Angular TS Enhanced Web-Apps

Course Outline

- The Single Page Applications
- Introduction to Typescript and ES6
- Angular Introduction
- Components & Directives
- Form, Data and Event Binding
- Services DI & HTTP Client
- Pipes and Data Formatting
- Routing

Target Audience

- **Must have on hands experience with**
  - HTML5
  - JavaScript
  - CSS3
  - Concept of Http/Web-Server
  - Rest Services

Single Page Applications

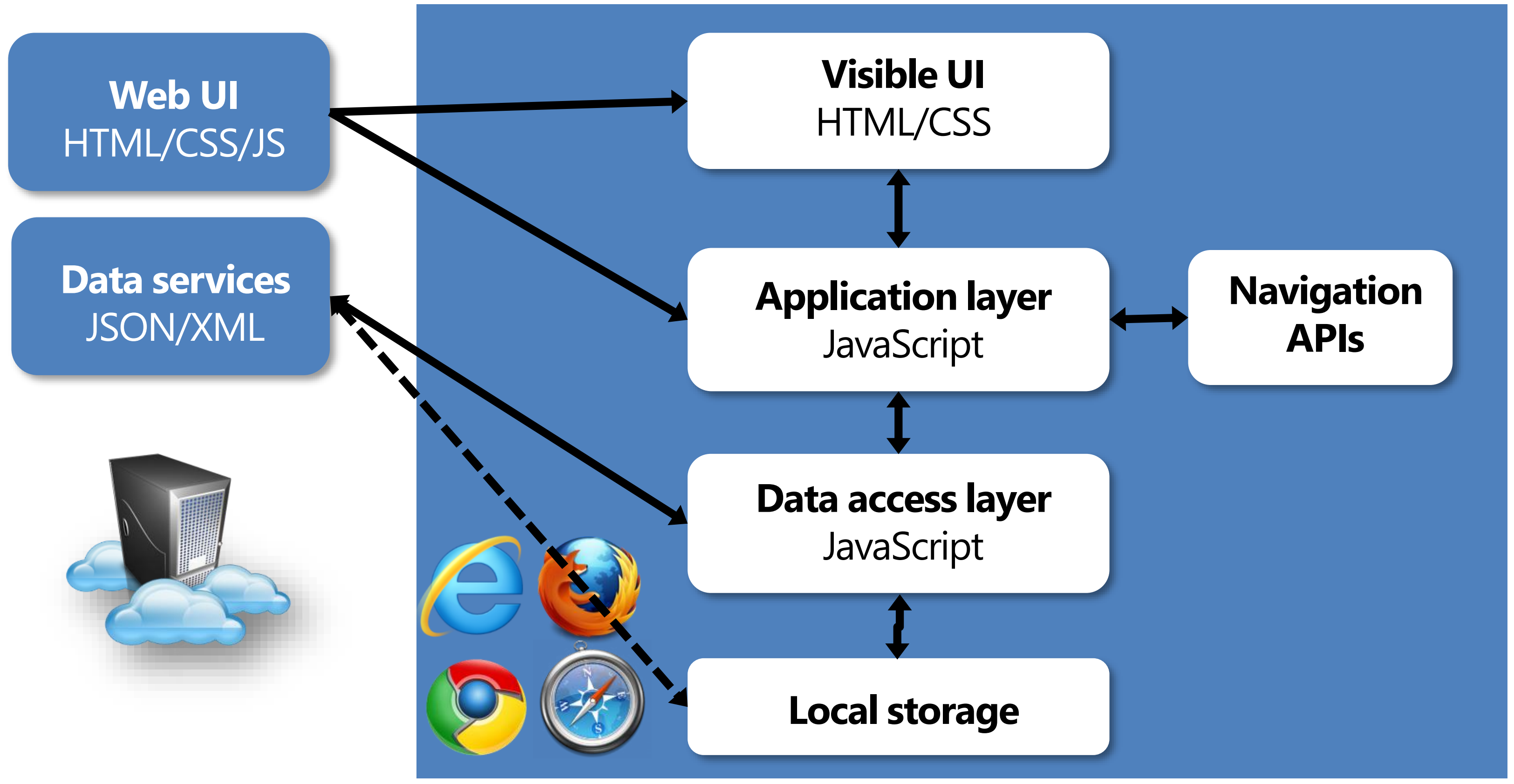
Single Page Application ?

- **What are single page application (SPA's) ?**
  - Rich
  - Responsive
  - Desktop Like
  - Built With HTML5
  - Non Fancy Simple Technology
  - Easy to Develop
- **Why Should I care ?**
  - Full page loading applications are slow and over the network
  - Need to rethink approaches that gives you fast responsive and modern approach

Few Benefits of SPA’s

- SPA’s allow different views to be loaded into a shell page on user interaction
- SPA’s maintain history of views that have been loaded.
- Enhanced User experience
- Runs on any device and form-factor
- Can work offline
- Easier to test
- Deployable on app-stores
- Mobile & Device ready

Architecture



Challenges

- **SPA' Need's you to know many technologies**
  - DOM manipulation (Jquery)
  - History (History.js)
  - Routing (Knockout.js)
  - Ajax
  - Databinding (BackBone.js)
  - CSS
  - More……. & More…..

