* Angular can be written in Javascript/ TypeScript/ Dart
* Type script is same as java script.. It is types that is the only difference—Like all the variables is typed.. have specific data type unlike javascript which allows var for all.
* Type script will be transpiled to javascript

Angular apps are written using

* HTML templates composed with Anularized markup
* Component classes to manage templates
* Application logic in services – like connecting to backend, rest api’s. Multiple componnets can share data from services
* Boxing components and services in modules
* Launch application by bootstrapping root module

8 main building blocks

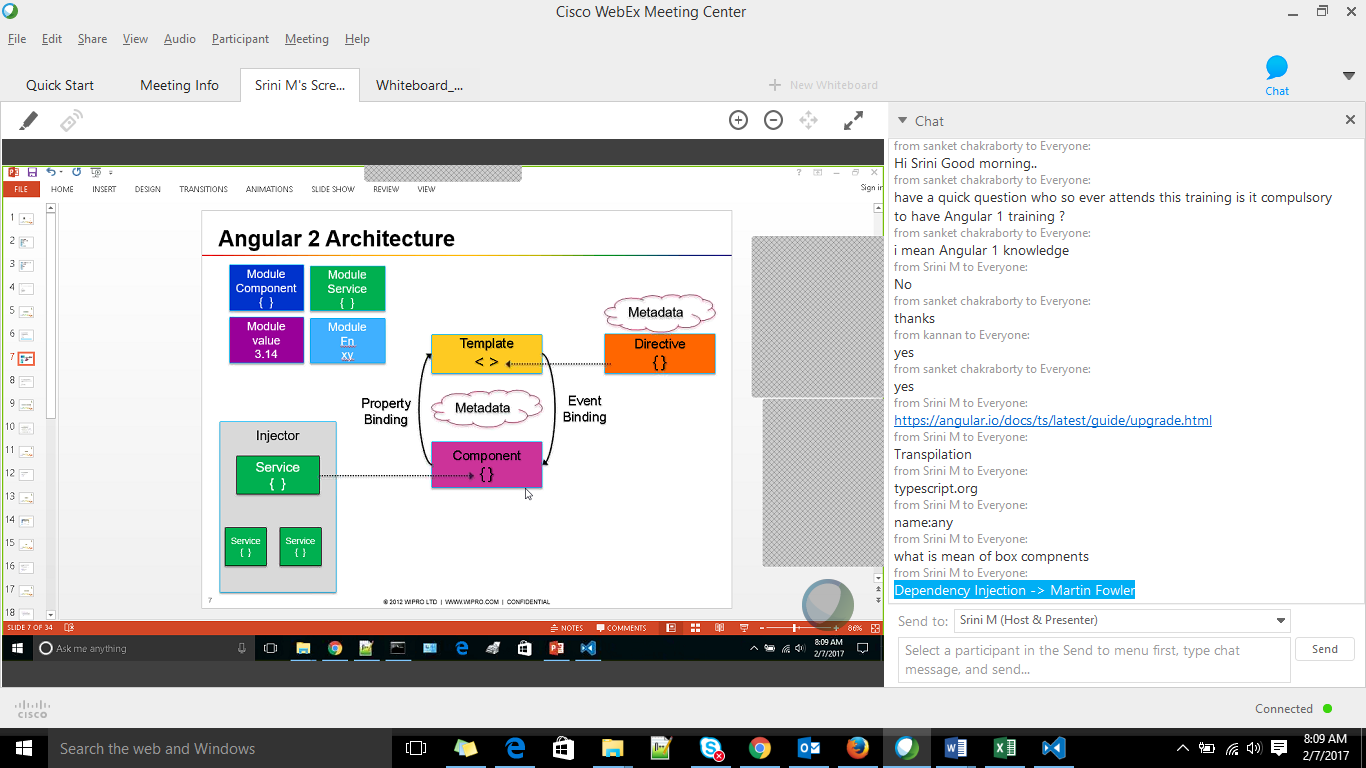
* Modules
* Components
* Templates
* Metadata – decorators like annotation
* Directives – tat which alters DOM
* Data Binding
* Services
* Dependency injection

