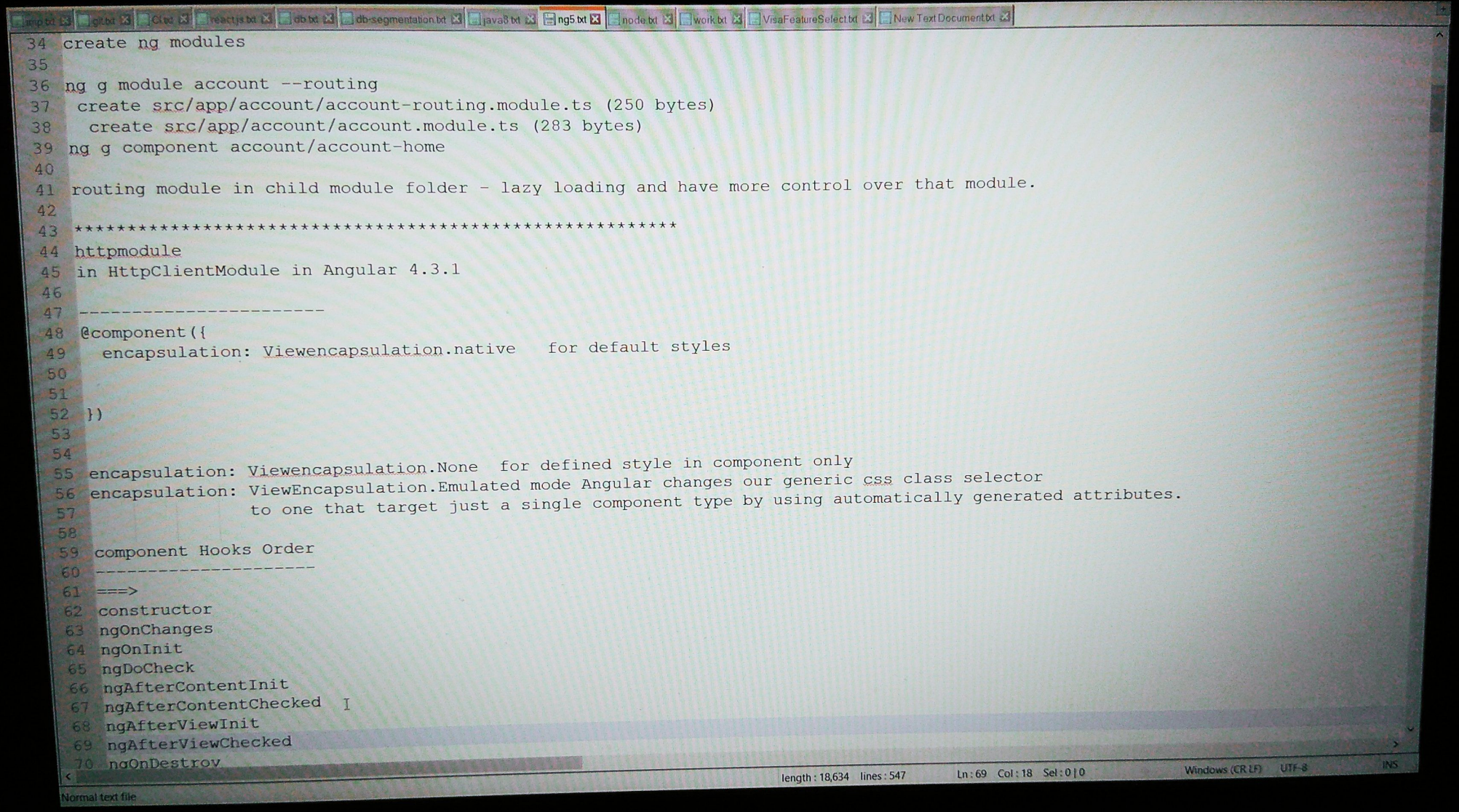
```
>> Router features, such as guards, resolvers, and child routing
Angular 5 Router Lifecycle
==========================
Router Event: NavigationStart
Router Event: RoutesRecognized
Router Event: GuardsCheckStart
Router Event: ChildActivationStart
Router Event: ActivationStart
Router Event: GuardsCheckEnd
Router Event: ResolveStart
Router Event: ResolveEnd
Router Event: ActivationEnd
Router Event: ChildActivationEnd
Router Event: NavigationEnd
*****************************************************

Guard Type: used for protecting urls
CanActivate: Check if a user has access
CanActivateChild: Check if a user has access to any of the child routes
CanDeactivate: Can a user leave a page? For example, they haven't finished editing a post
Resolve: Grab data before the route is instantiated
Lazy loading feature modules.
CanLoad: Interface,  Decides if a module can be loaded lazily.If a canLoad guard returns false,
the router will not load the bundle and user can not see the source code also.

create guard : ng g guard auth

guard parameter:
component: Component this is the component itself.
route: ActivatedRouteSnapshot – this is the future route that will be activated if the guard passes, we can use it's params proper
state: RouterStateSnapshot – this is the future RouterState if the guard passes, we can find the URL we are trying to navigate to

*****************************************************
create ng modules

ng g module account --routing
  create src/app/account/account-routing.module.ts (250 bytes)
```

