Front - End Development

Introduction to UI stack

Armine Grigoryan

02/26/2021

Introduction: Speaker

Armine Grigoryan

Safeway - UI Lead

- Manage web applications in Catalog

Agenda

- Web Best Practice
- Chrome Developer Tool
- UI Stack
  - HTML5/CSS3
  - Javascript/ES6/Typescript
  - Angular
  - Angular CLI
- Setting up the local environment and workspace
- Exercise
- Catalog UI overview


Web Best Practice

Web Best Practice

When doing web development, the main source of uncertainty comes from the fact that you don't know what combination of technology each user will use to view your web site:

- User 1 might be looking at it on an iPhone, with a small, narrow screen.
- User 2 might be looking at it on a Windows laptop with a widescreen monitor attached to it.
- User 3 might be blind, and using a screen reader to read the web page out to them.
- User 4 might be using a really old desktop machine that can't run modern browsers.

Because you don't know exactly what your users will use, you need to design defensively — make your web site as flexible as possible, so that all of the above users can make use of it, even if they might not all get the same experience. In short, we are trying to make the web work for all, as much as possible.

You'll come across the below concepts at some point in your studies.

Web Best Practice

- **Cross-browser compatibility** is the practice of trying to make sure your webpage works across as many devices as possible.

a. If you use a well-established framework, whether for styling (eg **Bootstrap**) or a JavaScript framework (such as **Angular** or **React**), generally this means someone else has taken care of a lot of the cross-browser compatibility work for you.

- **Responsive web design** is the practice of making your functionality and layouts flexible so they can automatically adapt to different browsers and devices.

b. Bootstrap gives you ability to create flexible and responsive web layouts with much less efforts.

c. Bootstrap 4 is the newest version of Bootstrap; with new components, faster stylesheet and more responsiveness.

d. Bootstrap 4 supports the latest, stable releases of all major browsers and platforms

e. https://kinsta.com/blog/responsive-web-design/

Web Best Practice

- **Performance** means getting web sites to load as quickly as possible, but also making them intuitive and easy to use so that users don't get frustrated and go somewhere else.

a. Lazy Loading technic
b. Minifying libraries (CSS, JS)
c. Reduce image size
d. Reduce the number of HTTP requests