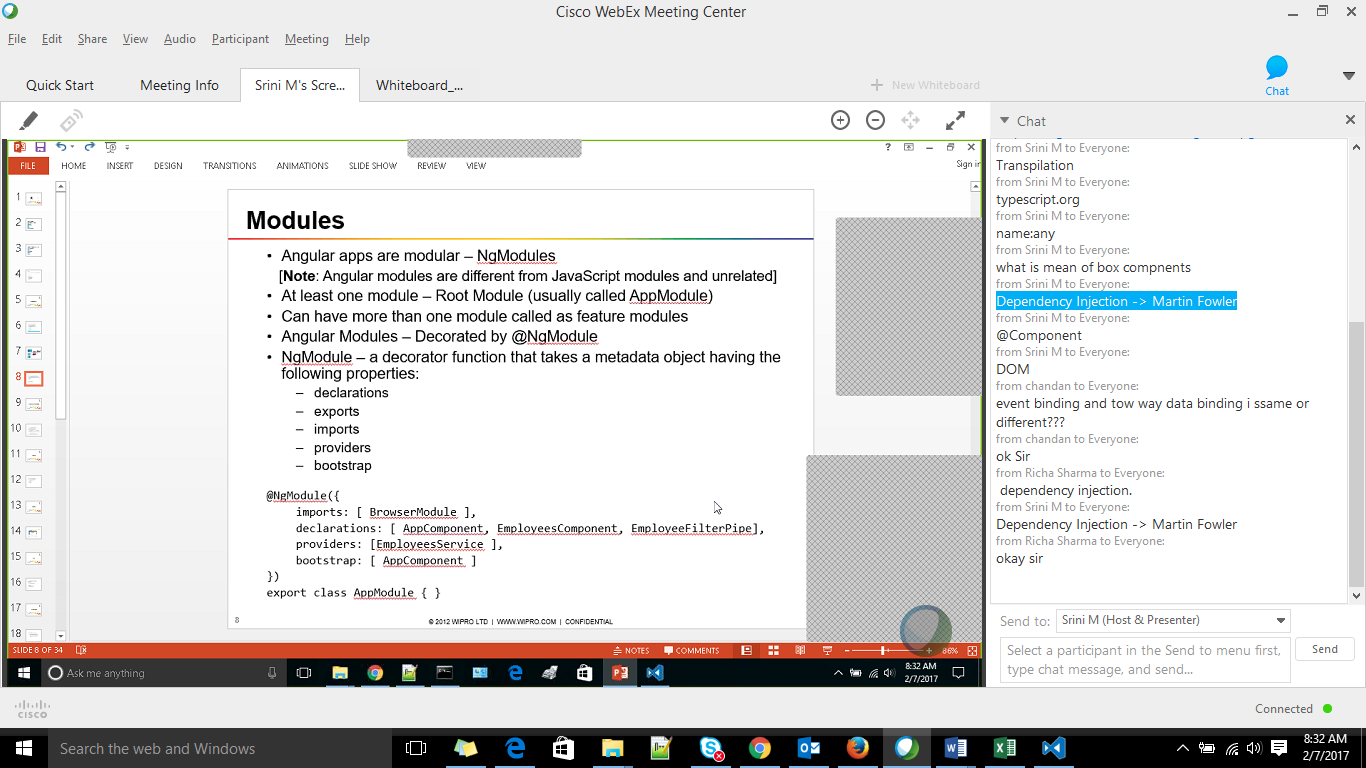
Data binding types

Property binding,Event binding, 2 way data binding

dOWNLOAD QUICKSTART-MASTER – FOR A SKELETON CODE

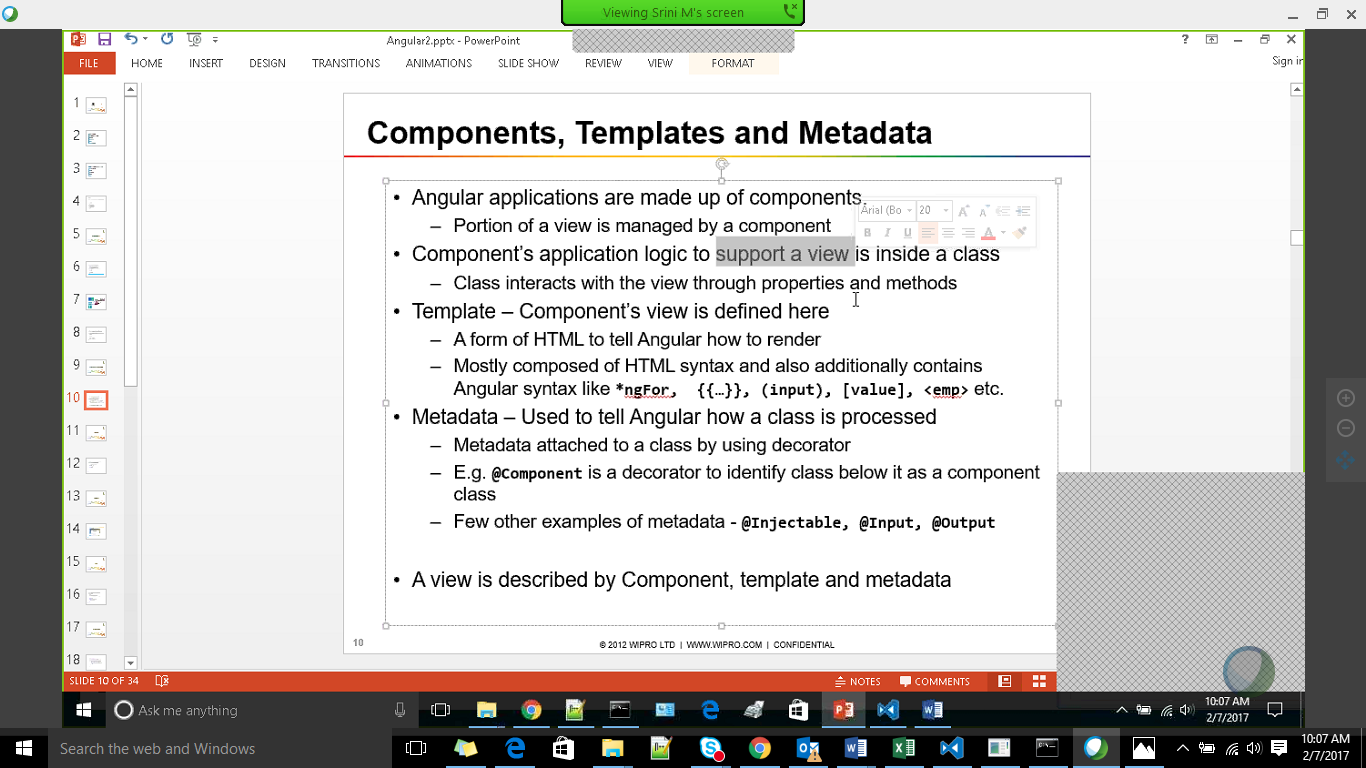
This can as well be generated using Angular CLI( command line tool)



