Angular

**Angular** is a popular open-source framework developed by Google for building dynamic, single-page web applications (SPAs). Angular is a framework which provides complete developer platform for Single Page Application and Progressive Web Applications. It uses TypeScript, a superset of JavaScript, and provides a powerful set of tools for developing complex applications with a focus on maintainability, scalability, and performance.

**Key Concepts of Angular:**

1. **Component-based Architecture**:
   * Angular applications are built using components, which are self-contained building blocks with their own logic and templates.
   * A component consists of an HTML template for the UI, a TypeScript class for handling logic, and metadata that defines how the component should behave.
2. **Modules**:
   * Angular applications are organized into modules. A module is a container that groups related components, services, and other code.
   * The root module, AppModule, bootstraps the Angular application.
3. **Templates and Data Binding**:
   * Angular uses templates to define the view, and it supports powerful data-binding mechanisms.
     + **Interpolation**: Binding data from the component to the template using {{ }}.
     + **Property Binding**: Setting properties of DOM elements with [property]="expression".
     + **Event Binding**: Handling user actions with (event)="handler()".
     + **Two-way Binding**: Synchronizing data between the component and the UI using [(ngModel)].
4. **Services and Dependency Injection**:
   * Angular promotes the use of services to share logic between components. Services are typically used for things like fetching data from APIs.
   * Dependency injection (DI) is a key feature, allowing Angular to efficiently provide instances of services where needed.
5. **Routing**:
   * Angular provides a powerful routing system to build single-page applications. The Angular Router maps URLs to components and helps manage navigation within the app without reloading the page.
6. **Directives**:
   * Directives are special markers in the DOM that tell Angular to attach a specific behavior to an element. Angular provides built-in directives like \*ngIf and \*ngFor, and you can create custom directives for reusable behaviors.
7. **Forms**:
   * Angular has two approaches to managing forms:
     + **Template-driven forms**: Simple and easy to use, but less scalable.
     + **Reactive forms**: More powerful and suitable for complex scenarios, offering better control over form validation and dynamic form creation.
8. **Pipes**:
   * Pipes are used to transform data in templates. For example, the | date pipe formats a date, and the | uppercase pipe converts text to uppercase. You can also create custom pipes.
9. **Testing**:
   * Angular encourages a test-driven development approach. It comes with built-in tools for unit testing components and services using frameworks like Jasmine and Karma.
10. **Performance Optimization**:
    * Angular provides various techniques for optimizing performance, including Ahead-of-Time (AOT) compilation, lazy loading of modules, and change detection strategies.

**Advantages of Angular:**

* **Structured Code**: Angular’s architecture promotes clean, maintainable, and testable code.
* **TypeScript**: Leveraging TypeScript allows for better tooling, refactoring, and catching errors at compile time.
* **Large Ecosystem**: Angular has a vast ecosystem of tools, libraries, and extensions.
* **Support from Google**: Being developed and maintained by Google ensures regular updates, strong community support, and stability.

**Getting Started with Angular:**

To start building Angular applications, you typically use the Angular CLI (Command Line Interface), which provides a powerful set of tools to scaffold projects, generate components, and run development servers.

1. **Install Angular CLI**:

bash

Copy code

npm install -g @angular/cli

1. **Create a New Angular Application**:

bash

Copy code

ng new my-angular-app

cd my-angular-app

ng serve

1. **Develop**: You can now start building your application by creating components, services, and other features using the Angular CLI and writing code in TypeScript.

Angular is widely used in the industry for building large-scale applications, and its comprehensive framework helps developers manage complex applications with ease.

**Advantages of Angular**:

1. **Component-based Architecture**: Promotes reusability, modularity, and organized code structure.
2. **Two-way Data Binding**: Simplifies synchronization between the model and the view.
3. **TypeScript Support**: Provides static typing, better tooling, and early error detection.
4. **Dependency Injection**: Enhances modularity and testability by decoupling components from their dependencies.
5. **Powerful CLI**: Automates tasks like project setup, scaffolding, testing, and building applications.
6. **Built-in Routing and State Management**: Supports advanced navigation and state handling in single-page applications.
7. **Comprehensive Documentation and Community**: Offers extensive resources and community support for problem-solving and learning.
8. **Performance Optimizations**: Features like Ahead-of-Time (AOT) compilation and lazy loading improve app performance.
9. **Cross-platform Development**: Enables building web, mobile (via Ionic), and desktop applications.
10. **Enterprise-ready**: Suited for large-scale, complex applications due to its robustness and scalability.