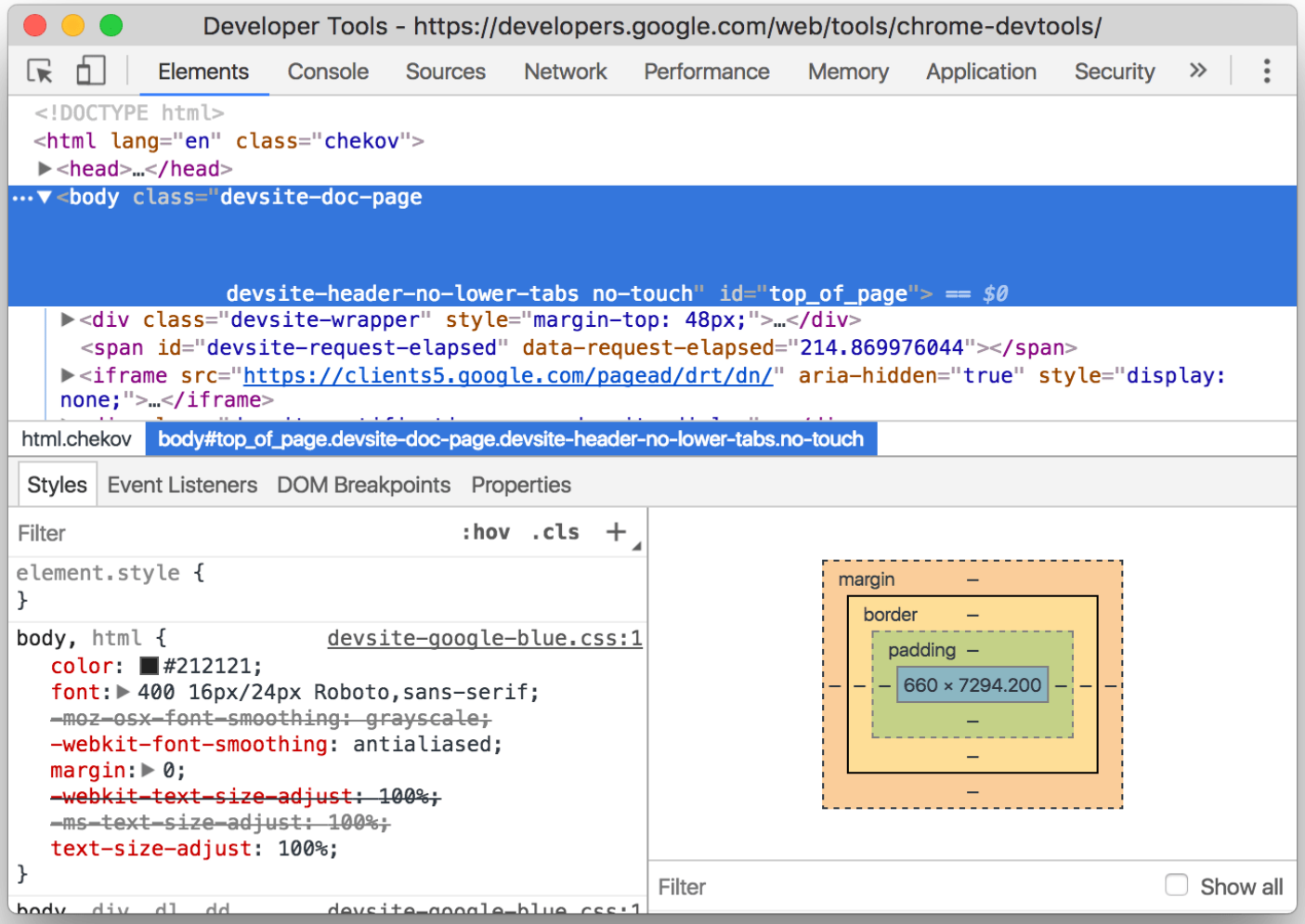
- **Accessibility** means making your websites usable by as many different kinds of people as possible (related concepts are diversity and inclusion, and inclusive design). This includes people with visual impairments, hearing impairments, cognitive disabilities, or physical disabilities.