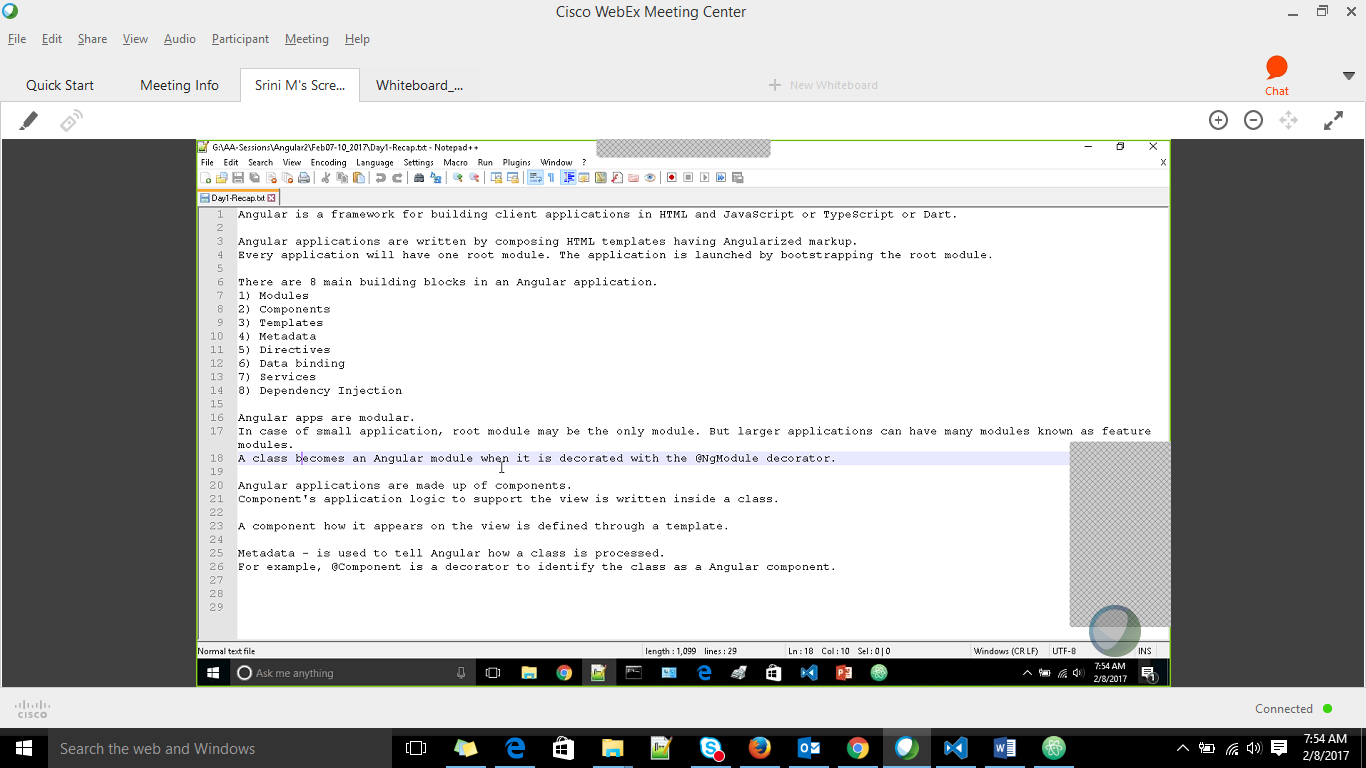


System js – similar to webpack

System\_src.js



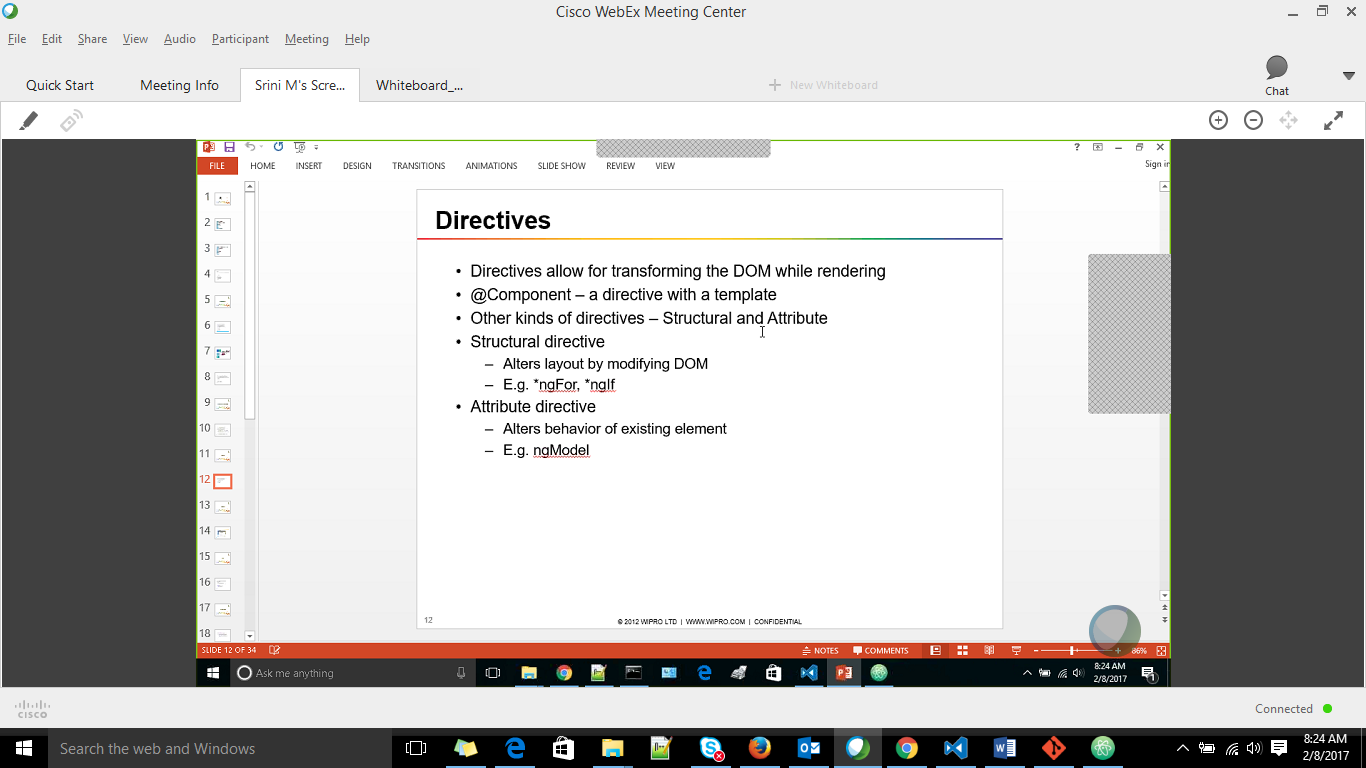
For the module system – to load all js files, tell only one starting file and rest dependent files would be loaded



