**Angular Framework**

Angular is a popular framework for building web applications. It's developed and maintained by Google, which assures its users about its reliability and consistency.

Here are a few key details about the Angular framework:

1. **Language**: Angular is written in TypeScript, a statically typed superset of JavaScript. This means it has additional features not found in regular JavaScript, which can help developers write more robust code.
2. **Components**: Angular is based on a component architecture. A component is a self-contained chunk of user interface (UI), such as a button or a form. This architecture makes it easier to organize code and reuse parts of your application's UI.
3. **Two-way data binding**: Angular provides a feature called two-way data binding that keeps the model and the view in sync. In other words, any changes to the model are automatically reflected in the view and vice versa. This makes it easier to write interactive applications.
4. **Dependency Injection**: Angular uses a design pattern called dependency injection, which can make your code more modular, more testable, and easier to change.
5. **Single Page Applications (SPA)**: Angular is excellent for building SPAs, which are web applications that load a single HTML page and dynamically update that page as the user interacts with the app.
6. **Tooling**: Angular includes powerful tools like the Angular CLI (Command Line Interface) which can generate code for you and handle various development tasks.
7. **Community and Support**: Angular has a huge community of developers and extensive resources and libraries available online, making it easier to find solutions to problems and to learn new techniques.

In simple terms, Angular is a powerful and comprehensive framework that helps you to build complex web applications. It's like a big toolkit that gives you everything you need to construct a web app, from simple components to complex interactions and data management.

**Why angular2 is better compared to Angular1**

Angular 2 represented a significant step forward from Angular 1 (or AngularJS) in terms of architecture, programming paradigms, and overall efficiency. Here are some reasons why Angular 2 is considered better compared to AngularJS:

1. **Performance**: Angular 2 was designed with a focus on speed and performance. It's significantly faster than AngularJS, especially when it comes to complex and dynamic web applications.
2. **Component-based architecture**: While AngularJS was based on the MVC (Model-View-Controller) architecture, Angular 2 shifted to a component-based architecture. This makes applications easier to understand, develop, and test. It also facilitates code reusability.
3. **TypeScript**: Angular 2 adopted TypeScript, a statically-typed superset of JavaScript, as its primary language. TypeScript provides powerful features like type annotations, compile-time error checking, and advanced tooling services, leading to more maintainable code and fewer runtime errors.
4. **Mobile Support**: Angular 2 was built from the ground up with mobile support in mind, unlike AngularJS which was primarily targeted towards desktop applications.
5. **Improved Dependency Injection (DI)**: DI is a design pattern in which a class receives its dependencies from external sources rather than creating them itself. Angular 2's improved DI system makes components easier to manage, test, and understand.
6. **Routing**: The router in Angular 2 offers multiple new features compared to the AngularJS router. Features like child routers and route lifecycle hooks provide a lot more control to developers.
7. **Reactive Programming**: Angular 2 is built around reactive programming using observables, which makes it easier to manage asynchronous data flows.

Remember that while Angular 2+ versions provide many improvements, the choice between AngularJS and Angular 2+ depends on the specific needs and constraints of a project, including factors like team expertise, project requirements, and legacy code considerations.

Top of Form

Regenerate response

**Few Web sites built on Angular**

Several well-known companies and organizations have built their web applications using Angular. As of my knowledge cut-off in September 2021, here are a few examples:

1. **Google**: Angular was developed by Google and is used in many of their products. For example, Google Ads, Google Cloud Platform, and the Google Shopping platform are known to use Angular.
2. **Microsoft**: Microsoft has used Angular in several of its products including Office Home and Xbox Live.
3. **IBM**: IBM has adopted Angular in some of their products and services.
4. **Forbes**: The well-known business magazine uses Angular to provide a dynamic and interactive experience on their website.
5. **Delta Airlines**: The airline uses Angular for their check-in system, which allows customers to check into their flights online.
6. **BMW**: The German car manufacturer's USA site is an example of a major commercial site built using Angular.
7. **Upwork**: The global freelancing platform uses Angular for its website.
8. **Deutsche Bank**: The corporate banking section of Deutsche Bank uses Angular for its web applications.

Remember, the technology used to build a website can change over time as the company's requirements change and new technologies emerge. Therefore, it's always good to use an up-to-date source when you're looking for this kind of information.