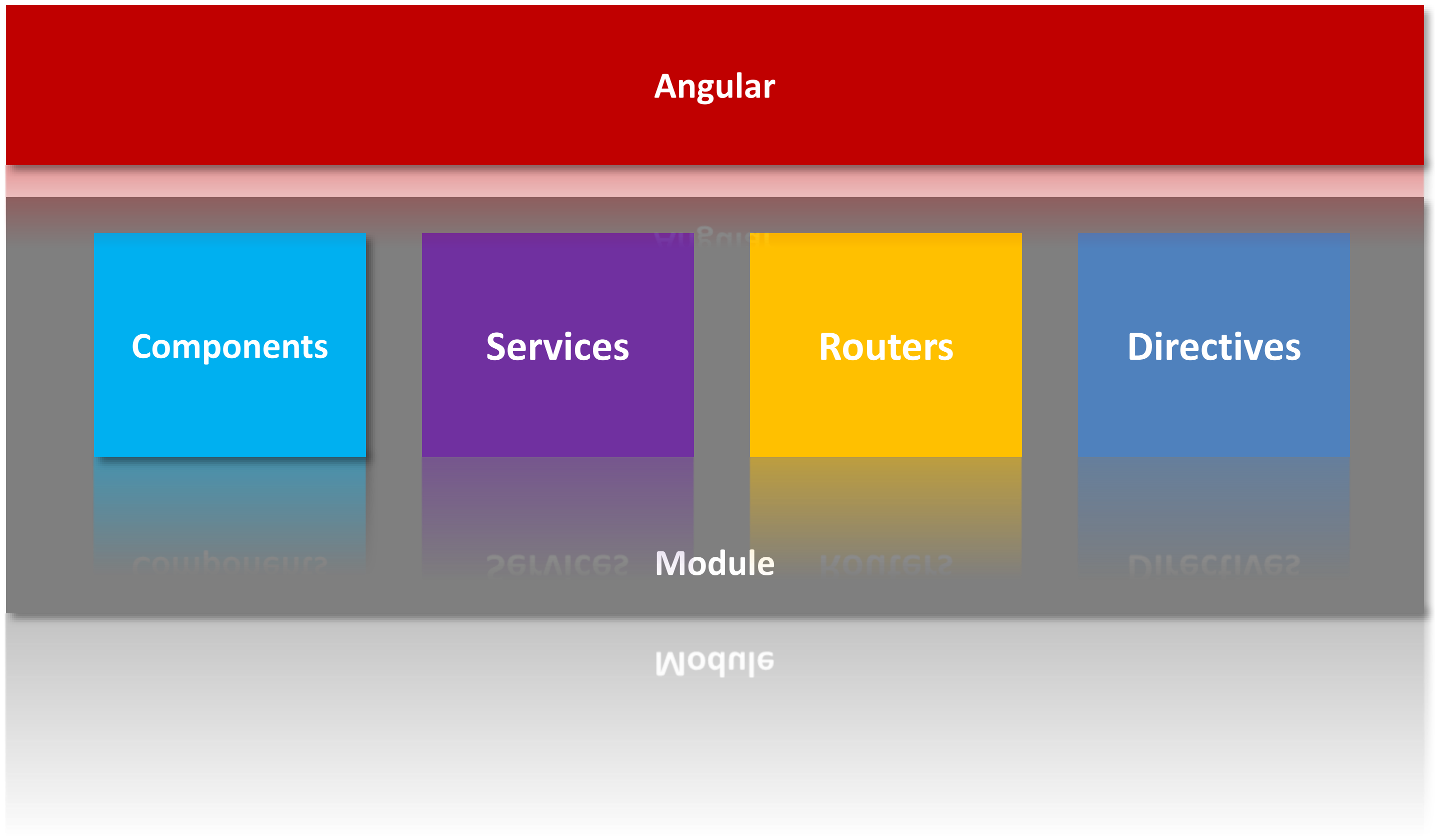
Angular

Why angular ?

- **Leading Framework in this space**
- **Backed up by google**
- **Written with best practices**
- **Google trend shows the up rise of the angular community**
- **Community Support**
- **Component driven architecture**
- **Supports TypeScript**