length : 18,634 lines : 547 Ln : 70 Col : 12 Sel : 11 | 1 Windows (CR LF) UTF-8 INS

Normal text file


mp.txt
git.txt
C.txt
reactjs.txt
db.txt
db-segmentation.txt
java8.txt
ng5.txt
node.txt
work.txt
VisaFeatureSelect.txt
New Text Document.txt
34 create ng modules
35
36 ng g module account --routing
37 create src/app/account/account-routing.module.ts (250 bytes)
38 create src/app/account/account.module.ts (283 bytes)
39 ng g component account/account-home
40
41 routing module in child module folder - lazy loading and have more control over that module.
42
43 **
44 httpmodule
45 in HttpClientModule in Angular 4.3.1
46
47 ------------------------
48 @component({
49 encapsulation: Viewencapsulation.native for default styles
50
51
52 })
53
54
55 encapsulation: Viewencapsulation.None for defined style in component only
56 encapsulation: ViewEncapsulation.Emulated mode Angular changes our generic css class selector
57 to one that target just a single component type by using automatically generated attributes.
58
59 component Hooks Order
60 ---------------------
61 ===>
62 constructor
63 ngOnChanges
64 ngOnInit
65 ngDoCheck
66 ngAfterContentInit
67 ngAfterContentChecked
68 ngAfterViewInit
69 ngAfterViewChecked
70 ngOnDestroy
length : 18,634 lines : 547
Ln : 69 Col : 18 Sel : 0 | 0
Windows (CR LF) UTF-8
INS
Normal text file

```
ngAfterViewChecked
ngOnDestroy
<====

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.

Hooks for the components children:;;;;
These hooks are only called for components and not directives.

ngAfterContentInit - Invoked after Angular performs any content projection into the components
view (see the previous lecture on Content Projection for more info).

ngAfterContentChecked - Invoked each time the content of the given component has been checked by the
change detection mechanism of Angular.

ngAfterViewInit - Invoked when the component's view has been fully initialized.

ngAfterViewChecked - Invoked each time the view of the given component has been checked by the change
detection mechanism of Angular.

@ViewChild -> used to access child DOM element, ViewChild returns the first element that matches a
          given component, directive or template reference selector.
@ViewChildren -> to access multiple children

 >>
components, directives and template reference variable with ElementRef or TemplateRef

================================
RXJS
Asynchronous data streams = in RXJS it is refered as observable sequences or also just called observables,
Observables are very flexible and can be used using push or pull patterns.

When using the push pattern, we subscribe to the source stream and react to new data as soon as
```

RXJS
Asynchronous data streams = in RXJS it is refered as observable sequences or also just called observables,
Observables are very flexible and can be used using push or pull patterns.

When using the push pattern, we subscribe to the source stream and react to new data as soon as
is made available (emitted).

When using the pull pattern, we are using the same operations but synchronously. This happens when
using Arrays, Generators or Iterables.

Subject VS Observables::
Probably a more important distinction between Subject and Observable is that a Subject has state,
it keeps a list of observers. On the other hand, an Observable is really just a function that sets up observation
While Subjects are Observables, Subjects also implement an Observer interface. That is to say, they have next,
error, and complete methods.