Bottom of Form

**Where can Angular ne used?**

Angular is a versatile framework that can be used to build a wide variety of applications. Here are some types of applications you can design using Angular:

1. **Single Page Applications (SPAs)**: SPAs are web applications that load a single HTML page and dynamically update that page as the user interacts with the app. Angular is perfect for these because of its robust data binding and dependency injection features.
2. **Progressive Web Apps (PWAs)**: PWAs are web applications that can function like desktop or mobile applications. They can work offline, send push notifications, and be installed on a user's device. Angular has built-in service workers that simplify the process of building PWAs.
3. **Enterprise Web Applications**: Due to its modular nature, Angular is well-suited for large scale, enterprise-level applications. It offers features like lazy loading, which allows you to load features of your app on demand.
4. **Dynamic Web Applications**: Angular is designed for building dynamic web applications where contents of the application can change dynamically based on user input and other factors.
5. **E-commerce Apps**: Angular can be used to create engaging and interactive e-commerce applications, thanks to its robust features like two-way data binding, form validations, HTTP services, and more.
6. **Server-side Rendered (SSR) Apps**: With Angular Universal, you can render applications on the server, which can improve performance and make your app more SEO-friendly.
7. **Mobile Apps**: With frameworks like Ionic, you can use Angular to build cross-platform mobile applications.
8. **Content Management Systems (CMS)**: Angular's ability to separate the view from the data makes it a good choice for building CMS.
9. **Real-time Applications**: Applications like instant messaging or live-streaming platforms can be built using Angular, thanks to its efficient data binding and ability to handle dynamic content.

Remember, while Angular is a powerful tool, it's not always the best solution for every project. The choice of framework should be based on the specific requirements of the application you're building.

**Where Angular is not best fit?**

Angular is a powerful and versatile framework that can be used to develop a wide range of web applications. However, there are some scenarios where it might not be the most suitable choice. Here are some examples:

1. **Static Websites**: If you're developing a static website that doesn't require any dynamic content or complex user interactions, using Angular could be overkill. Simple HTML, CSS, and maybe a touch of JavaScript would be more suitable and efficient for this kind of project.
2. **Lightweight Applications**: Angular is a fairly heavy framework due to its wide array of features and functionalities. For small, lightweight applications, a simpler library like React or Vue.js could be more efficient and easier to implement.
3. **Microservices UI**: In cases where you need to develop a micro front-end application, each part of the application is independent and might be developed by different teams. For such scenarios, lighter and more flexible libraries like React might be more suitable.
4. **Server-side Heavy Operations**: Angular runs in the user's browser and is not designed for heavy server-side operations like interacting with a database or file manipulation on the server. For these operations, you'll need a backend language like Node.js, Python, or Java, among others.
5. **Real-time Multiplayer Games**: While Angular can certainly handle some real-time updates via WebSockets or similar technologies, it's not built to handle the rapid, continuous updates and high-performance rendering needed for real-time multiplayer games. For that, a game development platform would be more appropriate.
6. **Native Mobile Apps**: While Angular can be used with frameworks like Ionic to create hybrid mobile apps, for high performance and fully native mobile apps, you'd typically use a platform like React Native or Swift (for iOS) or Kotlin (for Android).

Remember, these are not hard and fast rules, and skilled developers can often work around such limitations. However, it's often more efficient and effective to choose the right tool for the job.

Top of Form

Bottom of Form

Several well-known companies and organizations have built their web applications using Angular. As of my knowledge cut-off in September 2021, here are a few examples:

1. **Google**: Angular was developed by Google and is used in many of their products. For example, Google Ads, Google Cloud Platform, and the Google Shopping platform are known to use Angular.
2. **Microsoft**: Microsoft has used Angular in several of its products including Office Home and Xbox Live.
3. **IBM**: IBM has adopted Angular in some of their products and services.
4. **Forbes**: The well-known business magazine uses Angular to provide a dynamic and interactive experience on their website.
5. **Delta Airlines**: The airline uses Angular for their check-in system, which allows customers to check into their flights online.
6. **BMW**: The German car manufacturer's USA site is an example of a major commercial site built using Angular.
7. **Upwork**: The global freelancing platform uses Angular for its website.
8. **Deutsche Bank**: The corporate banking section of Deutsche Bank uses Angular for its web applications.

Remember, the technology used to build a website can change over time as the company's requirements change and new technologies emerge. Therefore, it's always good to use an up-to-date source when you're looking for this kind of information.

Top of Form

**What are components?**

Component-based technology refers to a software development approach where the software is structured as a set of loosely coupled, reusable components. Each component is a self-contained entity with its own functionality and behavior and can be independently developed and tested. Components can be combined and reused across different projects or parts of the same project.

Advantages of Component-Based Technology:

1. **Reusability**: Components can be reused in different parts of an application or even across different applications, which can save development time and effort.
2. **Modularity**: Applications are structured as a collection of components, which makes the codebase easier to manage and understand.
3. **Ease of Testing**: Components can be tested independently, which simplifies the testing process and can lead to a higher quality codebase.
4. **Separation of Concerns**: Each component is responsible for a single piece of functionality, which helps keep the codebase organized and improves maintainability.
5. **Scalability**: Applications can be easily scaled by adding new components or updating existing ones.