Architecture

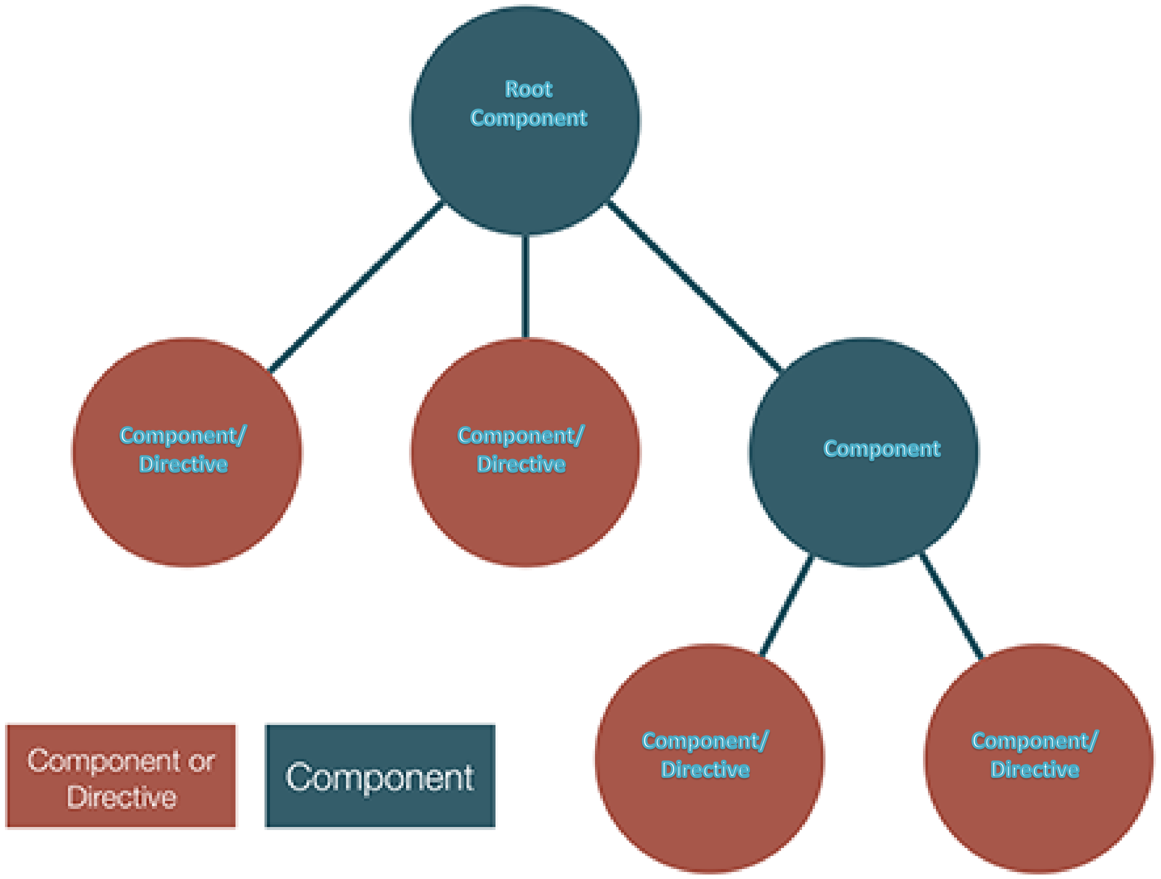
Angular Architecture



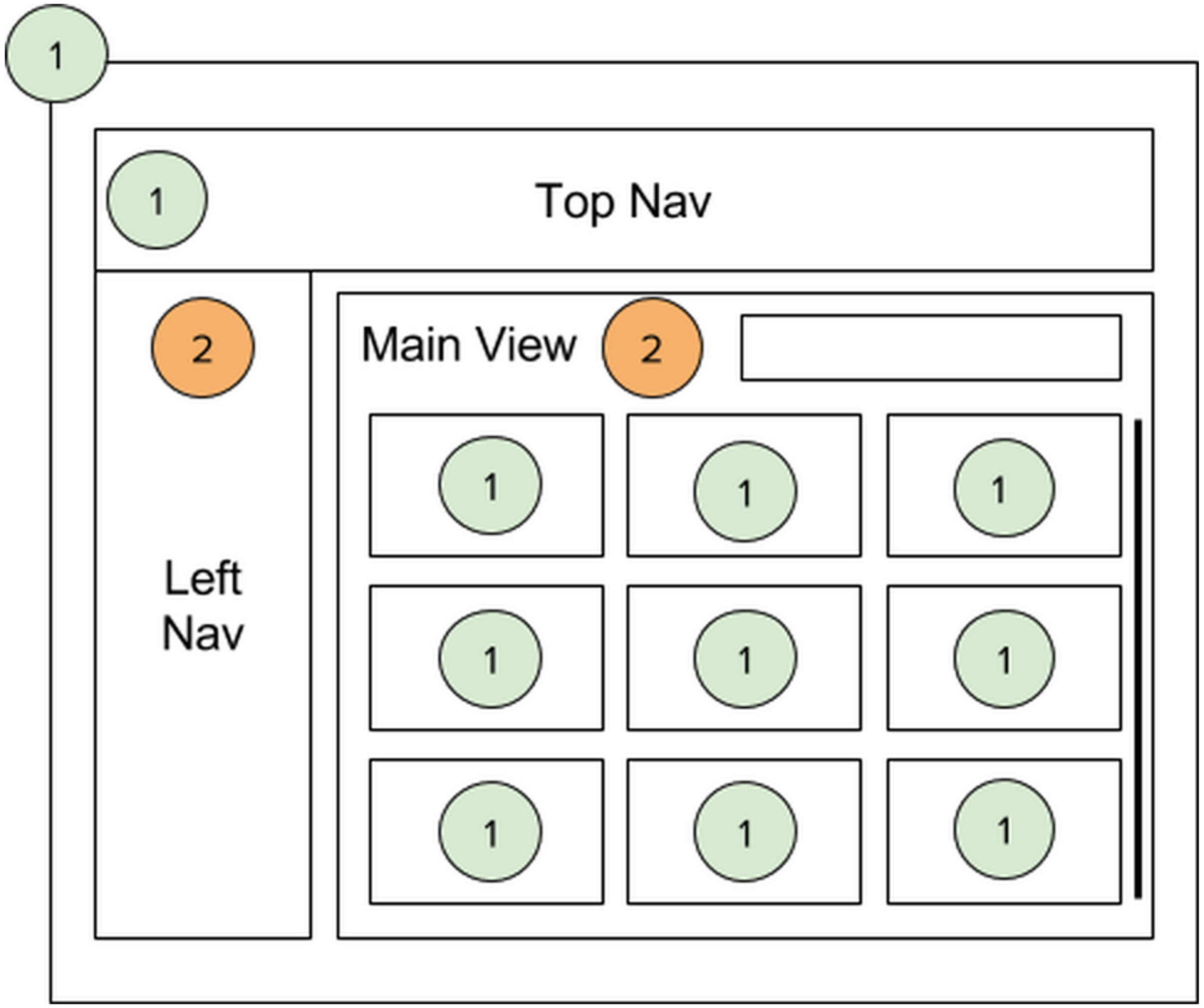
Component

- Encapsulates the Template, Data, Behavior of a view
- Known as view component
- Every angular app has atleast one component or the root component
- A component can contain other components
