app.pluralsight.com/ilx/video-courses/12d7d1cc-b3ec-40b2-b13e-3c35e8ba07a5/97fbb958-e828-4258-8fc0-531723bc5206/a2752987-674c...

Component

app.component.ts

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

@Component({
    selector: 'pm-root',
    template: `
    <div><h1>{{pageTitle}}</h1>
        <div>My First Component</div>
    </div>
    `
})
export class AppComponent {
 pageTitle: string = 'Acme Product Management';
}
```

Import

Metadata & Template

Class

Exit Course

Angular: Getting Started

by Deborah Kurata

2 Introduction 13m 9s

3 First Things First 25m 17s

4 Introduction to Components 30m 42s

- Introduction 1m 34s
- What is a Component? 1m 57s
- Creating the Component Class 2m 7s
- Defining the Metadata with a Decorator 2m 25s
- Importing What We Need 1m 37s
- Demo: Creating the App Component 4m 55s
- Bootstrapping Our App Component 5m 2s
- Demo: Bootstrapping Our App Component 3m 41s
- Something's Wrong! 4m 3s
- Checklists and Summary

Autoplay

app.pluralsight.com/ilx/video-courses/12d7d1cc-b3ec-40b2-b13e-3c35e8ba07a5/84d6d5db-2f3d-4f1e-8278-7d9029236920/d920a2c3-698c...

Data Binding

DOM

Interpolation: {{pageTitle}}

Property Binding: <img [src]='product.imageUrl'>

Event Binding: <button (click)='toggleImage()'>

Two-Way Binding: <input [(ngModel)]='listFilter'/>

Component

Exit Course

Angular: Getting Started

by Deborah Kurata

- Introduction 1m 46s
- Property Binding 4m 27s
- Handling Events with Event Binding 5m 57s
- Handling Input with Two-way Binding 4m 54s
- Transforming Data with Pipes 3m 55s
- Checklists and Summary 2m 40s

7 More on Components 34m 40s

8 Building Nested Components 25m 1s

9 Services and Dependency Injection 18m 12s

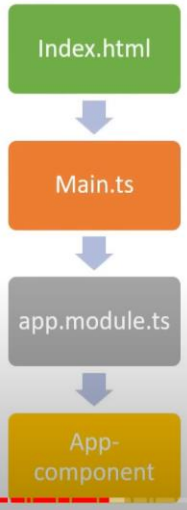
10 Retrieving Data Using HTTP 34m 23s

11 Navigation and Routing Basics 25m 57s

Autoplay

Top 50 Angular Interview Questions

How an Angular App gets Loaded and Started? What are index.html, app-root, selector and main.ts?



- Angular is used to create Single Page Applications. index.html file is that single page. Index.html will invoke main.js file which is the javascript version of main.ts file.
- main.ts file is like the entry point of web-app. It compiles the web-app and bootstraps the AppModule to run in the browser.
- App module file will then bootstrap the Appcomponent.
- App-component or app-root component is the html which you will see finally.

23:00 / 2:12:58 • How an Angular App gets Loaded and Started? What are index.html, app-roo

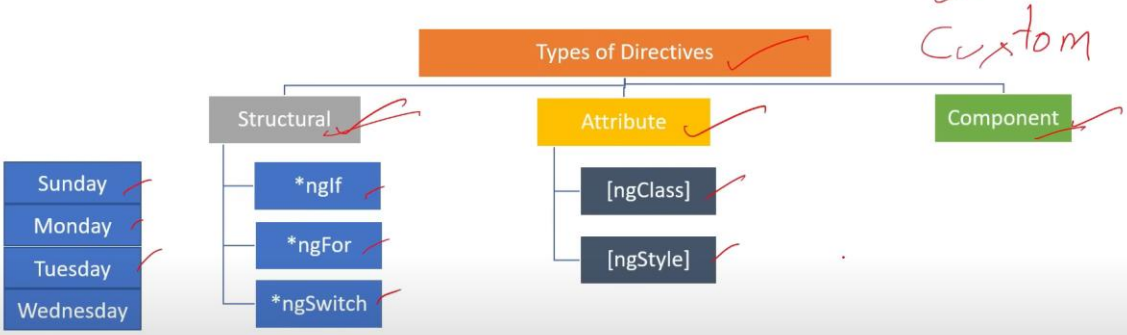
Top 50 Angular Interview Questions

What are Directives? What are the type of directives?

- Directives are classes that add additional behavior to elements in your Angular applications.

Submit

Custom



Structural directives change appearance of DOM by adding or removing elements.

Attribute directives change the appearance or behavior of an element.

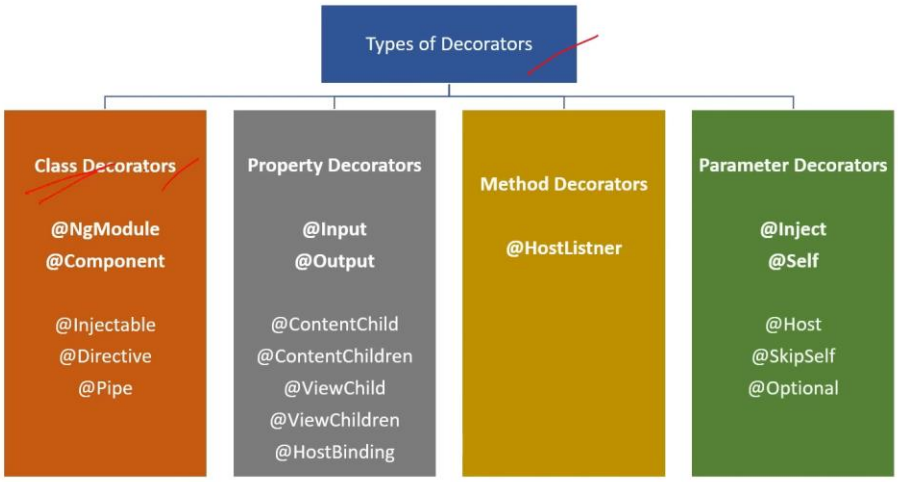
Component directive are directives with its own template.

44:38 / 2:12:58 • What are Directives? What are the type of directives? >

Q What is the difference between Component, Attribute and Structural Directives?

Component Directive	Structural Directive	Attribute Directive
1. Component directive is responsible for showing the first whole view. It is the most used one.	Structural directive is responsible for adding and deleting html elements in the view.	Attribute directive is responsible for changing the appearance of view by adding or removing styles/cssclasses of the html elements.
2. Starts with @ sign. Like: @Component	Starts with * sign. Like: *ngIf, *ngFor, *ngSwitch	Set inside square brackets. Like: [ngClass], [ngStyle]

Q What are the types of Decorator?



Q What is Provider in Angular?

- A provider is an object declared inside decorators which inform Angular that a particular service is available for injecting inside the components.