Add observer in the subject
```
const subject = new Subject();
// add observer1 to the list of observers
const sub1 = subject.subscribe(observer1);
// add observer2 to the list of observers
const sub2 = subject.subscribe(observer2);
// notify all observers in the list with "hi there"
subject.next('hi there');
// remove observer1 from the list
sub1.unsubscribe();
```

Multicasting: With a multicasting observable, you don't register multiple listeners on the document,
but instead re-use the first listener and send values out to each subscriber.

---

create module with routing ng5

```
$ ng g module xyz --routing
```
this command will create these two files:

```
Multicasting: With a multicasting observable, you don't register multiple listeners on the document,
but instead re-use the first listener and send values out to each subscriber.
-----------------------------------------
create module with routing   ng5

$ ng g module xyz --routing
this command will create these two files:

xyz.module.ts
xyz-routing.module.ts
-------------------------------------
pipe vs directive
 i would say a pipe is to manipulate data, while a directive is more for DOM manipulation.

 Let's say you want to show the currency value in bold text
 and use an image-icon as a currency symbol you probably take a directive
--------------------------------
Types/Interfaces/Decorators are some new in angualar2/4

===============
What is <ng-container>?

A: A grouping element that does not interfere with styles or layout (it's analogous to curly braces in JavaScript).

What is <ng-template>?

A: It's an Angular element for rendering HTML when using structural directives. The ng-template itself does not render to anything
--------------------------
How do components communicate with each other?

A: Various ways - for example: Input/Output properties, services, ViewChild/ViewContent.
--------------------
How would you support logging in your Angular app?

PA: One way would be to use angular2-logger, which is a package inspired by log4j.
-----------------------------
How would you use the ngClass directive?
```

```
159
160 A: Various ways - for example: Input/Output properties, services, ViewChild/ViewContent.
161 ------------------------
162 How would you support logging in your Angular app?
163
164 PA: One way would be to use angular2-logger, which is a package inspired by log4j.
165 ------------------------------------
166 How would you use the ngClass directive?
167
168 A: For example: <div [ngClass]="{firstCondition: 'class1', secondCondition: 'class2'}">...</div>
169 ------------------------------
170 What is the purpose of NgModule?
171
172 A: It's to give Angular information on a particular module's contents, through decorator properties like: declarations, imports,
173 ------------------------------
174 What are the attributes that you can define in an NgModule annotation?
175 or NgModule metadata
176
177 A: Declarations, imports, exports, providers, bootstrap
178
179 Declarations -> component, pipe directive,
180 exports ->
181 imports -> module
182 providers -> service, gaurd
183 bootstrap -> bootstrap the app component   [AppComponent]
184
185 ---------------------------------
186 Can we import a module twice?
187
188 A: Yes, and the latest import will be what is used.
189 ---------------------------------
190 Can you re-export classes and modules?
191
192 A: Yes.
193 ---------------------
194 What kind of classes can you import in an angular module?
195
```

```
A: Components, pipes, directives
-------------------------------------

What is the use case of services?

A: One very common use case is providing data to components,
-------------------------------------
What is the difference between RouterModule.forRoot() vs RouterModule.forChild()? Why is it important?

A: forRoot is a convention for configuring app-wide Router service with routes, whereas forChild is for configuring the routes of

module ->
RouterModule => RouterModule.forRoot() or RouterModule.childRoot()
Routes => define route
const routes: Routes = [
  { path: 'users', component: UsersComponent, pathMatch: 'full', canActivate: [AuthGuard], data: { breadcrumb: "Users" } },
  ..
  ]

=> routerLink="/account" routerLinkActive="active"
-------------------------------------
How would you intercept 404 errors in Angular 2?

A: Can provide a final wildcard path like so: { path: '***', component: PageNotFoundComponent }
-------------------------------------
platformBrowserDynamic is a function used to bootstrap an Angular application.

BrowserModule registers critical application service providers.
It also includes common directives like NgIf and NgFor
which become immediately visible and usable in any of this modules component templates.
-------------------------------------
what is component :

component encapsulate data, logic and markup behind a view.
-------------------------------------
Directive
```

what is component :

component encapsulate data, logic and markup behind a view.

Dierective

A directive is a class with a @Directive decorator.