- Plain Classes written in typescript
- Binds view to the properties , events and

Component View



Component Based View

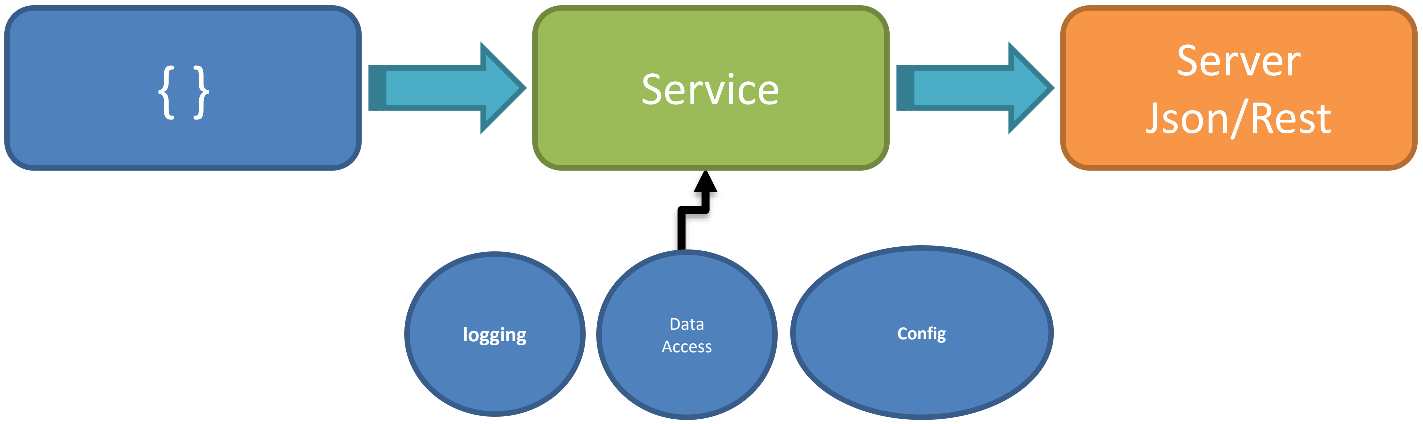


Benefits

- Break up application in to manageable components
- Role Separation
- Re-usability
- Decoupled
- Test-Driven or unit testable
- Object-oriented
- Freedom from DOM based programming