```
@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    MenuComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [LoggingService],
  bootstrap: [AppComponent, LoginCo
})
export class AppModule { }
```

```
@Injectable({
  providedIn: 'root'
})
export class LoggingService {

  constructor() { }

  LogError()
  {
    console.log("Error Logged1");
    console.log("Error Logged2");
    console.log("Error Logged3");
  }
}
```

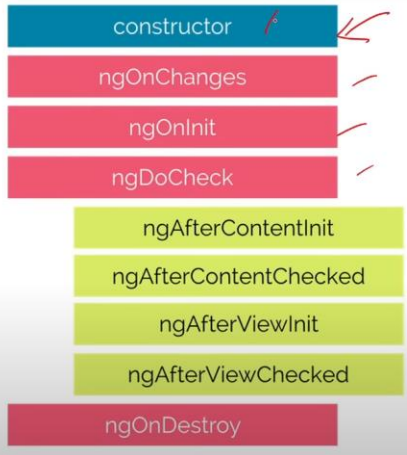
```
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css'],
  providers: [LoggingService]
})
export class LoginComponent implements OnInit {

  constructor(private loggingService: LoggingServi

    this.loggingService.LogError();
    //const ls = new LoggingService();
    //ls.LogError();
    // console.log("Error Logged1");
    // console.log("Error Logged2");
    // console.log("Error Logged3");
  }
```

Top 50 Angular Interview Questions

What are Lifecycle Hooks in Angular? **V. IMP.**