**Disadvantages of Component-Based Technology:**

1. **Overhead**: For simple applications, a component-based approach might be overkill and can lead to unnecessary complexity.
2. **Component Integration**: Components need to communicate with each other, which can sometimes lead to complexity and issues related to data flow and state management.
3. **Quality Variation**: The quality of components can vary, especially if they are sourced from different developers or third-party libraries.
4. **Dependency**: Using components from third-party sources can lead to dependency on those sources. If they stop maintaining the component or it has security vulnerabilities, it could affect your application.
5. **Design Constraints**: Components have defined interfaces for communication, which can sometimes limit flexibility in the application design.

Despite these potential disadvantages, the component-based approach is widely used in modern software development due to its many advantages, especially for larger, more complex applications. Many popular frameworks like React, Angular, Vue.js, and others use a component-based architecture.

Top of Form

Bottom of Form

**Component based technologies**

Component-based technologies encourage code reusability, ease of maintenance, and separation of concerns. Here are a few examples of popular component-based frameworks and libraries:

1. **React**: Developed by Facebook, React is a JavaScript library that is used for building user interfaces, especially for single-page applications. It encourages the development of reusable UI components, which can help manage the display of data as it changes over time.
2. **Angular**: As we discussed before, Angular (from Angular 2 onwards) is a TypeScript-based open-source framework developed by Google. It shifted from an MVC architecture in AngularJS to a component-based architecture, allowing for more modular and maintainable code.
3. **Vue.js**: Vue.js is an open-source JavaScript framework for building user interfaces and single-page applications. It also utilizes a component-based architecture, which allows for reusability and encapsulation of code.
4. **Svelte**: Svelte is a relatively new JavaScript framework that also uses a component-based architecture. Unlike some other frameworks, Svelte compiles your components into efficient imperative code that surgically updates the DOM, resulting in fast runtime performance.
5. **Blazor**: Blazor is a free and open-source web framework developed by Microsoft. It enables developers to create web applications using C# and HTML, using a component-based architecture.
6. **Ember.js**: Ember.js is an open-source JavaScript web framework that uses a component-service pattern. It allows developers to create scalable single-page applications by incorporating common idioms and best practices into the framework.
7. **Preact**: Preact is a lightweight alternative to React. It's faster and smaller, but maintains largely the same API, making it an easy switch for developers already familiar with React.

These technologies all promote the use of components as a fundamental method for creating efficient, scalable applications.

Top of Form

**Creating applications using Angular**

Bottom of FormAngular CLI (Command Line Interface) is a powerful tool that helps in creating, managing, building, and testing Angular applications. Here are some common commands that are used with Angular CLI:

1. **Installation of Angular CLI**: To install Angular CLI globally on your system, you use the following command:

**npm install -g @angular/cli**

1. **Creating a new Angular application**: To generate a new Angular application, you use the **ng new** command followed by the name of the application. For example:

**ng new my-app**

1. **Running the application**: You can start the local development server with the following command:

**ng serve**

Example:

1. **ng new 1-basics**

? Would you like to add Angular routing? No

? Which stylesheet format would you like to use? CSS

CREATE 1-basics/README.md (1061 bytes)

CREATE 1-basics/.editorconfig (274 bytes)

CREATE 1-basics/.gitignore (548 bytes)

CREATE 1-basics/angular.json (2710 bytes)

CREATE 1-basics/package.json (1039 bytes)

CREATE 1-basics/tsconfig.json (901 bytes)

CREATE 1-basics/tsconfig.app.json (263 bytes)

CREATE 1-basics/tsconfig.spec.json (273 bytes)

CREATE 1-basics/.vscode/extensions.json (130 bytes)

CREATE 1-basics/.vscode/launch.json (470 bytes)

CREATE 1-basics/.vscode/tasks.json (938 bytes)

CREATE 1-basics/src/main.ts (214 bytes)

CREATE 1-basics/src/favicon.ico (948 bytes)

CREATE 1-basics/src/index.html (293 bytes)

CREATE 1-basics/src/styles.css (80 bytes)

CREATE 1-basics/src/app/app.module.ts (314 bytes)

CREATE 1-basics/src/app/app.component.css (0 bytes)

CREATE 1-basics/src/app/app.component.html (23083 bytes)

CREATE 1-basics/src/app/app.component.spec.ts (898 bytes)