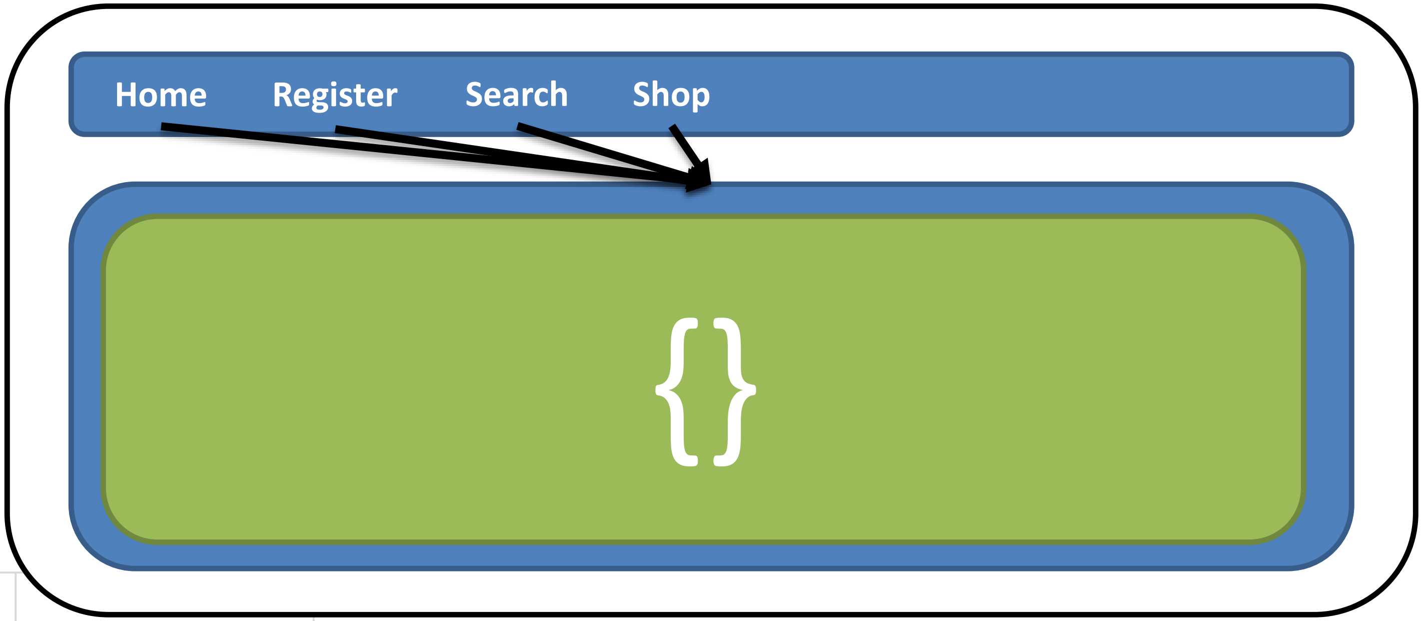
Services

- Components need to talk to backend api
- Services are singleton objects
- Instantiated once and can be reused multiple times
- Common reusable code can reside in the services
- Services can be used to make rest calls / jsonp calls / load data from server


{ }
Service
Server
Json/Rest
logging
Data
Access
Config

Router

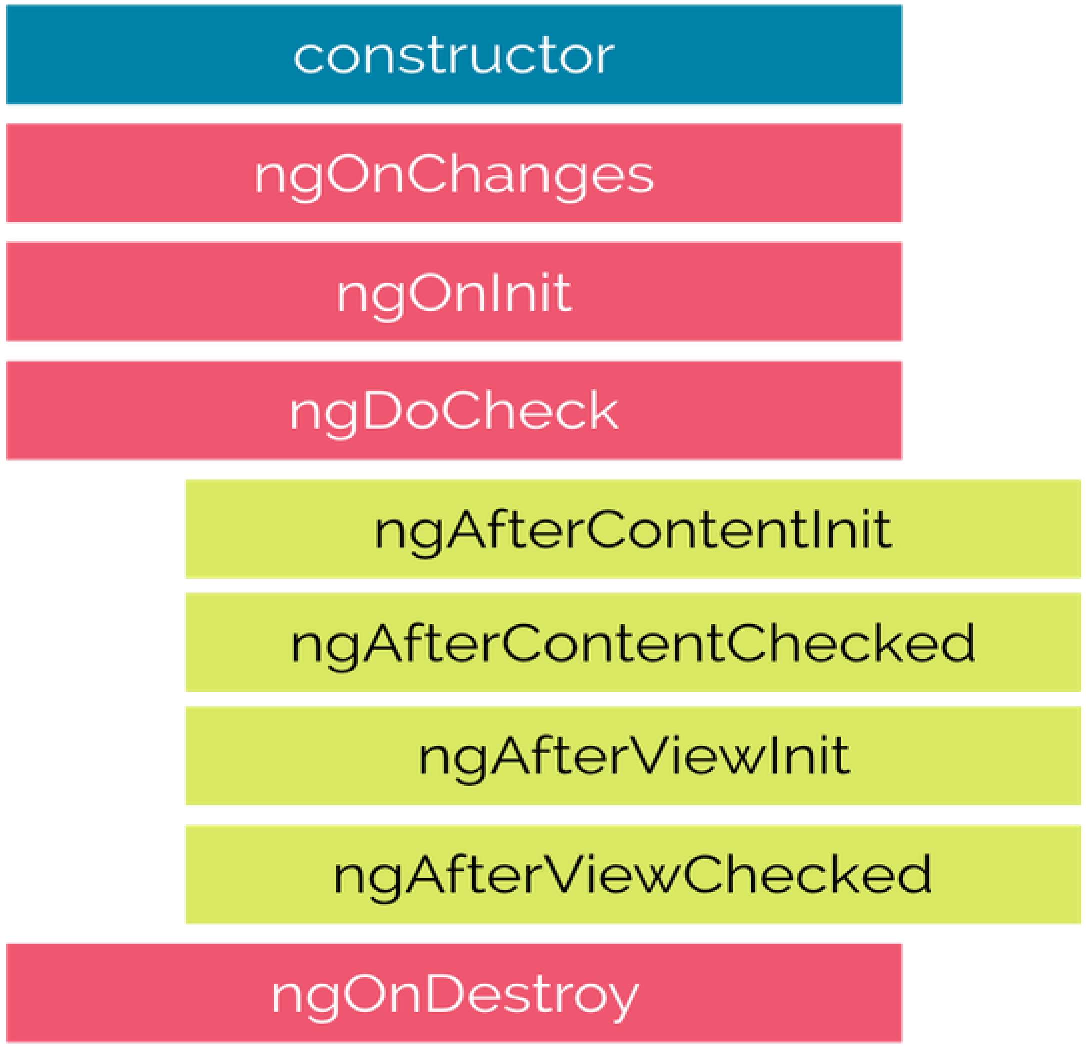
- Responsible for Navigation
- Responsible for history Management
- Responsible for displaying components on specific link being clicked



