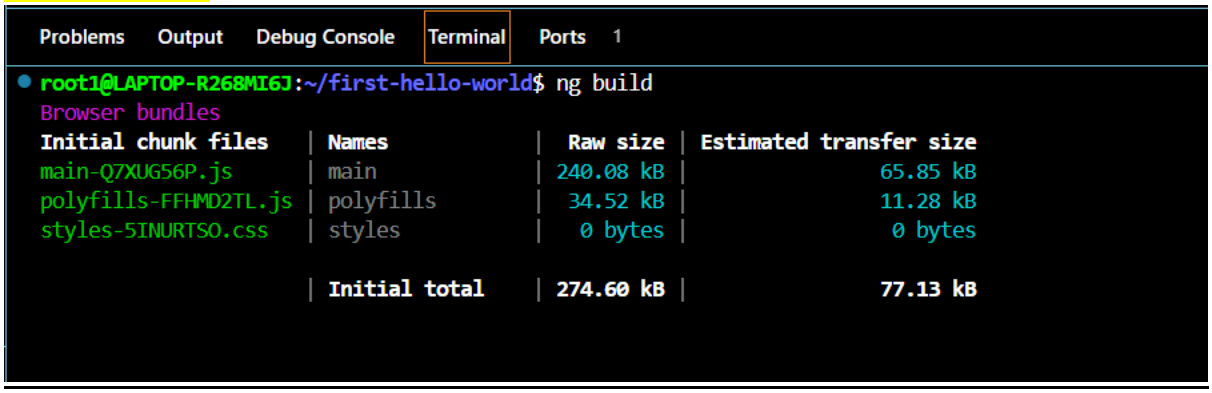
OUTPUT:

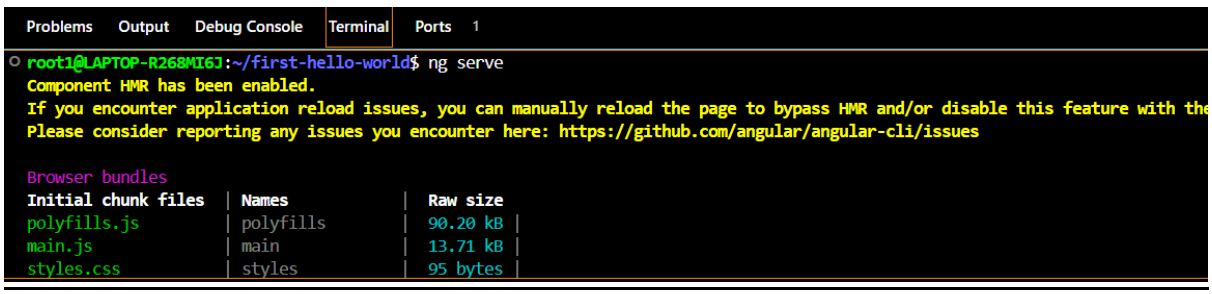


Now we need to install angular extension before coding in cursor:



And then to run this file we use “ng build” and then “ng serve” commands:





Now for establishing routes in angular:

In app.routes.ts we write:

```
import { Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';
export const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'about', component: AboutComponent },
  { path: 'contact', component: ContactComponent },
  { path: '**', redirectTo: '' } // Wildcard route for unknown paths to
redirect to home
];
```

Then we create separate components using this command:

```
rootjinesh@DESKTOP-KN25QO6:~/first-hello-world/src$ ng g c home
CREATE src/home/home.component.css (0 bytes)
CREATE src/home/home.component.html (19 bytes)
CREATE src/home/home.component.spec.ts (578 bytes)
CREATE src/home/home.component.ts (206 bytes)
rootjinesh@DESKTOP-KN25QO6:~/first-hello-world/src$ ng g c about
CREATE src/about/about.component.css (0 bytes)
CREATE src/about/about.component.html (20 bytes)
CREATE src/about/about.component.spec.ts (585 bytes)
CREATE src/about/about.component.ts (210 bytes)
rootjinesh@DESKTOP-KN25QO6:~/first-hello-world/src$ ng g c contact
```

Then in each folder's ".ts" file we add the following code:

About.component.ts:

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-about',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './about.component.html',
  styleUrl: './about.component.css'
})
export class AboutComponent {
  constructor() {}
}
```

contact.component.ts:

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-contact',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './contact.component.html',
  styleUrl: './contact.component.css'
})
export class ContactComponent {
  constructor() {}
}
```

home.component.ts:

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-home',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './home.component.html',
  styleUrl: './home.component.css'
})
export class HomeComponent {
  constructor() {}
}
```

And then in app.component.ts we import following libraries and write following code:

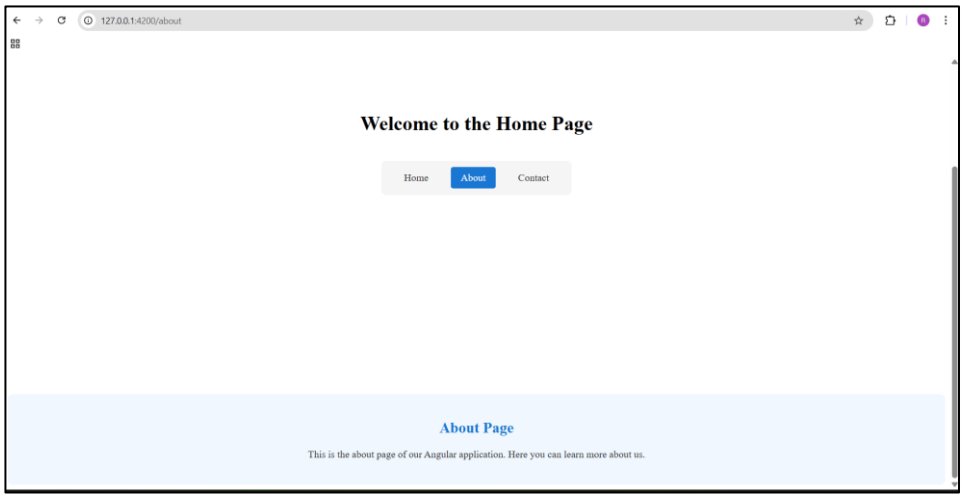
```
import { Component } from '@angular/core';
import { RouterOutlet, RouterLink, RouterLinkActive } from '@angular/router';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet, RouterLink, RouterLinkActive],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'second-app';
  currentColor = '#cccccc';

  changeColor(color: string) {
    this.currentColor = color;
  }
}
```

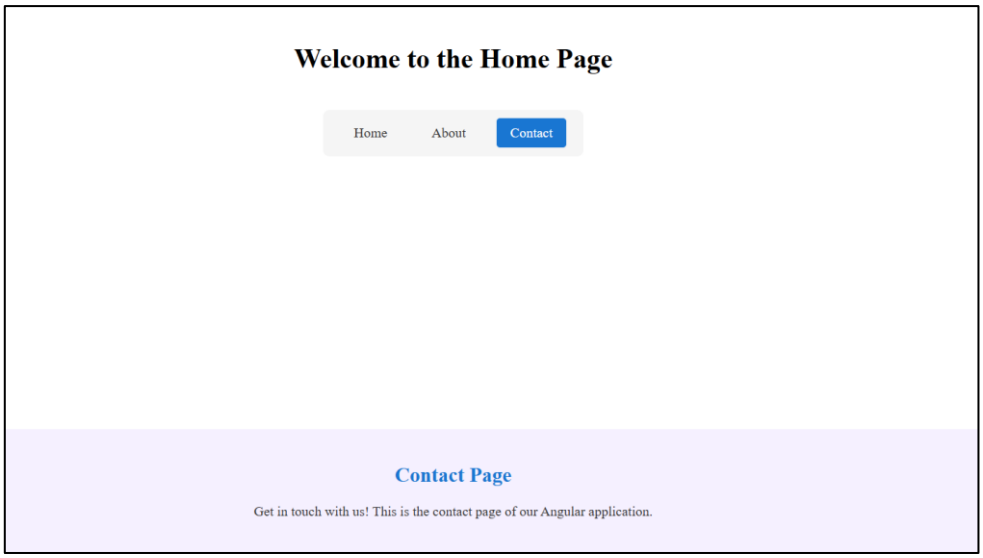
And then we can modify the code from app.html and app.css:

```
<main class="main">
  <div class="container">
    <h1>Welcome to the Home Page</h1>
    <nav class="navigation">
      <a routerLink="/" routerLinkActive="active"
[routerLinkActiveOptions]="{exact: true}">Home</a>
      <a routerLink="/about" routerLinkActive="active">About</a>
      <a routerLink="/contact" routerLinkActive="active">Contact</a>
    </nav>
    <router-outlet></router-outlet>
  </div>
  <style>
    .container {
      text-align: center;
      padding: 20px;
    }
    .navigation {
      margin: 20px 0;
      padding: 10px;
      background-color: #f5f5f5;
      border-radius: 8px;
    }
    .navigation a {
      display: inline-block;
```

```
    padding: 8px 16px;
    margin: 0 10px;
    color: #333;
    text-decoration: none;
    border-radius: 4px;
    transition: all 0.3s ease;
  }
  .navigation a:hover {
    background-color: #e0e0e0;
  }
  .navigation a.active {
    background-color: #1976d2;
    color: white;
  }
</style>
```

OUTPUT:





PYTHON LEETCODE QUESTIONS:

Description | Accepted × | Editorial | Solutions | Submissions

1768. Merge Strings Alternately

Solved

Easy | Topics | Companies | Hint

You are given two strings `word1` and `word2`. Merge the strings by adding letters in alternating order, starting with `word1`. If a string is longer than the other, append the additional letters onto the end of the merged string.

Return *the merged string.*

Example 1:

```
Input: word1 = "abc", word2 = "pqr"
Output: "apbqcr"
Explanation: The merged string will be merged as so:
word1:  a   b   c
word2:    p   q   r
merged: a p b q c r
```

Example 2:

```
Input: word1 = "ab", word2 = "pqrs"
Output: "apbqrs"
Explanation: Notice that as word2 is longer, "rs" is appended to the end.
word1:  a   b
word2:    p   q   r   s
merged: a p b q   r   s
```

Example 3:

Code

Python3 | Auto

```python
class Solution:
    def mergeAlternately(self, word1: str, word2: str) -> str:
        s=''
        i, j = 0, 0
        while i < len(word1) and j < len(word2):
            s+=word1[i]+word2[j]
            i+=1
            j+=1
        while i<len(word1):
            s+=word1[i]
            i+=1
        while j<len(word2):
            s+=word2[j]
            j+=1
        return s
```

Saved | Ln 13, Col 16

Testcase | Test Result

Accepted Runtime: 44 ms

• Case 1 • Case 2 • Case 3

Input

word1 =

"abc"

Description | Editorial | Solutions | Accepted × | Submissions

1071. Greatest Common Divisor of Strings

Solved

Easy | Topics | Companies | Hint

For two strings `s` and `t`, we say "`t` divides `s`" if and only if `s = t + t + t + ... + t + t` (i.e., `t` is concatenated with itself one or more times).

Given two strings `str1` and `str2`, return *the largest string* `x` *such that* `x` *divides both* `str1` *and* `str2`.

Example 1:

```
Input: str1 = "ABCABC", str2 = "ABC"
Output: "ABC"
```

Example 2:

```
Input: str1 = "ABABAB", str2 = "ABAB"
Output: "AB"
```

Example 3:

```
Input: str1 = "LEET", str2 = "CODE"
Output: ""
```

Constraints:

- `1 <= str1.length, str2.length <= 1000`
- `str1` and `str2` consist of English uppercase letters.

Code

Python3 | Auto

```python
class Solution:
    def gcdOfStrings(self, str1: str, str2: str) -> str:
        def check(a, b):
            c=""
            while len(c) < len(b):
                c+=a
            return c == b
        for i in range(min(len(str1), len(str2)), 0,-1):
            x =str1[:i]
            if check(x, str1) and check(x, str2):
                return x
        return ""

```

```python
class Solution:
    def gcdOfStrings(self, str1: str, str2: str) -> str:
        if str1 + str2 == str2 + str1:

            x = gcd(len(str1),len(str2))

            return str1[:x]

        else:

            return ""
```

Description | Editorial | Solutions | Submissions

There are `n` kids with candies. You are given an integer array `candies`, where each `candies[i]` represents the number of candies the `i`th kid has, and an integer `extraCandies`, denoting the number of extra candies that you have.

Return *a boolean array* `result` *of length* `n`*, where* `result[i]` *is* `true` *if, after giving the* `i`th *kid all the* `extraCandies`*, they will have the* ***greatest*** *number of candies among all the kids, or* `false` *otherwise.*

Note that **multiple** kids can have the **greatest** number of candies.

Example 1:

```
Input: candies = [2,3,5,1,3], extraCandies = 3
Output: [true,true,true,false,true]
Explanation: If you give all extraCandies to:
- Kid 1, they will have 2 + 3 = 5 candies, which is the greatest among the kids.
- Kid 2, they will have 3 + 3 = 6 candies, which is the greatest among the kids.
- Kid 3, they will have 5 + 3 = 8 candies, which is the greatest among the kids.
- Kid 4, they will have 1 + 3 = 4 candies, which is not the greatest among the kids.
- Kid 5, they will have 3 + 3 = 6 candies, which is the greatest among the kids.
```

Example 2:

```
Input: candies = [4,2,1,1,2], extraCandies = 1
Output: [true,false,false,false,false]
Explanation: There is only 1 extra candy.
Kid 1 will always have the greatest number of candies, even if a different kid is given the extra
candy.
```

Example 3:

```
Input: candies = [12,1,12], extraCandies = 10
Output: [true,false,true]
```

Constraints:

- `n == candies.length`
- `2 <= n <= 100`
- `1 <= candies[i] <= 100`
- `1 <= extraCandies <= 50`

Code

Python3 | Auto

```python
class Solution:
    def kidsWithCandies(self, candies: List[int], extraCandies: int) -> List[bool]:
        max_candies = max(candies)
        most_candies=[(child_candies + extraCandies) >= max_candies for child_candies in candies]
        return most_candies
```

Description | Editorial | Solutions | Submissions

151. Reverse Words in a String

Medium | Topics | Companies

Given an input string `s`, reverse the order of the **words**.

A **word** is defined as a sequence of non-space characters. The **words** in `s` will be separated by at least one space.

Return *a string of the words in reverse order concatenated by a single space.*

Note that `s` may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces.

Example 1:

```
Input: s = "the sky is blue"
Output: "blue is sky the"
```

Example 2:

```
Input: s = "  hello world  "
Output: "world hello"
Explanation: Your reversed string should not contain leading or trailing spaces.
```

Example 3:

```
Input: s = "a good   example"
Output: "example good a"
Explanation: You need to reduce multiple spaces between two words to a single space in the reversed string.
```

Code

Python3 | Auto

```python
class Solution:
    def reverseWords(self, s: str) -> str:
        return ' '.join(reversed(s.split()))

```

238. Product of Array Except Self

Medium | Topics | Companies | Hint

Given an integer array `nums`, return *an array* `answer` *such that* `answer[i]` *is equal to the product of all the elements of* `nums` *except* `nums[i]`.

The product of any prefix or suffix of `nums` is **guaranteed** to fit in a **32-bit** integer.

You must write an algorithm that runs in `O(n)` time and without using the division operation.

Example 1:

```
Input: nums = [1,2,3,4]
Output: [24,12,8,6]
```

Example 2:

```
Input: nums = [-1,1,0,-3,3]
Output: [0,0,9,0,0]
```

Constraints:

- `2 <= nums.length <= 10^5`
- `-30 <= nums[i] <= 30`
- The input is generated such that `answer[i]` is **guaranteed** to fit in a **32-bit** integer.

Follow up: Can you solve the problem in `O(1)` extra space complexity? (The output array **does not** count as extra space for space complexity analysis.)

Code

Python3 | Auto

```python
class Solution:
    def productExceptSelf(self, nums: List[int]) -> List[int]:
        zeros = 0
        idx = -1
        prod = 1
        # Count zeros and track the index of the zero
        for i in range(len(nums)):
            if nums[i] == 0:
                zeros += 1
                idx = i
            else:
                prod *= nums[i]

        res = [0] * len(nums)
        # If no zeros, calculate the product for all elements
        if zeros == 0:
            for i in range(len(nums)):
                res[i] = prod // nums[i]
        # If one zero, set product only at the zero's index
        elif zeros == 1:
            res[idx] = prod
        return res

```

Saved | Ln 13, Col 1