- A component from creation to destruction goes through several stages and these stages are the life cycle hooks.

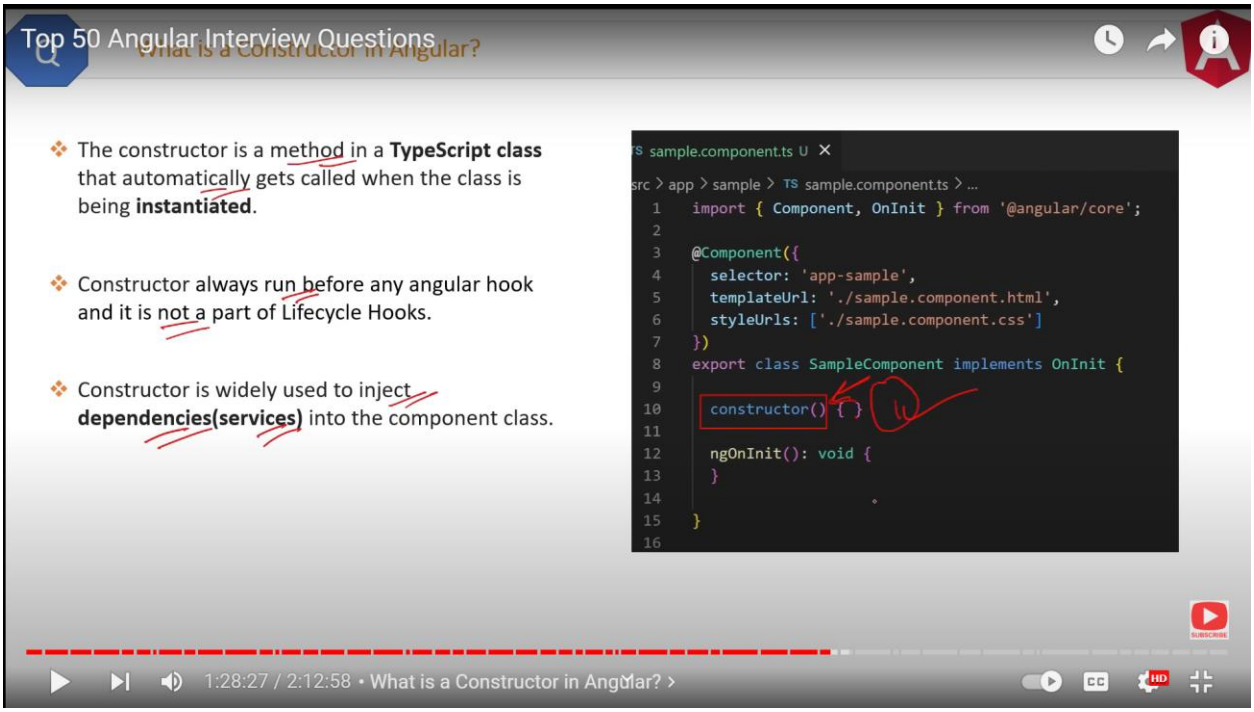
- The stages will cover activities like:
  - Component instantiating.
  - Rendering the component html view.
  - Creating the child components(if required).
  - Destroying the component.



1:24:54 / 2:12:58 • What are Lifecycle Hooks in Angular? >

Q What are Lifecycle Hooks in Angular? **V. IMP.**

constructor	It is a default method of the typescript class that is executed when the class is instantiated. It always run before any angular hook and it is not a part of Lifecycle Hooks.
ngOnChanges	Called when input property changes. @Input
ngOnInit	Called when the component is created.
ngDoCheck	
ngAfterContentInit	Called for component children.
ngAfterContentChecked	
ngAfterViewInit	
ngAfterViewChecked	
ngOnDestroy	Called when the component is destroyed.

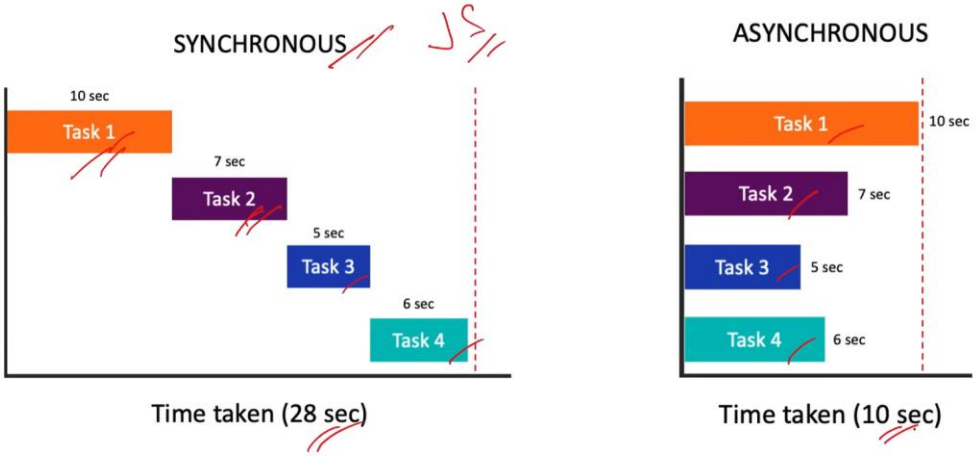


Q What is the difference between Constructor and ngOnInit? **V. IMP.**

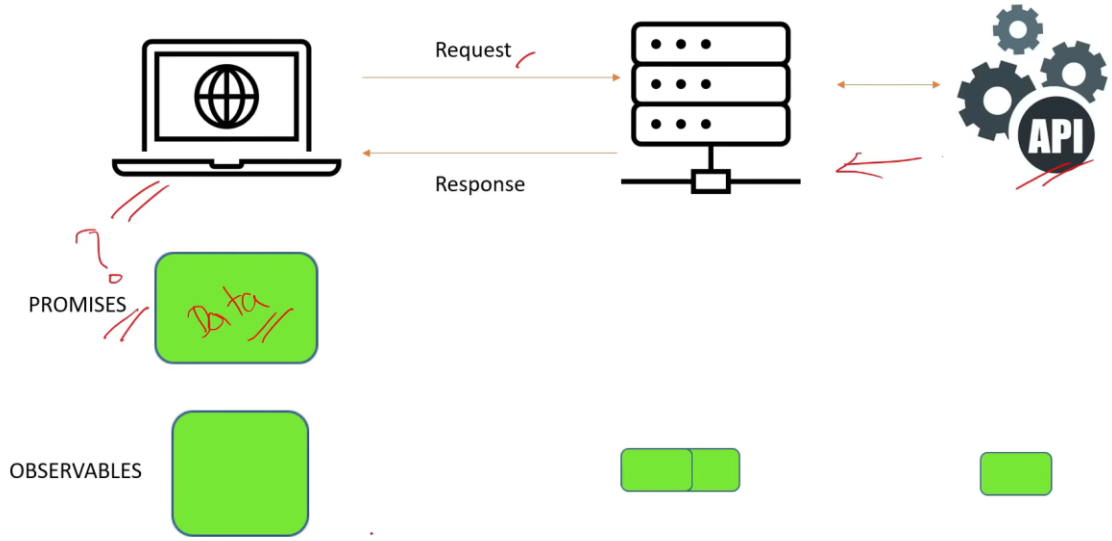