Directives

- **Structural Directives**
  - It alters the layout of the DOM by adding, replacing and removing its elements.
- **Attribute directives**
  - It changes the appearance or behavior of a DOM element. These directives look like regular HTML attributes in templates.
- **Component Directive**
  - It is a *directive-with-a-template* and the *@Component* decorator which is indeed a *@Directive* decorator wherein the template-oriented features is extended

Angular Lifecycle



Hooks for the Component

- **constructor**

This is invoked when Angular creates a component or directive by calling new on the class.

- **ngOnChanges**

Invoked every time there is a change in one of th input properties of the component.

- **ngOnInit**

Invoked when given component has been initialized.

This hook is only called once after the first ngOnChanges

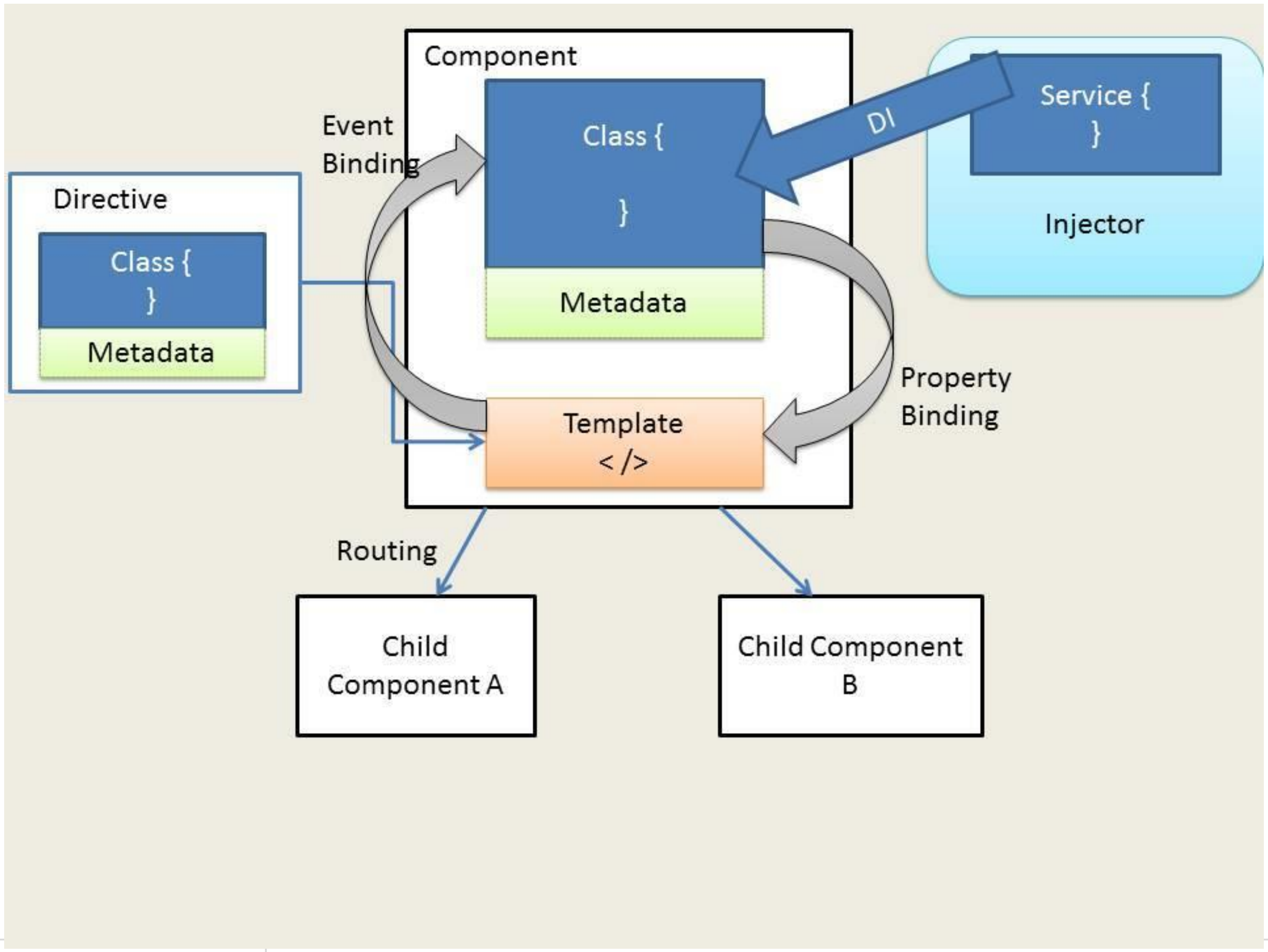
- **ngDoCheck**

Invoked when the change detector of the given component is invoked. It allows us to implement our own change detection algorithm for the given component.