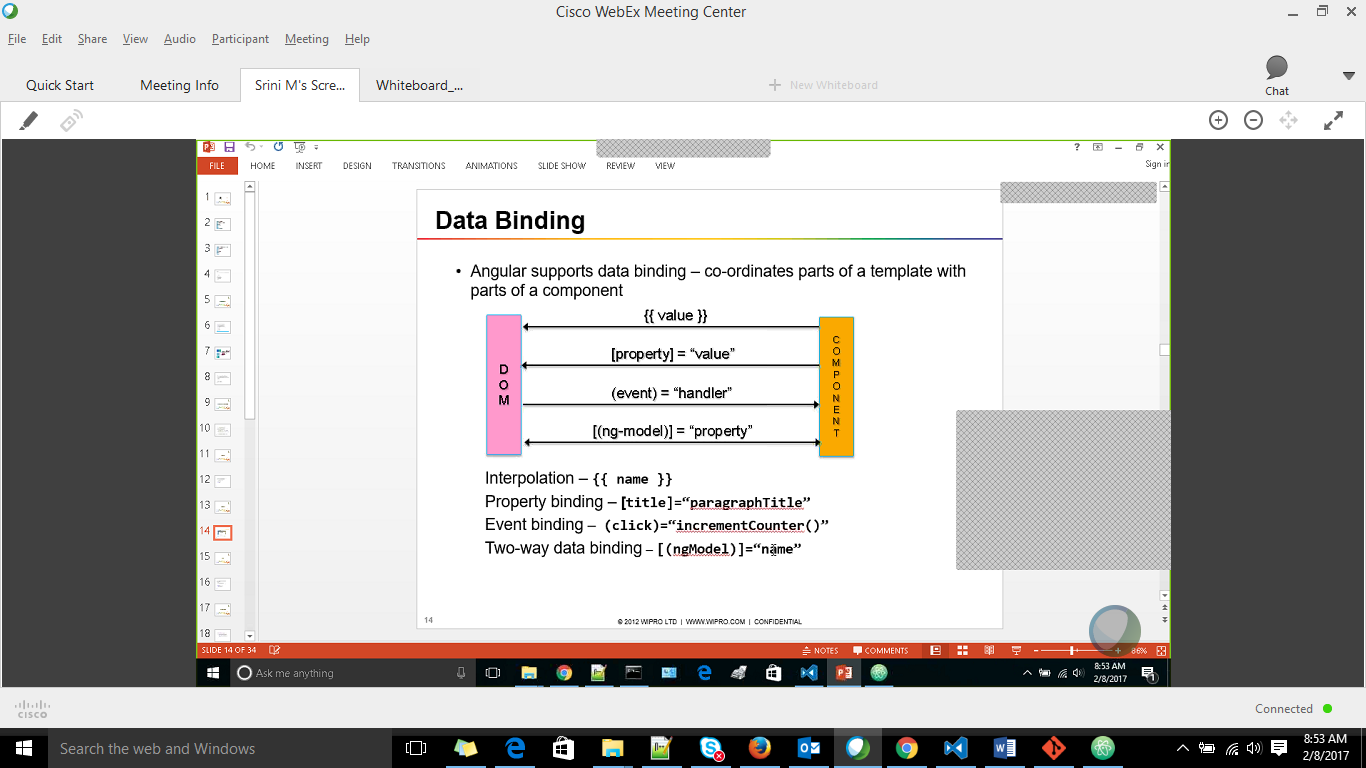
Back tick ` is used for multi line HTML in a template

Or use a templae file using template url

To run the project do ‘ npm start’ from src folder



* ngFor similar to ngRepeat in Angular1
* Structural directive alters DOM by modifying DOM
* \* Represents structural directive- use\*ngIf, \*ngFor
* @Component Is also a directive

nnnd

Data Binding

* Angular2 supports one way data binding. 2way binding should be used only if needed
* Interpolation and property binding is from component to DOM
* Event binding from DOM to component
* 2 way data binding in both property and event binding combined into one