**Disadvantages of Angular**:

1. **Steep Learning Curve**: Complex concepts can be overwhelming for beginners.
2. **Verbose Code**: Requires more boilerplate code compared to other frameworks.
3. **Complexity**: Overkill for small or simple applications due to its extensive features.
4. **Performance Issues**: May struggle with very large applications if not optimized properly.
5. **Frequent Updates and Breaking Changes**: Updates often introduce breaking changes that require codebase adjustments.
6. **Heavy Framework**: Larger bundle size can affect initial load times compared to lighter frameworks.
7. **Complex Integration with Legacy Systems**: Challenging to integrate with non-modern JavaScript frameworks or legacy systems.
8. **Verbose Testing**: Writing tests, especially for complex components, can be time-consuming and complicated.
9. **Opinionated Structure**: Enforces specific conventions and architecture, limiting flexibility for developers who prefer more freedom.

**Q: Difference Between Angular and AngularJS**

|  |  |  |
| --- | --- | --- |
| **Category** | **Angular** | **AngularJS** |
| Creator | Google | Google |
| Language supported | JavaScript and Typescript | JavaScript |
| Mobile Development friendly | Compatible for mobile-development | Not compatible |
| Architecture | It uses components and directives. | Support model-view-controller (MVC) and model-view-view-model (MVVM) architectures. |
| Testing | Supports unit testing with Karma | Testing is done through third-party applications |
| CLI | Comes with Angular CLI | No support for CLI |
| Dependency Injection | Uses hierarchal dependency injection | Does not use dependency injection. Uses directives |
| Performance | Supports server-side rendering which offers a speedy performance | Overall performance is slow as compared to Angular |
| Example | Gmail and Upwork | Netflix and Lego |

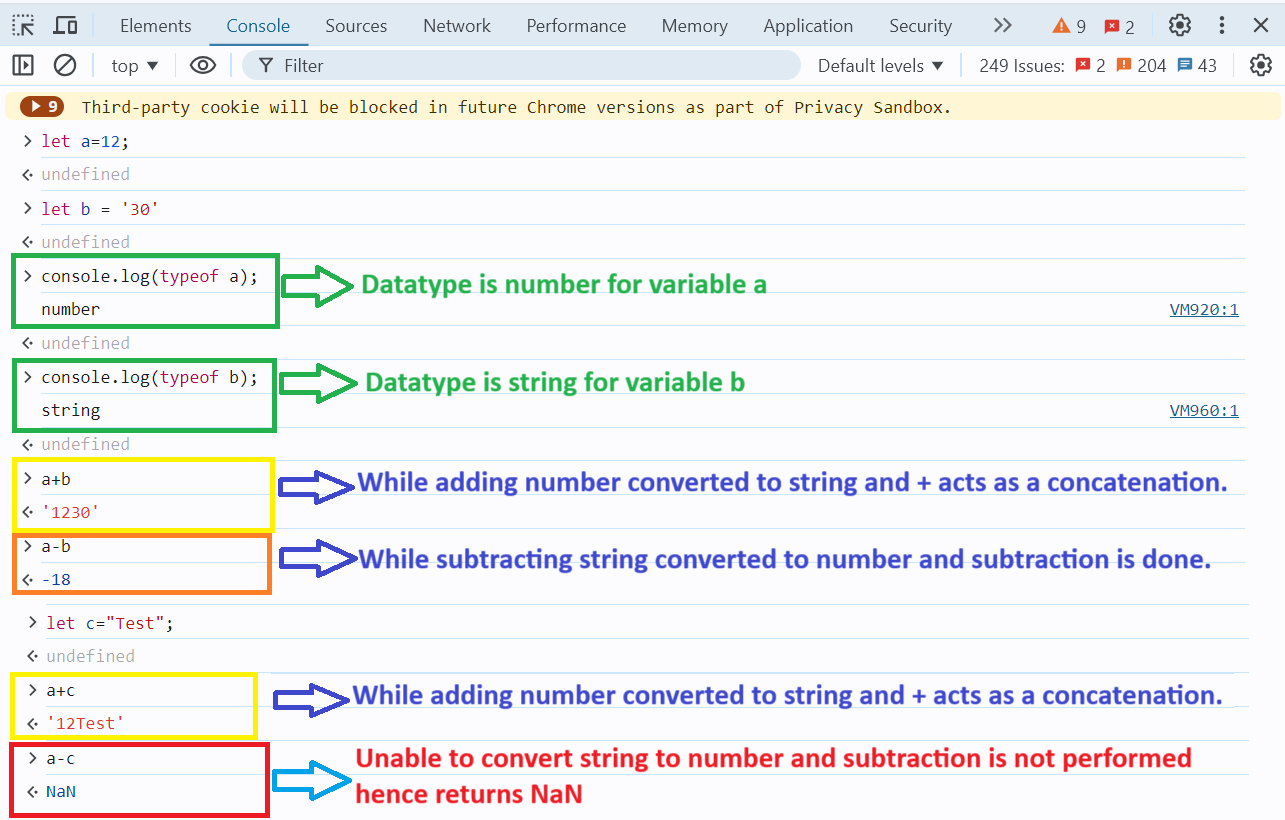
**Q: Difference Between React & Angular:**