Debug with Developer Tools

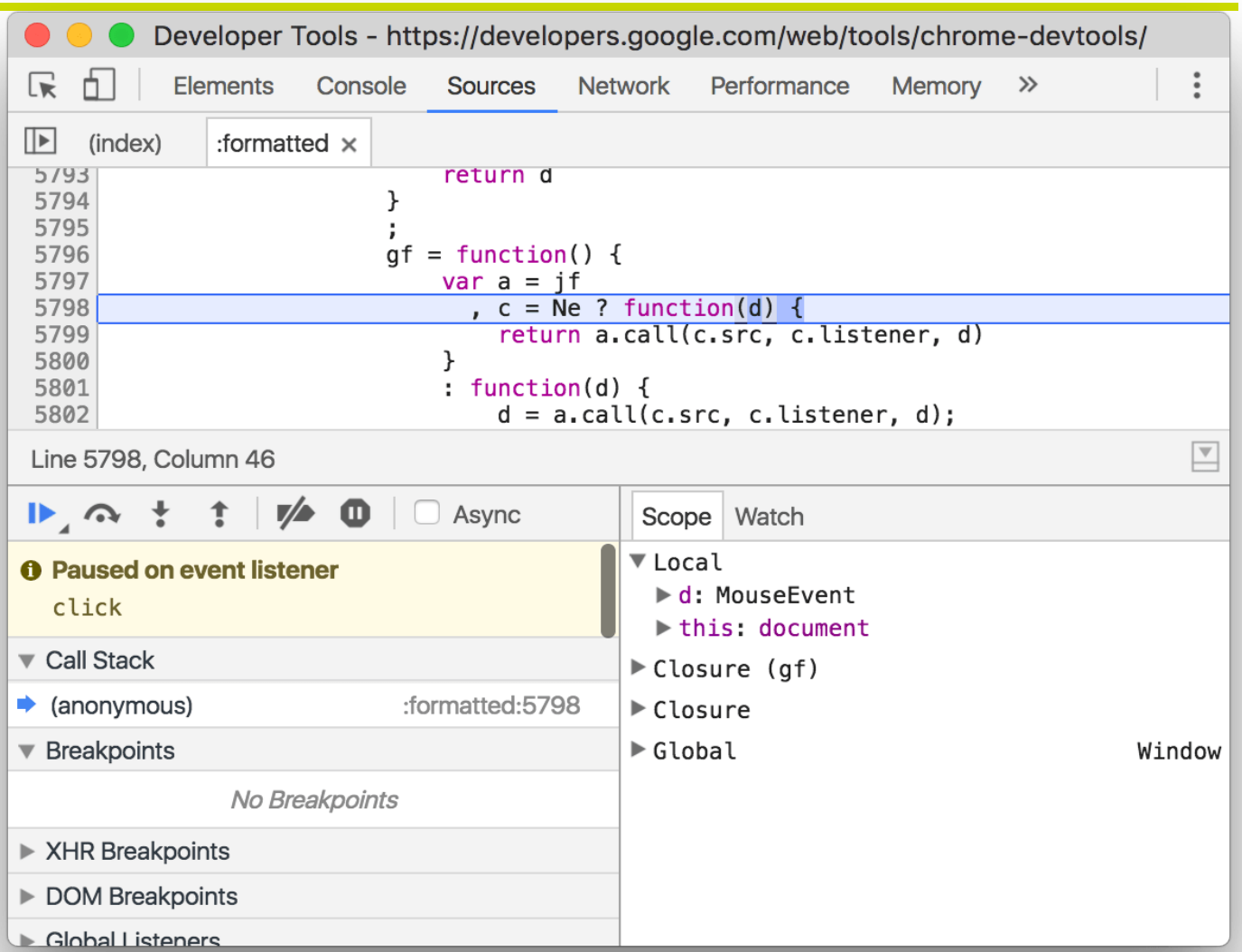
Chrome DevTools is a set of web developer tools built directly into the Google Chrome browser. DevTools can help you edit pages on-the-fly and diagnose problems quickly, which ultimately helps you build better websites, faster.