ngOnInit	Constructor
1. NgOnInit is an **Angular lifecycle hook**, which signals the activation of the created component.	The constructor is a method in a **TypeScript class,** that automatically gets called when the class is being instantiated.
2. ngOnInit is called **after** ngOnChanges lifecycle-hook.	Constructor is called **before** any lifecycle-hook.
3. When ngOnInit called, everything about component is already ready, so it's use is to perform most of the **business logic** on component.	When constructor called, everything in component is not ready, so it's mostly used for **injecting dependencies** only**.**

Q What are Asynchronous operations?

- Observables are used to perform asynchronous operations and handle asynchronous data



Q What is the difference between Promise and Observable? **V. IMP.**



Q What is the difference between Promise and Observable?

Observables	Promises
1. Emit **multiple** values over a period of time. Also called streaming of data.	Emit a **single** value at a time.
2. Are lazy: they're not executed until we **subscribe** to them using the subscribe() method. memory	Are not lazy: execute **immediately** after creation.
3. Have subscriptions that are **cancellable** using the unsubscribe() method.	Are **not cancellable**.

Q What is RxJS?

RxJS - Introduction

rxjs.dev/guide/overview

RxJS OVERVIEW REFERENCE TEAM

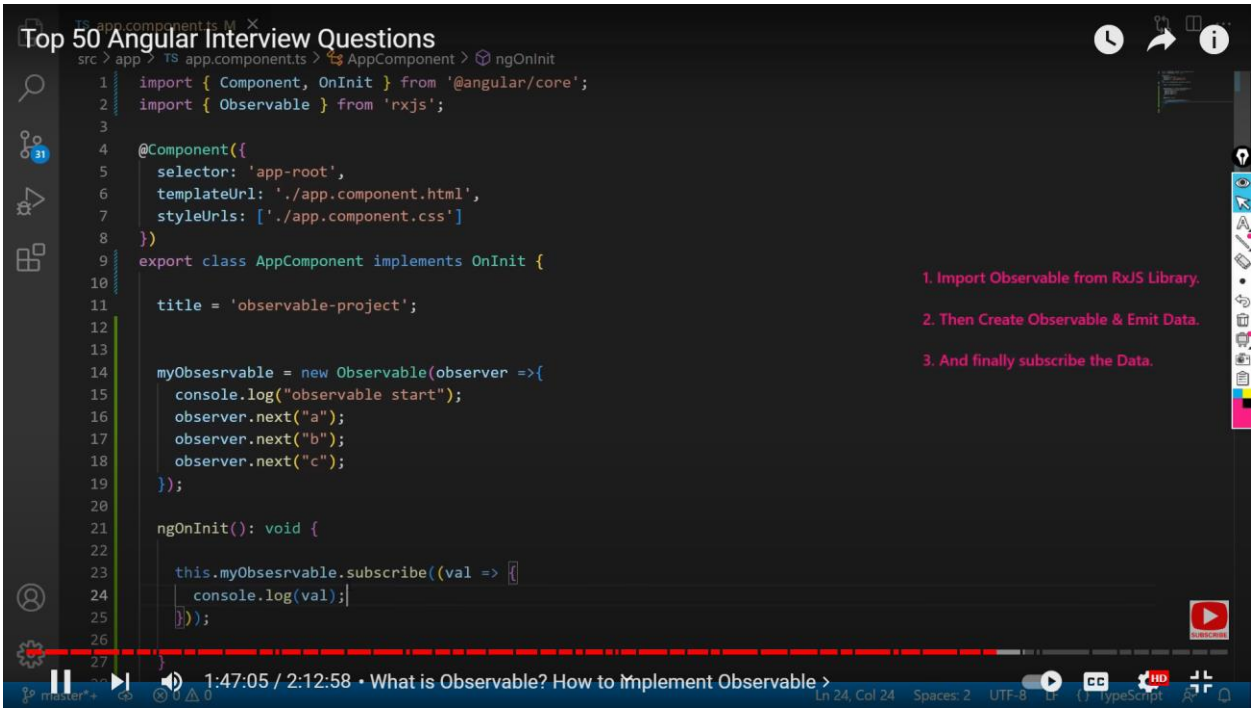
Introduction

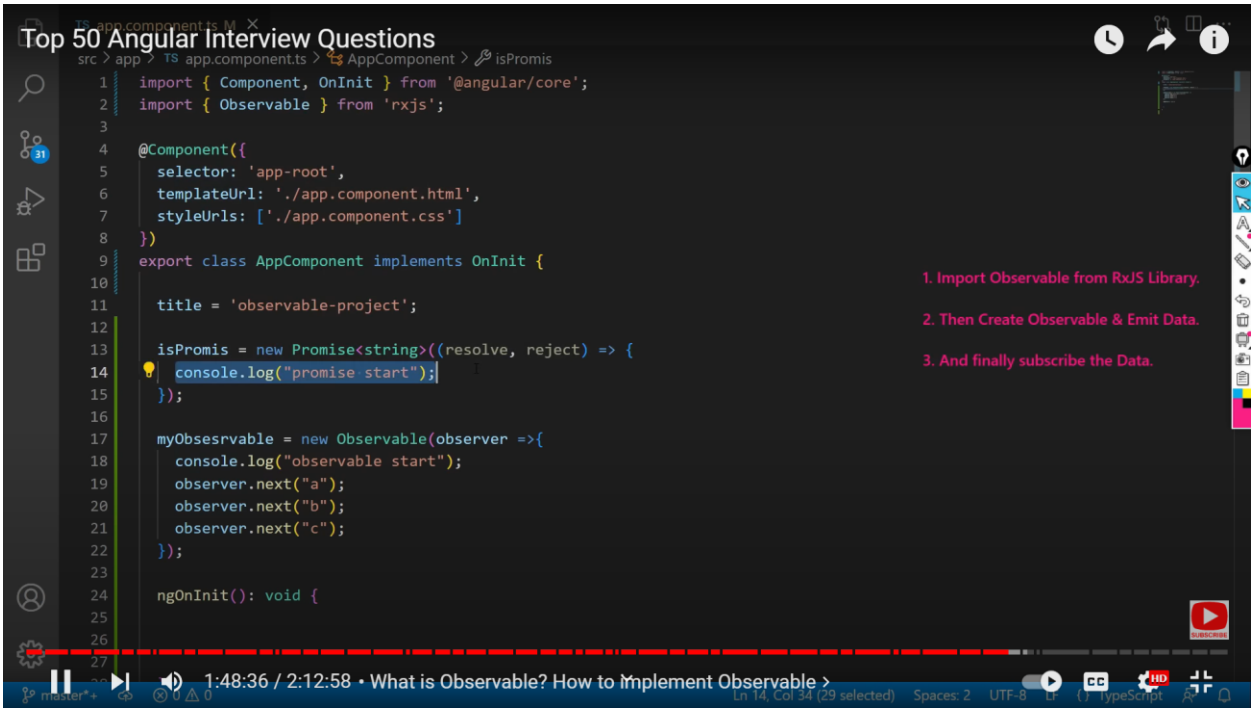
RxJS is a library for composing asynchronous and event-based programs by using observable sequences. It provides one core type, the Observable, satellite types (Observer, Schedulers, Subjects) and operators inspired by `Array` methods (`map`, `filter`, `reduce`, `every`, etc) to allow handling asynchronous events as collections.

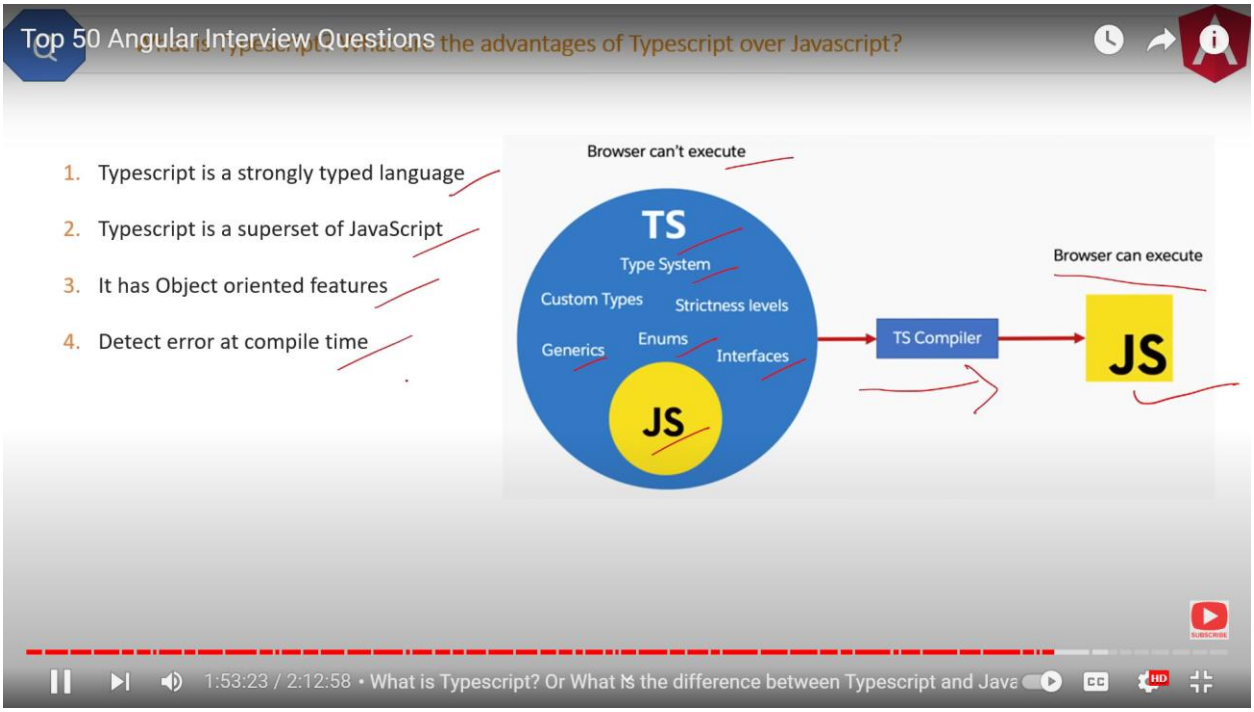
- RxJS is a javascript library that allow us to work with asynchronous data stream with the help of observables.
- Observables introduced by **RxJS** library. JS, client framework
- RxJS stands for Reactive Extensions for JavaScript.