- **ngOnDestroy**

This method will be invoked just before Angular destroys the component.

- **Important**
- **ngDoCheck and ngOnChanges should not be implemented together on the same component.**


Component
Event Binding
Class {
}
Metadata
DI
Service {
}
Injector
Directive
Class {
}
Metadata
Property Binding
Template
< />
Routing
Child Component A
Child Component B

Introduction to TypeScript

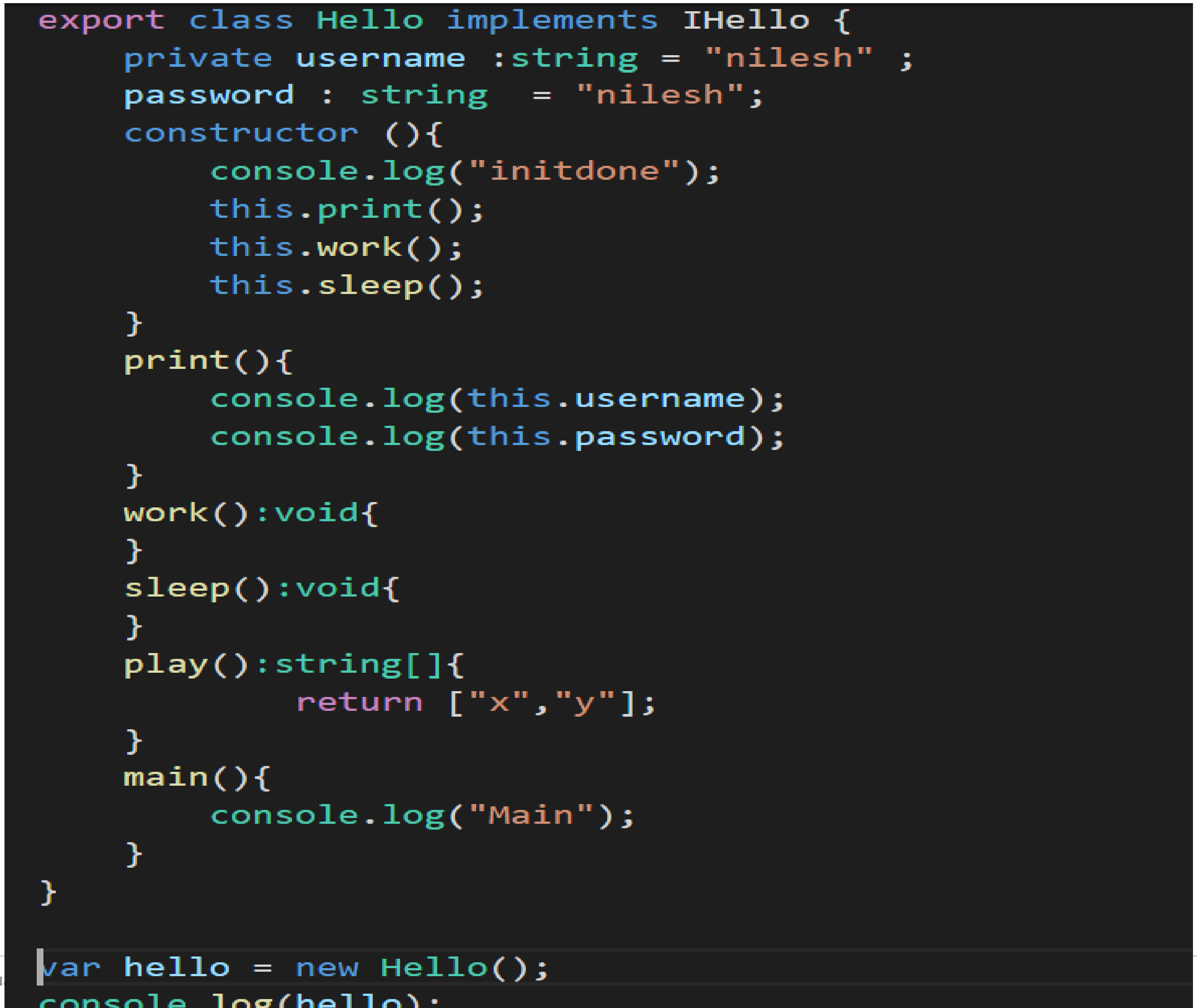
TypeScript

- **Super Set of Javascript**
- **More Preditable**
- **Provided compile time validation**
- **Data Type Support**
- **Modular**
- **Object oriented**
- **Converts to JavaScript**
- **Angular 2 is written using typescript**

