

Environmental setup

- install node
- install angular cli
 - node -v
 - npm install -g @angular/cli
 - ng --version
- update angular cli
 - npm uninstall -g angular-cli
 - npm cache clean/npm cache verify (if npm > 5)
 - npm install -g @angular/cli@latest
- downgrade angular cli
 - rm -rf node_modules
 - npm uninstall --save-dev angular-cli
 - npm install --save-dev @angular/cli@1.2.2
 - npm install

Building Blocks Of Angular Application

- Components
- Directives
- Services
- Routes
- Modules
- Pipes
- Guards

Understanding Angular Cli Sommands

- **ng new appname**
 - flags (same flags may be available in other commands too)
 - --dry-run
 - --inline-style
 - --inline-template
 - --routing
 - --prefix
 - --style
 - --skip-tests
- **ng serve**
 - flags

- --port
 - --host
 - --ssl
 - --ssl-key
 - --ssl-cert
 - --open
- **ng lint**
 - flags
 - --fix
- **ng test**
- **ng build**
 - flags
 - --prod
 - --build-optimizer
 - --aot
 - --target
- **ng g cl class-name**
- **ng g c component-name**
 - flags
 - --spec
 - --view-encapsulation
 - --flat
 - --selector
- **ng g d directive-name**
- **ng g e enum-name**
- **ng g m module-name**
- **ng g p pipe-name**
- **ng g s service-name**

Understanding Basic Debugging

- `Console.log(), console.debug(), console.warn(), console.info(), console.error()`
- `console.table();`
- `console.time();console.timeEnd();`
- `console.trace();`
- `debug(fn);`
- `debugger;`

- `console.log('%c ' + question, 'background: #222; color: #bada55');`
- `monitor(fn);`

Understanding File Structure

- Angular Cli and Webpack
- package.json
 - ~ (supports patch release)
 - ^ (supports minor release)
 - refer semver for more versioning details
- Angular.json / Angular-cli.json (version <=5)
- main.ts
- Appmodule.ts
- Dependency Injection

HTML Best Practices

- Reset.css
- Getting Most Out of HTML5
 - `<article>`
 - `<aside>`
 - `<details>`
 - `<figcaption>`
 - `<figure>`
 - `<footer>`
 - `<header>`
 - `<main>`
 - `<nav>`
 - `<section>`
 - `<summary>`

Components

- Decorators
- Constructor
- Template
- Styles
- Template Expression / bindings `{{}}`

Structural Directives

- *ngFor
- *ngIf
- *ngSwitch

Conditional Styling

- [style.stylename] ----->apply single style on condition
- [ngStyle] ----->apply multiple styles on condition
- [ngClass] ----->apply multiple class on condition
- [class.classname] ----->apply single class on condition

Component Interaction

- Input []
 - example
`<child-comp [name]="firstname"></child-comp>`
.....
`@Input() name:string;`
- Output (event emitter) ()
 - example
`<child-comp (test)="childEvent($event)"></child-comp>`
.....
`@Output() test:EventEmitter<string> = new EventEmitter();`
- Template reference (or) local variables (#refence)
- Events
 - example (built in events)
 - (focus)="myMethod()"
 - (blur)="myMethod()"
 - (submit)="myMethod()"
 - (scroll)="myMethod()"
 - (cut)="myMethod()"
 - (copy)="myMethod()"
 - (paste)="myMethod()"
 - (keydown)="myMethod()"
 - (keypress)="myMethod()"
 - (keyup)="myMethod()"
 - (mouseenter)="myMethod()"

- (mousedown)="myMethod()"
- (mouseup)="myMethod()"
- (click)="myMethod()"
- (dblclick)="myMethod()"
- (drag)="myMethod()"
- (dragover)="myMethod()"
- (drop)="myMethod()"

- Input, Output Aliasing
 - @Output('test') testChild = new EventEmitter<any>();
 - @Input('name') firstName : string;

Troube Shooting Component

- Declarations
- Circular Dependency
- Safe Traversal ?
- ngNonBindable -->{{ raw word }}

Syntax Usage

- <https://angular.io/guide/cheatsheet>

Component Style Guide

- one component in one file
- small functions: max line 75
- File naming convention: filename.type.ts
- use - for descriptive name: shopping-kart.component.ts
- Do use **UpperCamelCase** for class names.
- Do match the name of the symbol to the name of the file.
- selector name should be seperated with - and must have an prefix(app-shopping-kart) **kebab-case**
- const - **lowerCamelCase**
- Interface - **UpperCamelCase** (avoid I prefix)
- Properties and methods - **lowerCamelCase**
- leaving one empty line between third party imports and application imports.
- Always extract styles and templates to external files.
- Always use decorators for input and output.
- Avoid aliasing input and output

Deployment

- Production Build
- Deploying in Server
- live-server
- now

Custom Directives

- @Directive() decorator
- ElementRef
- TemplateRef
- ViewContainerRef
- Renderer2
- @HostListener()
- @HostBinding()

Pipes

- Built in pipes
 - Date pipe
 - Currency pipe
 - Decimal pipe
 - TitleCase pipe
 - LowerCase pipe
 - UpperCase pipe
 - SlicePipe
 - PercentPipe
 - Json pipe
- Custom pipes
 - @Pipe() decorator
 - transform()
 - impure pipe

Forms

- Template driven
 - [(ngModel)]
 - ngForm
 - states of formcontrol

- touched
- untouched
- dirty
- pristine
- valid
- invalid
- Auto applied css classes
 - .ng-touched
 - .ng-untouched
 - .ng-dirty
 - .ng-pristine
 - .ng-valid
 - .ng-invalid
- `form.reset();`