https://developers.google.com/web/tools/chrome-devtools

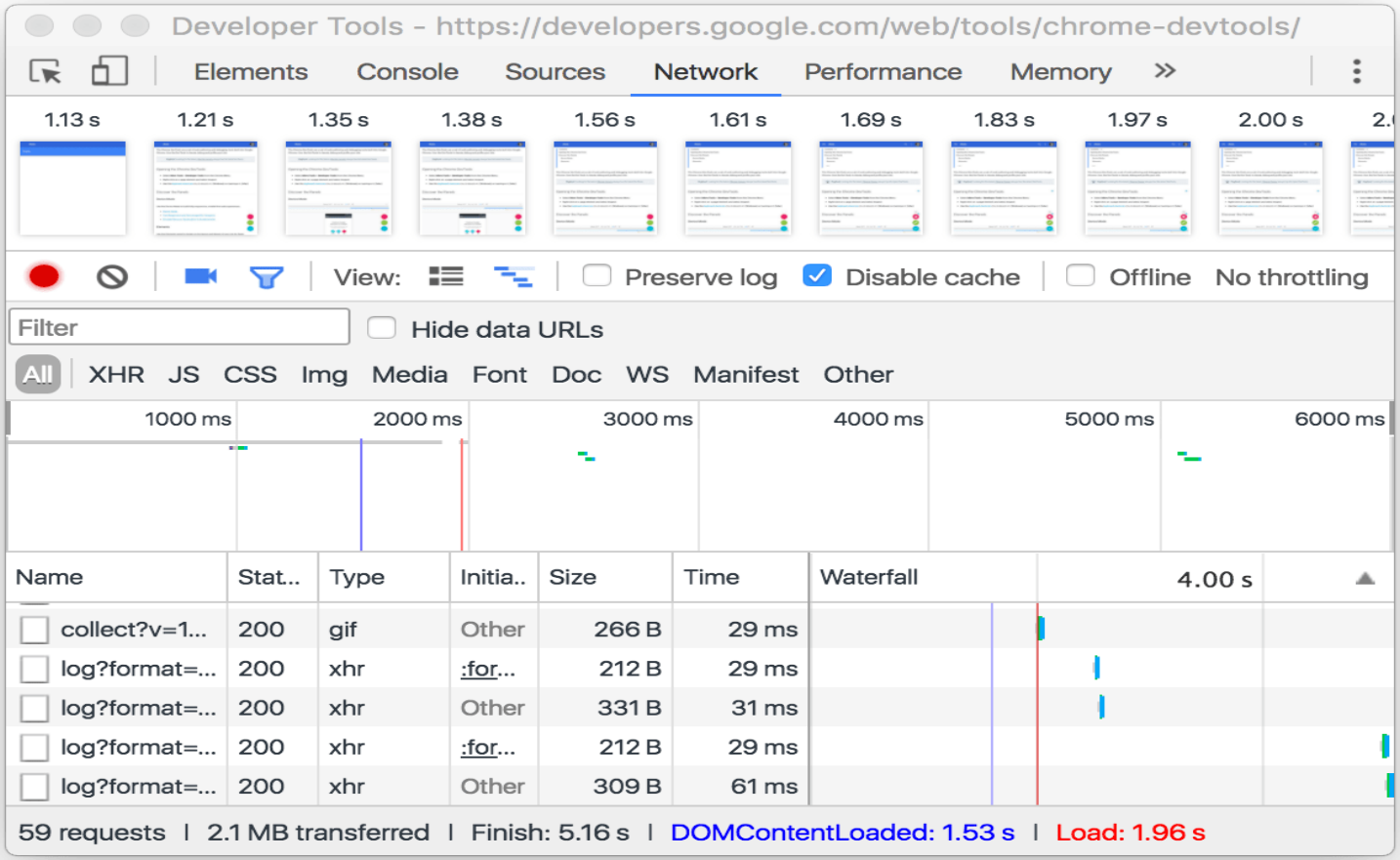
Chrome Developer Tool: elements panel



Chrome Developer Tool: sources panel



Chrome Developer Tool: network panel



UI Stack

- HTML5/CSS3
- Javascript/ES6/Typescript
- Angular
- Angular CLI

UI Stack

- HTML, the language that gives web content structure and meaning
- CSS, the language used to style web pages
- JavaScript, the scripting language used to create dynamic functionality on the web

Angular Basic Architecture

Angular is a development platform, built on TypeScript. As a platform, Angular includes:

- A component-based framework for building scalable web applications
- A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
- A suite of developer tools to help you develop, build, test, and update your code

Angular Life Cycles or Lifecycle Hooks:

ngOnChanges(): ngOnInit(): ngDoCheck(): ngAfterContentInit(): ngAfterContentChecked(): ngAfterViewInit(): ngAfterViewChecked(): ngOnDestroy():

Directives:

NgModel, NgFor, NgIf

```
<input [(ngModel)]="currentItem.name" />
<div *ngIf="currentCustomer">Hello, {{currentCustomer.name}}</div>
<div *ngFor="let item of items">{{item.name}}</div>
```

Angular Basic Architecture

Modules: NgModule

The purpose of a NgModule is to declare each thing you create in Angular, and group them together

Example:

```
@NgModule({
declarations: [SomeComponent, SomeDirective, SomePipe],
providers: [SomeService]
})
```