1 - Structural directives => like *ngIf, *ngFor

2 - Attribute Directive => like [(ngModel)]="hero.name"

Create attribute directive:

src/app/highlight.directive.ts

```
import { Directive, ElementRef, HostListener, Input } from '@angular/core';

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {

  @Input('highlightColor') highlightColor: string;

  constructor(private el: ElementRef) {
    //el.nativeElement.style.backgroundColor = 'yellow';
  }
  @HostListener('mouseenter') onMouseEnter() {
    this.changeColor(this.highlightColor);
  }
  @HostListener('mouseleave') onMouseLeave() {
    this.changeColor(null);
  }
  @HostBinding('class.active') get isActive() {
    return this._active;
  }
  private changeColor(color: string) {
    this.el.nativeElement.style.backgroundColor = color;
  }
```

```
use
 <h3 appHighlight>Users</h3>
 <h3 appHighlight highlightColor="red">passing color</h3>
-----------------------------------------------

others in details:

AppModule -> root module, can contains serveral functional module.

-----------------------------------------------
create pipe

include-text.pipe.ts
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'includeText'
})
export class IncludeTextPipe implements PipeTransform {

  transform(value: any, args?: any): any {
    return value + '('+ args + ')' ;
  }

}

use
{{h.name | includeText: 60}}

-----------------------------------------------------

attribute binding:
<td      [attr.colspan]="mycolsapn"
mycolsapn = 4
=====

class binding:
```

```
class binding:
<    [class.active]="isActive"

 isActive - true/false will render class on citeria
========
style binding:
<a    [style.backgroundColor]="isActive?  "red" : "green" >
<a    [style.backgroundColor]="{'success': 'green', 'danger': 'red'}" >
========
event binding:
<div (click)="doSomething" >

<input (keyup)="keyupqq($event)"
<input (keyup.enter)="keyupqq()"

keyupqq(event)
{
}

<input #email (keyup.enter)="onKeyUp(email.value)" />
=======
<input [value]="email"
one way binding

<input [(ngModel)]="email"
2 way binding  (formsMOdule need to import)

sqaure bracket / parenthesis
-------------------------------------
interface -represent shape of the object

-----------------------------------
shadow DOM
Allows us to apply scoped styles to elements without bleeding
out to the outer world.

var el = document.querySelector('favorite');
```

```
shadow DOM
Allows us to apply scoped styles to elements without bleeding
out to the outer world.

var el = document.querySelector('favorite');
var root = el.createShadowRoot();

@Component({

encapsulation: viewEncapsulation.Emulated; //default uses its own shadow dom approach
                                  .nativeElement;  //
                                  .None  // leaked to outer world
----------------------------------------

ngContent - use in compnent if input is provided to fill that area
<ng-content>
</ng-content>
----------------------------------------
<ng-container>sdasdasd<ng-container>

will not render the ng-container in run time, you will see the content only
----------------------------------------
httpmodule


patch vs put method
The PATCH method - modify only the property. good yo use
put   replace all resource
----------------------------------------
Handling errors:
401 - unauthorized
404 - not found
400 - bad request
405 Method Not Allowed
200 - OK
201 Created success
204 No Content
```

length : 18,634 lines : 547 Ln : 370 Col : 18 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