|  |  |  |
| --- | --- | --- |
| **Field** | **React** | **Angular** |
|  |  |  |
|  | It is a **JavaScript library**. | Angular is **a framework.** |
| **Written** | React.js written in JavaScript. | Written in Microsoft’s Typescript language, which is a superset of ECMA Script 6 (ES6). |
| **Dependency Injection** | React.js Does not use the DI concept. | Angular Hierarchical DI system used. |
| **Routing** | Routing is not easy in React JS. | Routing is comparatively easy as compare to React JS. |
| **Scalability** | It is highly scalable. | It is less scalable than React JS. |
| **Data Binding** | It supports Uni-directional data binding that is one way data binding. | It supports Bi-directional data binding that is two data binding. |
| **DOM** | It has virtual DOM. | It has regular DOM. |
| **Testing** | It supports Unit Testing. | It supports both Unit testing and Integration testing. |
| **Used as** | React.js is a JavaScript library. As it indicates react js updates only the virtual DOM is present and the data flow is always in a single direction. | Angular is a framework. Angular updates the Real DOM and the data flow is ensured in the architecture in both directions. |
| **Released** | It was released in 2013. | It was released in 2010. |
| **Architecture** | React.js is more simplified as it follows MVC ie., Model View Control. This like angular includes features such as navigation but this can be achieved only with certain libraries like Redux and Flux. Needs more configuration and integration. | The architecture of angular on the other hand is a bit complex as it follows MVVM models ie., Model View-ViewModel. This includes lots of tools and other features required for navigation, routing, and various other functionalities. |
| **Performance** | React.js holds JSX hence the usage of HTML codes and syntax is enabled. But this doesn’t make react js a subset of HTML. This is purely JavaScript-based. | Angular, on the other, is a mere subset of HTML. |
| **Preference** | React.js is preferred when the dynamic content needed is intensive. As react js holds more straightforward programming and since it is reliable many apps such as Instagram, Facebook, and Twitter still prefer to react js over angular. | Angular is platform-independent and hence is compatible to work in any platform. Hence, the HTML app which is compatible with all the browsers can prefer angular. One major app which uses angular is YouTube. |

**Q: Difference Between TypeScript and JavaScript**

* TypeScript is known as an Object-oriented programming language whereas JavaScript is a prototype-based language.
* TypeScript has a feature known as Static typing but JavaScript does not support this feature.
* TypeScript supports Interfaces but JavaScript does not.

|  |  |  |
| --- | --- | --- |
| **Feature** | **TypeScript** | **JavaScript** |
| Typing | Provides static typing. | Dynamically typed. |
| Tooling | Comes with IDEs and code editors. | Limited built-in tooling. |
| Syntax | Similar to JavaScript, with additional features like static typing. | Standard JavaScript syntax. |
| Compatibility | Backward compatible with JavaScript. | Cannot run TypeScript in JavaScript files. |
| Debugging | Stronger typing can help identify errors. | May require more debugging and testing. |
| Type | Object Oriented Programming Language. | Prototype Based Language. |
| Learning curve | Can take time to learn additional features. | Standard JavaScript syntax is familiar. |
| Example | Let a=20;  A=’Mahesh’ //Invalid | Let a=20;  A=’Mahesh’ //Valid |
| Execution | Browser doesn’t understand TypeScript. | Browser understands JavaScript, html, css. |
|  | Needs to be converted to JavaScript before it reaches to browser. | Browser will directly understand the code. |



This is the official bug reported over angular website also, mentioning Angular is dynamically typed, meaning that the datatype will change according to the value.