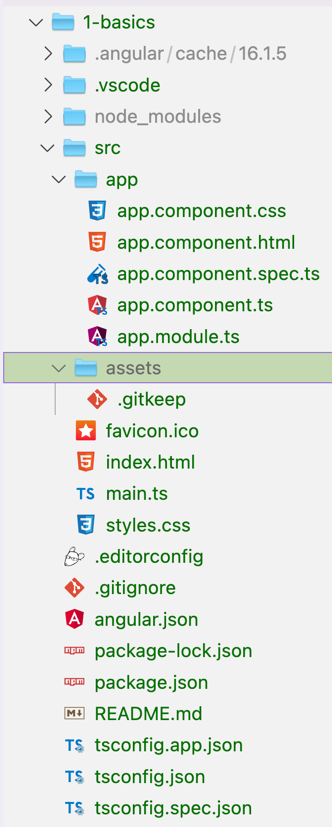
CREATE 1-basics/src/app/app.component.ts (212 bytes)

CREATE 1-basics/src/assets/.gitkeep (0 bytes)

✔ Packages installed successfully.

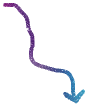
Directory is already under version control. Skipping initialization of git.

1. **cd 1-basics**
2. **ng serve –open**

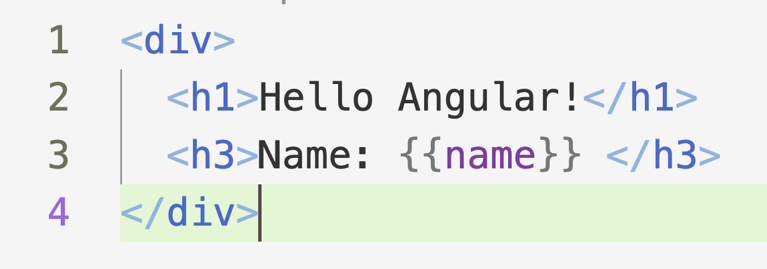








component



View Template

**How angular loads initial files?**

1. Open angular.json file and note **"main": "src/main.ts" at line no 18**
2. Open src/main.ts and note down name of bootstrap module : **AppModule**
3. Opne src/app/app.module.ts file. This files comes with 4 sections
   1. declarations : all inbuilt and custom components must be declared here. Usually all components to be declared here
   2. imports : Any built in or used defined modules must be declared here
   3. providers : all dependency classes should be declared here. Usually service class
   4. bootstrap : name of default component to be loaded should be defined here. Note that AppComponent class is defined here as default
4. Open AppComponet.ts from the src/app folder
5. This files comes with ‘@Component decarator’ which is having 3 parts
   1. selector : this is the name where templateUrl code gets placed. Default name is app-root. This selector name is defined inside index.html file. So all the code of templateUrl goes to index.html <div id=’root’> </div> area
   2. templateUrl : it’s a HTML template where template of a view can be designed. Default name is app.component.html. if you wish to provide view templates with out html, then change templateUrl as ‘template’ and provide HTML view there.
   3. styleUrls : Its an array comes with name of one or more css file names. Default name is app.component.css
6. When you run the application with ng serve command, then app.component.ts loads the data into app.component.html and get displayed at browser.

**Cli customization while creating application**

1. **Skip Installation**: When creating a new Angular application, you might want to skip installing npm packages. This could be useful when you are in an environment with limited network connectivity. To skip the installation, use the --skip-install flag:

**ng new my-app --skip-install**

1. Skip Git: By default, Angular CLI initializes a git repository in the new project. If you want to skip this, use the --skip-git flag:

**ng new my-app1 --skip-git**

If you want the application to automatically open in your browser, you can use the **--open** (or **-o**) option:

**ng serve --open**

1. **Generating Components, Services, Modules, etc.**: Angular CLI can generate various items with the **ng generate** (or **ng g**) command:
   * To generate a component:

**ng generate component component-name** or **ng g c component-name**

* Flags
  + Skiiping tests : --skip-tests
  + Skipping css: --inline-style
  + Skip html : --line-template

* + To generate a service:

**ng generate service service-name** or **ng g s service-name**

* + To generate a module:

**ng generate module module-name** or **ng g m module-name**

1. And so forth for directives, pipes, classes, interfaces, enums, and guards.
2. **Building the application**: To build your application for deployment, you use the **ng build** command. By default, it builds in development mode. To build in production mode, you use the **--prod** option:

**ng build --prod**

1. **Running Tests**: Angular CLI uses Karma for unit tests and you can run them with the following command:

**ng test**

And for end-to-end (E2E) tests, it uses Protractor and you can run them with this command:

**ng e2e**

Remember to run these commands from within the directory that contains your Angular application.

The Angular CLI has many more commands and options that you can use. For a full list, you can check the official Angular CLI documentation or use the **--help** option with the CLI:

**ng –help**

Top of Form

Bottom of Form

Bottom of Form