Normal text file

```
200 - OK
201 Created success
204 No Content
500 internal server error
503 Service Unavailable

expeected error, unexpected error
handle unexpected error:
.subscribe(res => {
    this.res = res;
    },
    error => {
      console.log('unexpected error')
    })

handle expected error:
...
(error: Response) => {
      if(error.status == 400) alert('dsd');
      else
      console.log('unexpected error', err)
    })
--------------------------------
throwing application specific error

implement the catch operator
import 'rxjs/add/operator/catch'

this.http.delete(33).
.catch((error: Response ) => {

  return Observable.throw(new AppError(error))

}

create error specc file in some common folder
create class app.error.ts
```

```
create error specc file in some common folder
create class app.error.ts
create class not-found-error.ts, bas-error.ts etc..
-----------------------------------
Observable vs Promises

Observale - lazy. need to subscribe to execute. allow reactive programming
promises - execute immidiately
-----------------------------------
combineLatest - combile multiple observable and create a single observable
import rxjs/add/operator/catch/combineLatest

switchMap parameter - cleaner approach
-----------------------------------
Authentication
JWT - A JSON Web Token (JWT) is a JSON object that is defined in RFC 7519 as a safe way to represent a set
of information between two parties. The token is composed of a header, a payload, and a signature.
Simply put, a JWT is just a string with the following format:
header.payload.signature

npm install angular2-jwt --save

import {JwtHelper} from 'angular2-jwt';


let jwtHelper  = new JwtHelper();
let expirationDate = jwtHelper.getTokenExpirationDate()
let isExpired = jwtHelper.isTokenExpired(token)

-------------------------------------------
Application Optimization techniques
minification,
uglification,
bundling,
dead code elimination
-------------------------------------------
ng build -prod
```

```
ng build -prod
-----------------------------------------
Firebase for authentication:

import {AngularFireAuth } angularfire2/auth

support login using google, facebook github, user/pass, etc
-----------------------------------------
graphQl - GraphQL is a query language for APIs and a runtime for
fulfilling those queries with your existing data

Send a GraphQL query to your API and get exactly what you need, nothing more and nothing less
-----------------------------------------
angular 6
-----------------------------------------

:host    -  if want to change the style of host element

/* other styles on app.component.css */
/* styles applied directly to the ap-root element only */
:host {
    border: 2px solid dimgray;
    display: block;
    padding: 20px;
}

::ng-deep      (  /deep/ and >>> deprecation   all are same but depriciated)
If we want our component styles to cascade to all child elements of a component, but not to any other element on the page,
we can currently do so using by combining the :host with the ::ng-deep selector:

:host ::ng-deep h2 {
    color: red;
}
                        I
-----------------------------------------
angular6  main chanages

```

```
angular6  main chanages

1 - Angular Elements - Angular Elements is the brainchild of Angular's Rob Wormald.
The Angular Elements package will give us the ability to create an Angular component and then
publish that component as a web component which can be used in any HTML page
(even if that page is not using the Angular framework) in other environments.

2 - RXJS 6 - new

3 - Service Worker Support

4 - <template> removed, we now need to use <ng-tempalate>.

5 - i18n - changes in Angular 6 is internationalization or i18n

6 - Ivy - is the next generation Angular rendering engine.
Angular compiles our templates into equivalent TypeScript code and then that TypeScript
code is compiled to JavaScript and then the result is shipped to our users.
So Ivy renderer is the new rendering engine which is basically designed to support backward
compatibility with existing renderers and then also focused to improve the speed of rendering
and it also optimized the size of the final package.

7 - ngModelChange - ngModelChange event was emitted before the said form control updating.
Now, in Angular 6, ngModelChange has emitted the value after the value is updated in the form control.

8 - ElementRef<T> - can use type ElementRef more strictly if we want.

ElementRef<HTMLInputElement>;

9 - Bazel Compiler - Instead of rebuilding the entire application, we build only the code which actually
changes and also that code that depends on those changes.

10 - Tree Shaking: to ensure any unused code does not get used in our final bundle
----------I----------

ControlValueAccessor: used when creating custom form control.
like CustomDatePicker.A ControlValueAccessor acts as a bridge between the Angular forms API and a native element in the DOM.
```

```
---------------------------------------------

ControlValueAccessor: used when creating custom form control.
like CustomDatePicker.A ControlValueAccessor acts as a bridge between the Angular forms API and a native element in the DOM.

import { Component, forwardRef, Host, Input, OnChanges, OnInit, Optional, SimpleChanges, SkipSelf, HostListener } from '@angular/c
import { AbstractControl, ControlContainer, ControlValueAccessor, NG_VALUE_ACCESSOR } from '@angular/forms';

@Component({
  selector: 'test-time-picker',
  templateUrl: './test-time-picker-soething.html',
  styleUrls: ['./test-something.scss'],
  providers: [
    {
      provide: NG_VALUE_ACCESSOR,
      useExisting: forwardRef(() => TestTimePickerCompoent),
      multi: true,
    },
  ],
})


3 methods to impelment:
writeValue
registerOnChange
registerOnTouched
-----------------------------------------------
AOT - Ahead of time compiler
-----------------------------------------------
The entryComponents array is used to define only components that are not found in html and created dynamically with
ComponentFactoryResolver. Angular needs this hint to find them and compile. All other components should just be
listed in the declarations array.

like modal services or popup
```