And that will create big problem as like below

**Ex:**

let a=33;

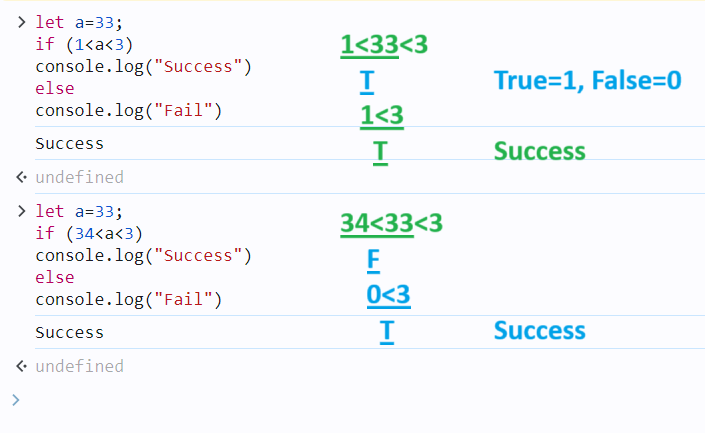
if (1<a<3)

console.log("Success")

else

console.log("Fail")

Success



**Basics of TypeScript:**

TypeScript is an open-source programming language developed by Microsoft that builds on JavaScript by adding static types. TypeScript is a superset of JavaScript, meaning any valid JavaScript code is also valid TypeScript code. However, TypeScript adds additional syntax to allow developers to specify types (such as string, number, boolean, etc.) for variables, function parameters, and return values.

To install TypeScript, you typically use npm (Node Package Manager), which comes with Node.js. (npm cannot be separately installed it comes with Node.js)

**1. Install Node.js**

* **Node.js**: If you haven’t installed Node.js yet, download and install it from [nodejs.org](https://nodejs.org). Installing Node.js will also install **npm (Node Package Manager).**
* **To update** npm use: **npm install -g npm**

**2. Install TypeScript Globally**

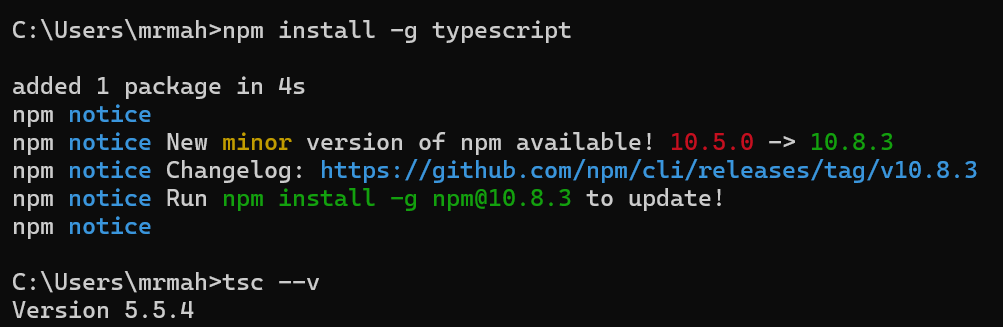
* Open your terminal (Command Prompt, PowerShell, or a terminal in your code editor) and run the following command to install TypeScript globally:

**npm install -g typescript@latest**

This installs the TypeScript compiler globally on system, so you can use it from any directory.

**3. Verify the Installation**

* To verify that TypeScript is installed correctly, run the following command in your terminal: **tsc –version**
* To Update Typescript use : **npm install -g typescript**



**4. Compile TypeScript Code**

* Once installed, you can compile a TypeScript file (.ts) into JavaScript by using the tsc command. For example: **tsc example.ts**

**Optional: Initialize a TypeScript Project**

* If you are starting a new TypeScript project, you can initialize it with a tsconfig.json file by running: **tsc --init**

This file helps configure TypeScript options such as the target ECMAScript version, module resolution, and more. Once installed and configured, ready to start developing in TypeScript!

**Note:**

Npm : Node Package Manager

-g : To install it globally in system. If not mentioned it will install in current folder.

@latest/version no: Will install particular version or latest version.

* In Visual Studio Code check the Auto Save option under file menu.
* Browser only understand HTML, CSS, JavaScript, it wont understand TypeScript.
* Install **Live Server** extension in VSCode editor.
* JavaScript Code must be added at the end of body not at the end of head and CSS at end of head section.