Components/Services and Observables

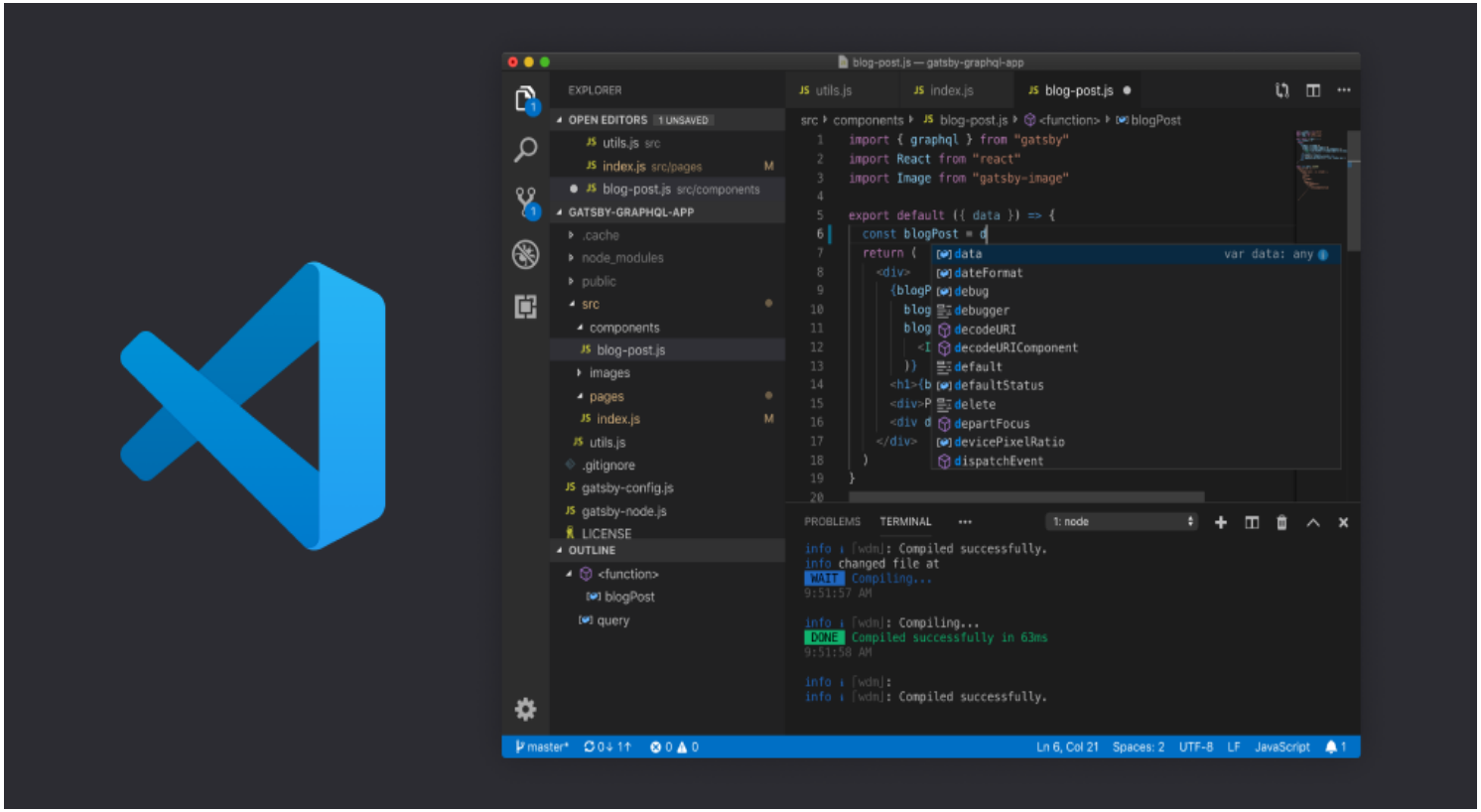
Components are the fundamental building blocks of Angular applications. They display data on the screen, listen for user input, and take action based on that input.