2 way data binding is always – [(ngModel)] – this syntax- banana in a box

If any change is to be done use [ngModel] and then (ngModelChange) = setUpperCase()

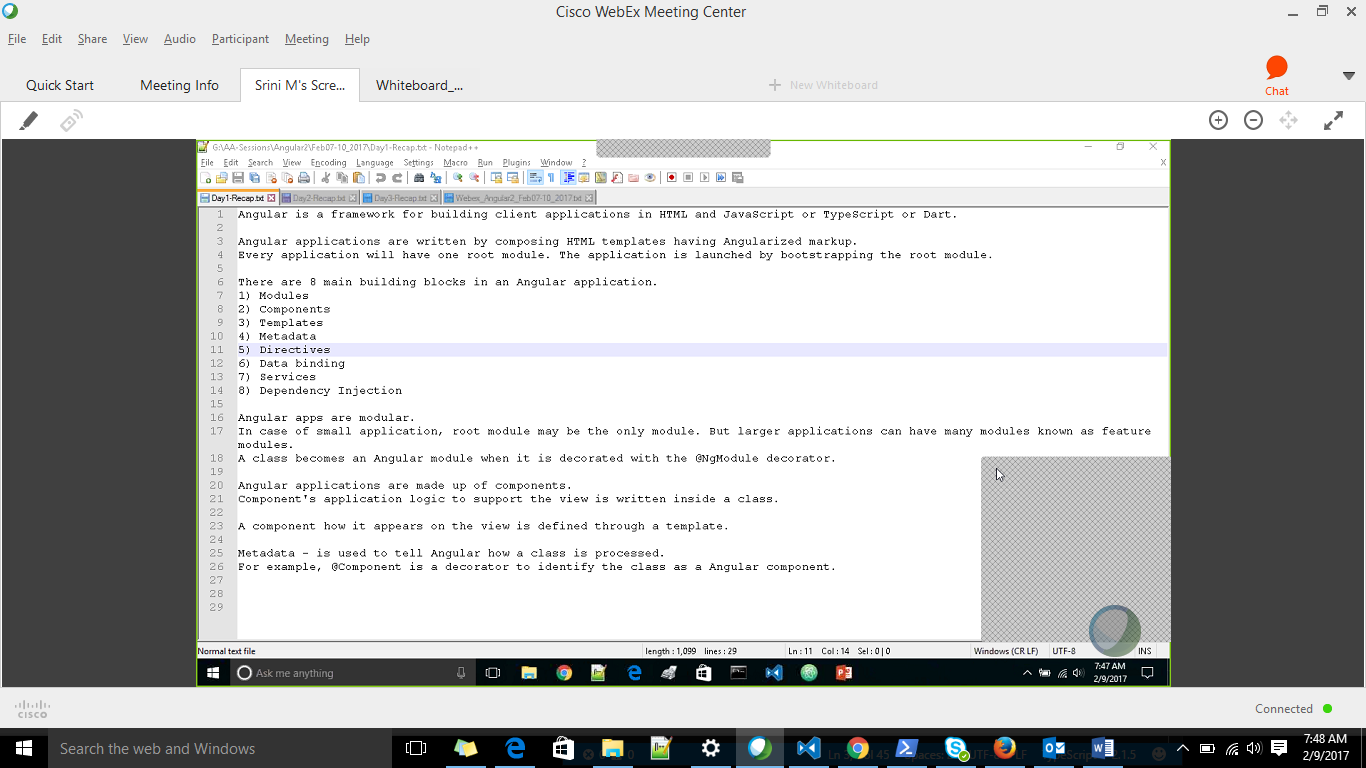
* Property binding – if variable is anything other than string
* For string both interpolation and property binding can b used
* Include FormsModule in ngModule to use ngModel