Installing TypeScript

- **Type Script Is available as node module**
- **Install using**
  - ***npm install –g typescript***
- **Provides tsc (TypeScript Compiler)**
  - tsc <yourtsfile>
  - tsc <yourtsfile> --watch

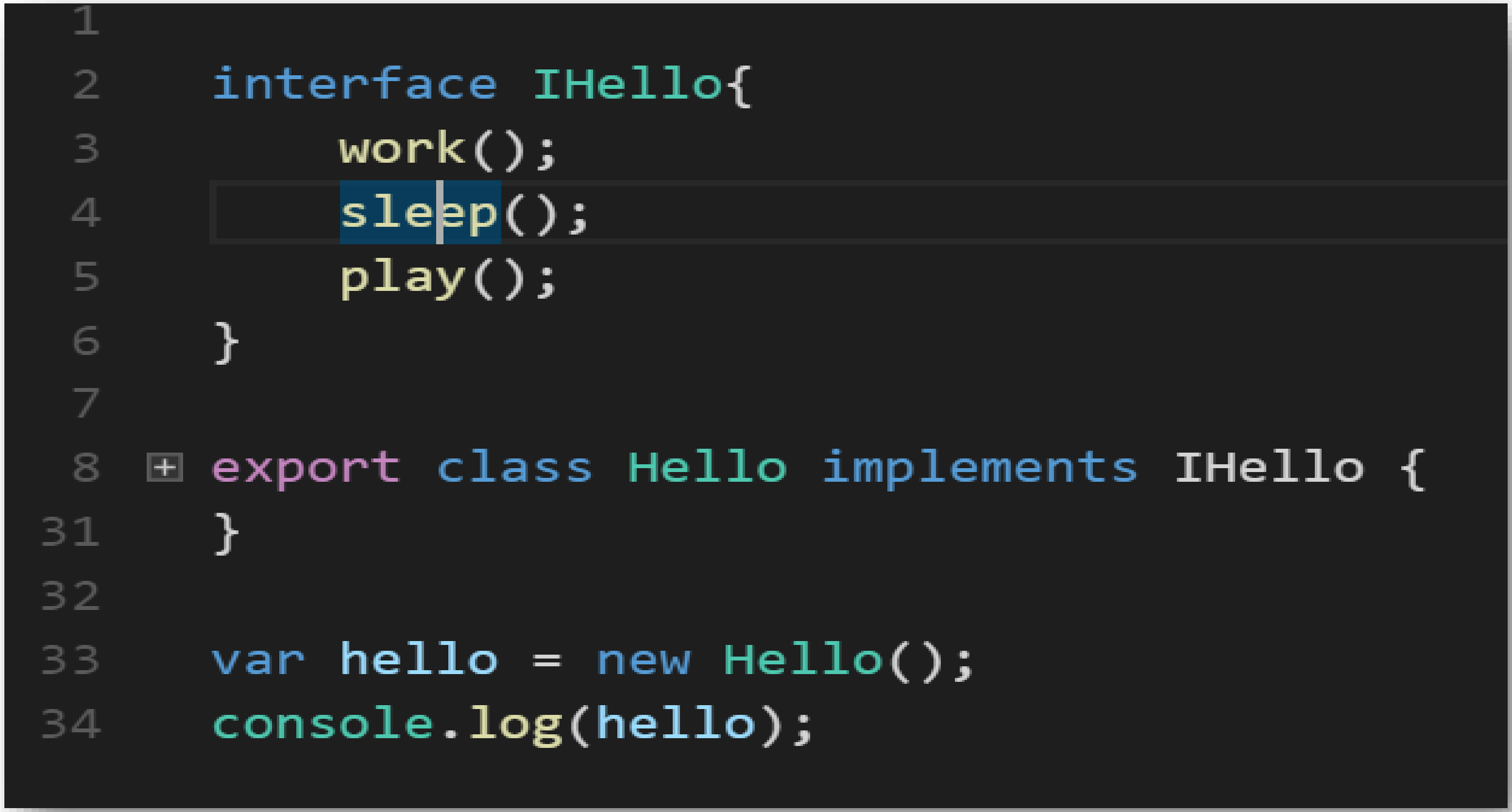
Sample typescript file



```
export class Hello implements IHello {
    private username :string = "nilesh" ;
    password : string  = "nilesh";
    constructor (){
        console.log("initdone");
        this.print();
        this.work();
        this.sleep();
    }
    print(){
        console.log(this.username);
        console.log(this.password);
    }
    work():void{
    }
    sleep():void{
    }
    play():string[]{
            return ["x","y"];
    }
    main(){
        console.log("Main");
    }
}

var hello = new Hello();
console.log(hello);
```

Interfaces



```
interface IHello{
    work();
    sleep();
    play();
}

export class Hello implements IHello {
}

var hello = new Hello();
console.log(hello);
```