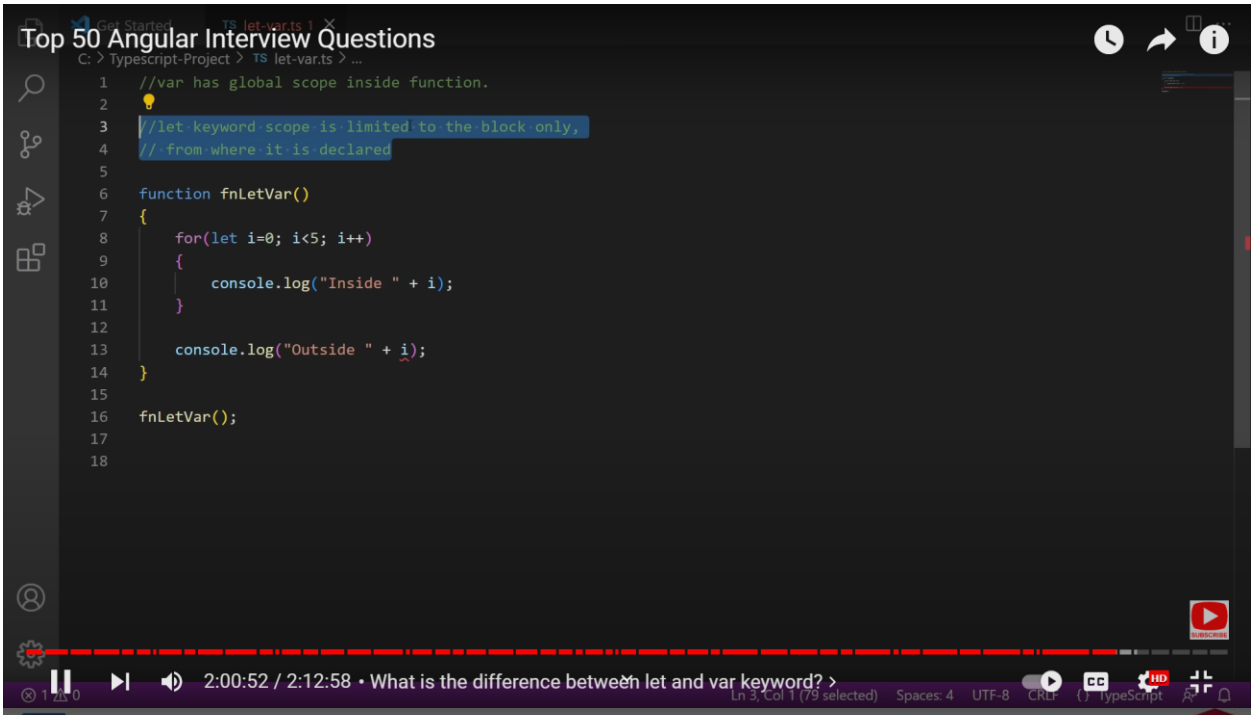
Stream of Data

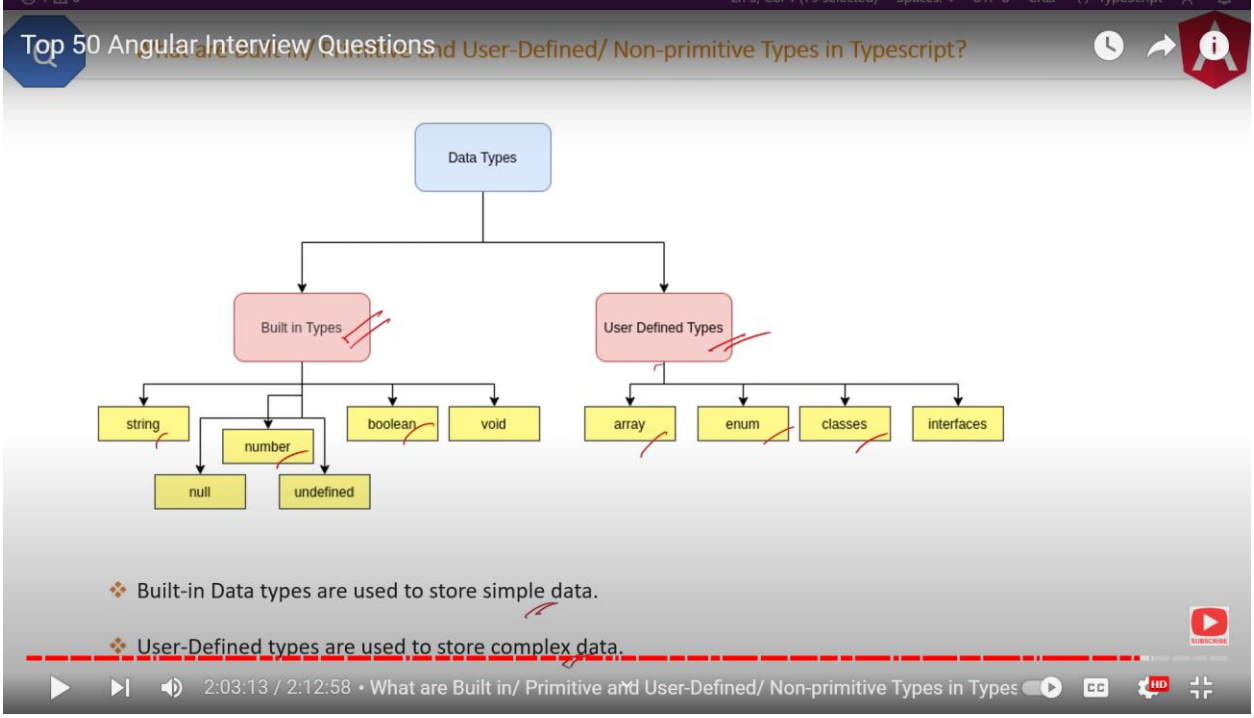
Subscriber




Top 50 Angular Interview Questions
src > app > app.component.ts > AppComponent > isPromis
import { Component, OnInit } from '@angular/core';
import { Observable } from 'rxjs';
@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
title = 'observable-project';
isPromis = new Promise<string>((resolve, reject) => {
console.log("promise start");
});
myObsesrvable = new Observable(observer =>{
console.log("observable start");
observer.next("a");
observer.next("b");
observer.next("c");
});
ngOnInit(): void {
1. Import Observable from RxJS Library.
2. Then Create Observable & Emit Data.
3. And finally subscribe the Data.
1:48:36 / 2:12:58 • What is Observable? How to Implement Observable ›


Top 50 Angular Interview Questions
1. Typescript is a strongly typed language
2. Typescript is a superset of JavaScript
3. It has Object oriented features
4. Detect error at compile time
Browser can't execute
TS
Type System
Custom Types
Strictness levels
Generics
Enums
Interfaces
JS
TS Compiler
Browser can execute
JS
1:53:23 / 2:12:58 • What is Typescript? Or What is the difference between Typescript and Java


Top 50 Angular Interview Questions
C: > Typescript-Project > let-var.ts > ...
//var has global scope inside function.
//let keyword scope is limited to the block only,
// from where it is declared
function fnLetVar()
{
for(let i=0; i<5; i++)
{
console.log("Inside " + i);
}
console.log("Outside " + i);
}
fnLetVar();
2:00:52 / 2:12:58 • What is the difference between let and var keyword? ›


Top 50 Angular Interview Questions
Data Types
Built in Types
User Defined Types
string
number
boolean
void
null
undefined
array
enum
classes
interfaces
Built-in Data types are used to store simple data.
User-Defined types are used to store complex data.
2:03:13 / 2:12:58 • What are Built in/ Primitive and User-Defined/ Non-primitive Types in Types

Get Started | AnyType.ts

C: > Typescript-Project > AnyType.ts > ...