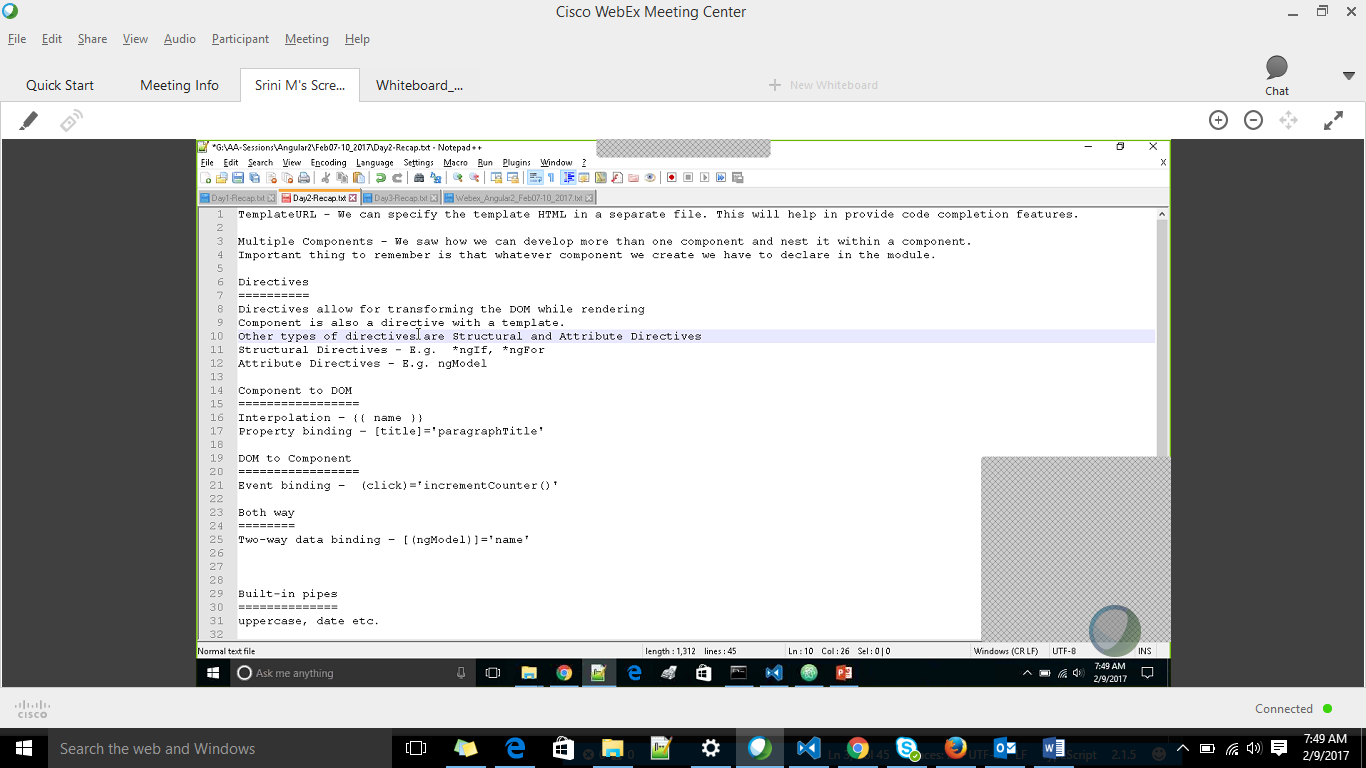
Pipes

* To process the output to different format
* Pre defined pipes
* Custom pipes – implements interface PipeTransform

CSS

* Can have component specific styles like templateURL, styleURLs





Class binding for CSS

<p [Class.selected] = “fruit.selected” onClick(fruit)>

Here .selected is set at component level style

Fruit.selected is changed in click…

When we have component specific CSS and To pass CSS of parent to child component

Use either .deep/ or >>>

Ex in Parent component

/deep/ h1 or >>>h1{

Color :red;

}

# Interaction between components

@Input – Parent to child – use property binding concept

@Output – child to parent – use event binding concept

Here EventEmitter concept of Node.js is used. The child component acts as a publisher and the parent component as a subscriber

Service

* Service class would have @Injectable

@Injectable()

Export class EmployeerService{

//AJX or DB call would come here

}

* Injector is a module that holds instances of all the services
* If any instance is already available then the same is returned else new one is created and returned
* In the component constructor, we define the dependency of the service
* Similarly if any service is dependent on other service then we define that dependency in the constructor of that service
* ngOnInit() – Lifecycle – get the data
* Angular1 uses promises for all the requests.. but these requests once given cannot be cancellable
* Angular2 uses Observables – is like a moving array -- here request can be cancelled
* this.\_http.get(this.\_employeesURL) --- this would return and observable object

this.\_http.get(this.\_employeesURL).map(function(response){

response.json(); -- here http response is converted to a JSON

})

The component subscribes to the data so that the data is received as when the response is recvd in the service

In the

ngOnInit(){

this.\_employeeService.getEmployees().subscribe(employees => this.employees = employees);