First.html:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

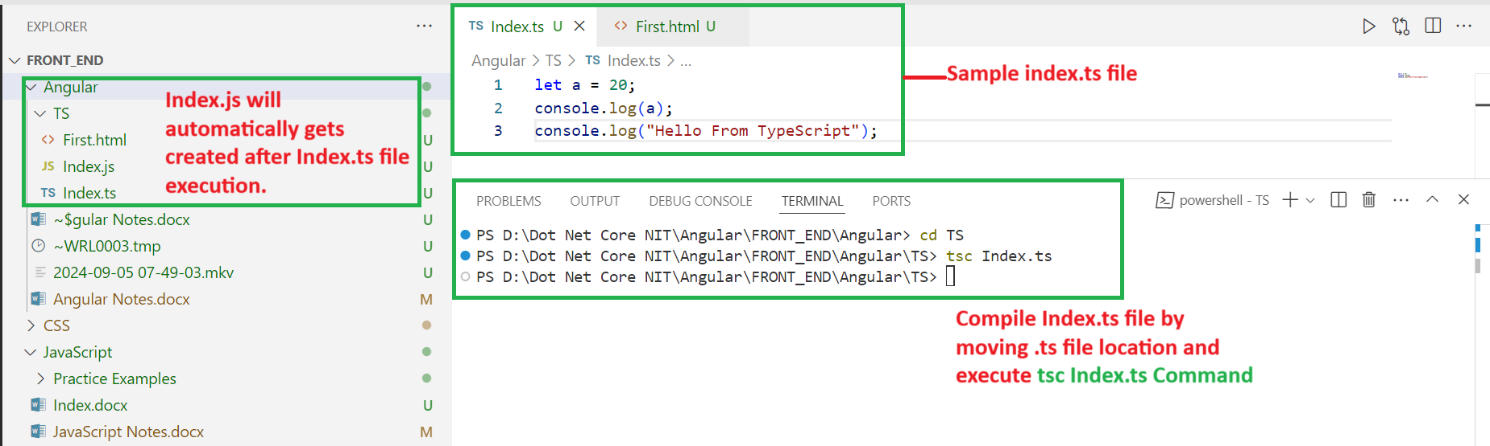
</head>

<body>

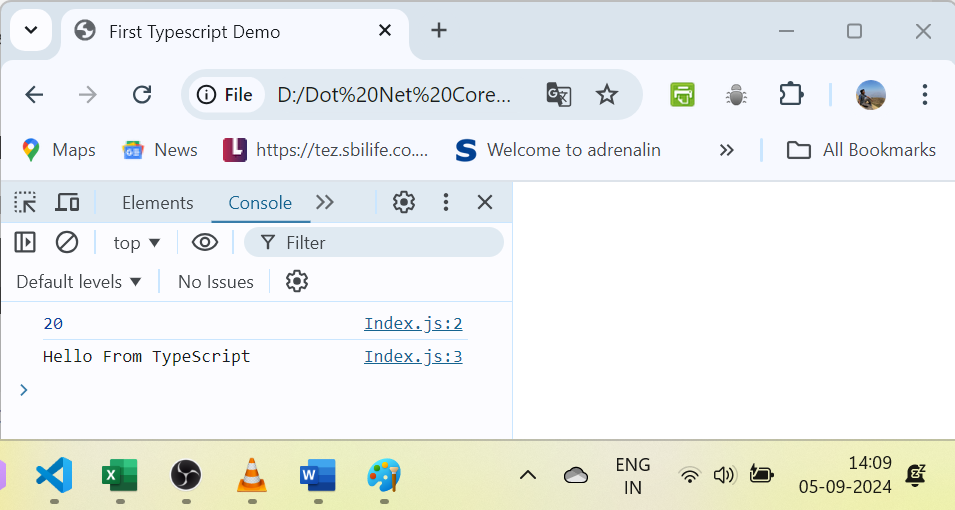
    <script src="Index.js"></script>

</body>

</html>



Output:



The watch mode in TypeScript allows you to automatically recompile your TypeScript code whenever changes are made to your .ts files. This is useful during development, as it eliminates the need to manually run the tsc command every time you make a change.

**How to Use the watch Command**

You can enable watch mode by using the --watch (or -w) flag with the tsc command.

**1. Basic Usage**

To watch a single TypeScript file for changes and automatically recompile it, run:

**tsc --watch <filename>.ts**

For example, if you have a file named app.ts, you would run:

**tsc --watch app.ts**

This will watch the file for any changes and recompile it automatically when changes are detected.

**2. Watch an Entire Project**

If you have multiple TypeScript files in your project and you want to watch all of them, it's more efficient to use the tsconfig.json configuration file. You can enable watch mode for the entire project by running:

**tsc --watch**

This will watch all the files specified in your tsconfig.json and automatically recompile them whenever changes are made.