```typescript

//Definition - When a value is of type any,
//then no typechecking will be done by compiler and
//the flexibility is there to do anything with the variable.

let i: any;

i = 1;

i = "Happy";

i = true;

i();

i.AnyProperty = 10;
```

0 0 | Ln 2, Col 4 | Spaces: 4 | UTF-8 | CRLF | {} TypeScript

Top 50 Angular Interview Questions

C: > Typescript-Project > types.ts > ...

```typescript
//Basic/ Primitive Types in Typescript

//Boolean
let isCompleted: boolean = false;

//Number
let count: number = 100;

//String
let myname: string = "Happy";

//Array
let list: number[] = [1, 2, 3];


//Tuple
// Declare a tuple type
let x: [string, number];

// Initialize it
x = ["Happy", 10]; // OK
```

2:03:56 / 2:12:58 • What are Built in/ Primitive and User-Defined/ Non-primitive Types in Types

Get Started | Enum.ts

C: > Typescript-Project > Enum.ts > ...

```typescript
//Defintion - - Enums allows to define a set of named constants.

const DirectionNorth = 0;
const DirectionSouth = 1;
const DirectionWest = 2;
const DirectionEast = 3;

let dir = DirectionEast;

enum Direction {
    North = 1,
    South,
    West,
    East
  }

let dirNew = Direction.East;
```

0 0 | Ln 2, Col 1 (63 selected) | Spaces: 2 | UTF-8 | CRLF | {} TypeScript

Get Started | TypeAssertion.ts

C: > Typescript-Project > TypeAssertion.ts > ...

```typescript
//Definition: Type assertion is a technique
//that informs the compiler about the type of a variable.

let myname;

myname = "Happy";

let isEndCharY = (<string>myname).endsWith("y");
```

0 0 | Ln 3, Col 58 | Spaces: 4 | UTF-8 | CRLF | {} TypeScript

Get Started | ArrowFunctions.ts

C: > Typescript-Project > ArrowFunctions.ts > ...

```typescript

//Arrow functions are the shorthand syntax
//for defining the anonymous function

let x = function(a, b)
{
    a * b;
}

let y = (a, b) =>  a * b;
```

0 0 | Ln 10, Col 26 | Spaces: 4 | UTF-8 | CRLF | {} TypeScript