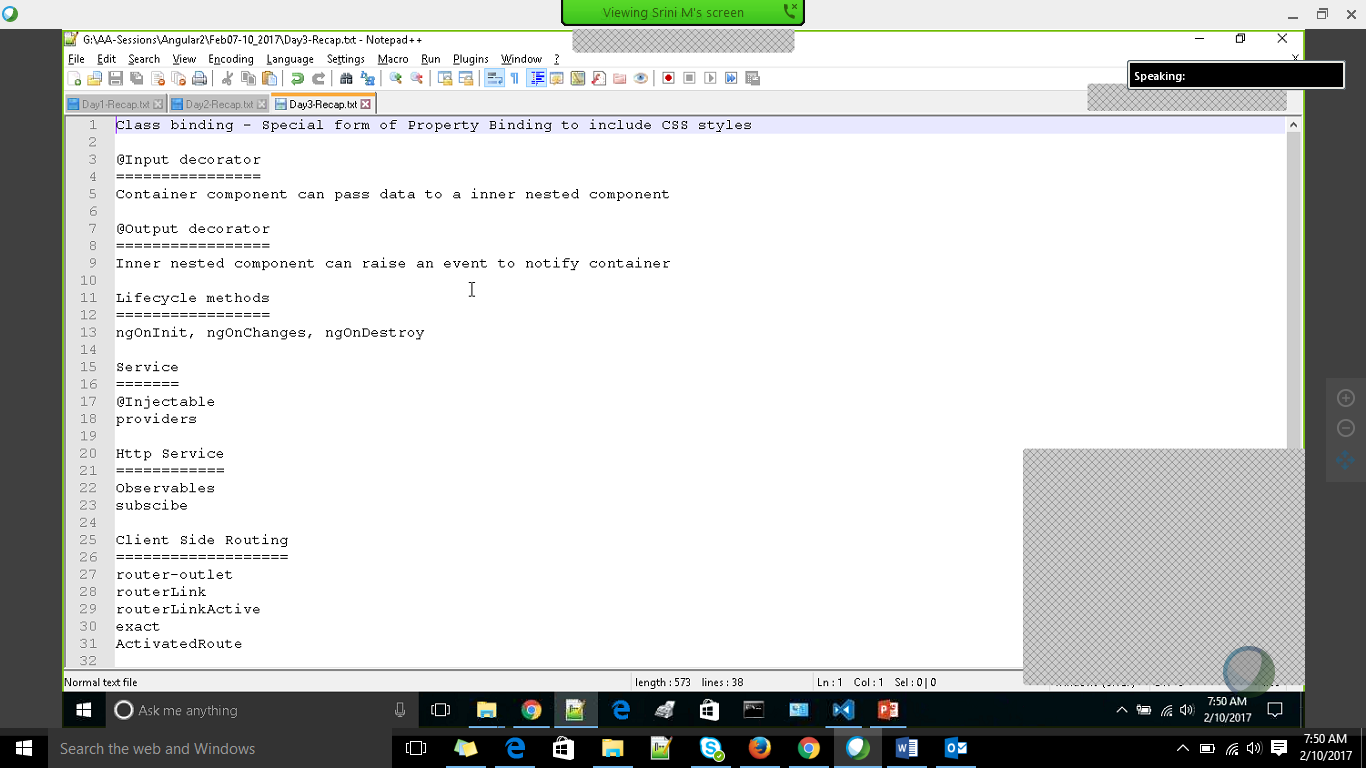
Routing

* router-outlet acts as the ngView – this is where the view from the routes is placed
* to pass dynamic values to route [router-link]=[“/employeeEdit”,employeeId]
* <a router-link = “some-local-route-path”> router link is used to redirect using the router defined
* If a href is given it would redirect to the browser instead pass it our router
* Routerlinkactiver attr for <a> is used to give the active css highlight for <a> tag
* If two routes match lets say the home path and some other then both <a> would be active. To overcome this used exact match

In the component

* Constructor uses ActivatedRoute to know which Is the current route selected like if the emp id is obtained from the route param in ActivatedRoute, we can get other details of emp using this id
* Constructor also has Location which can be used to implmenet back feature

}



References

* https://angular.io/docs/js/latest/guide/upgrade.html
* https://www.typescriptlang.org/
* Dependency Injection -> Martin Fowler
* RxJS – ReactiveX library for Javascript—For observer concepts—to cancel http requests and so
* Angular cli – Build tool https://cli.angular.io/