Services are usually implemented through *dependency injection*. When we talk about an *injectable service class*, we're simply talking about common, service-oriented code that can be reused between separate components.

Observable are used within Angular itself, including Angular's event system and its http client service. To use observable, Angular uses a third-party library called Reactive Extensions (**RxJS**)

IDE: Visual Studio Code

Project in Visual Studio

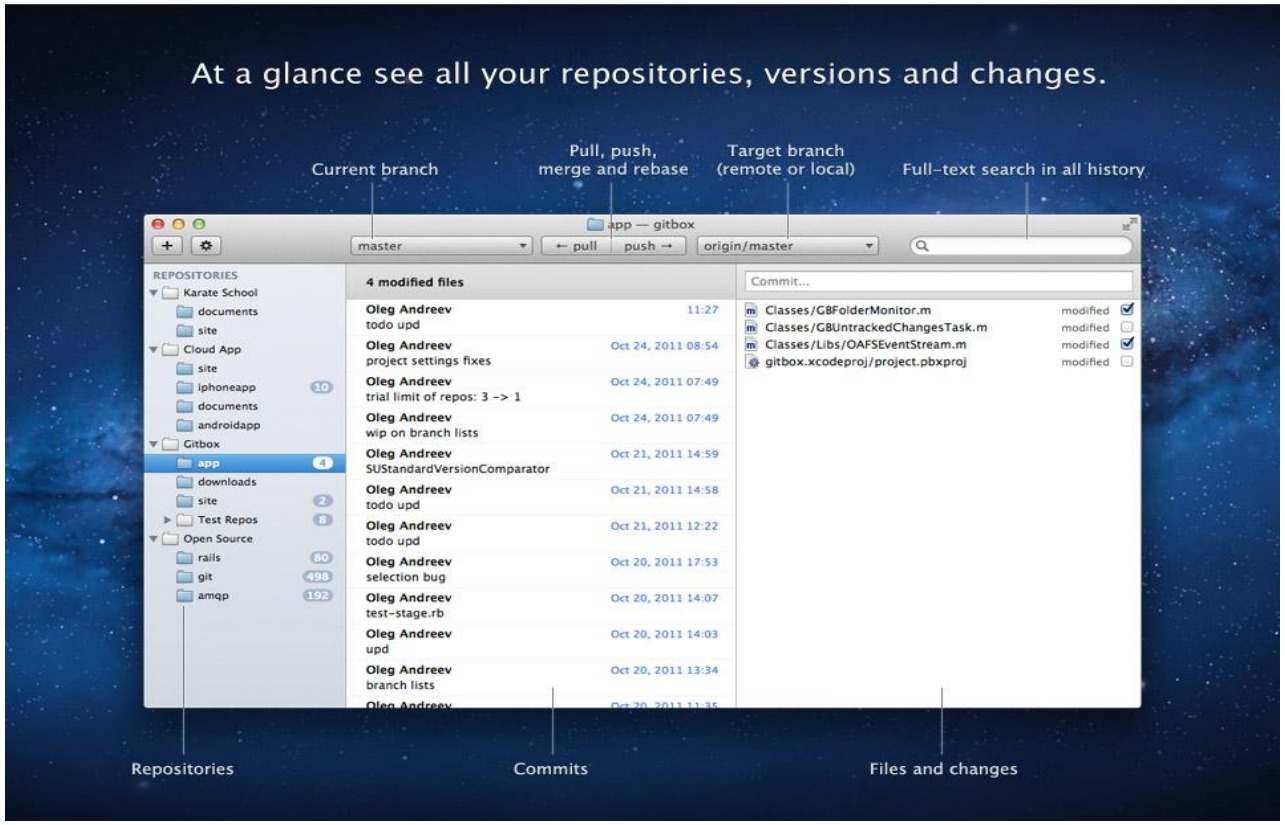


Git

Basics GIT commands: git pull origin develop,

git push origin <master>, command git checkout -b <branch-name>, git merge <branch-name>

Gitbox

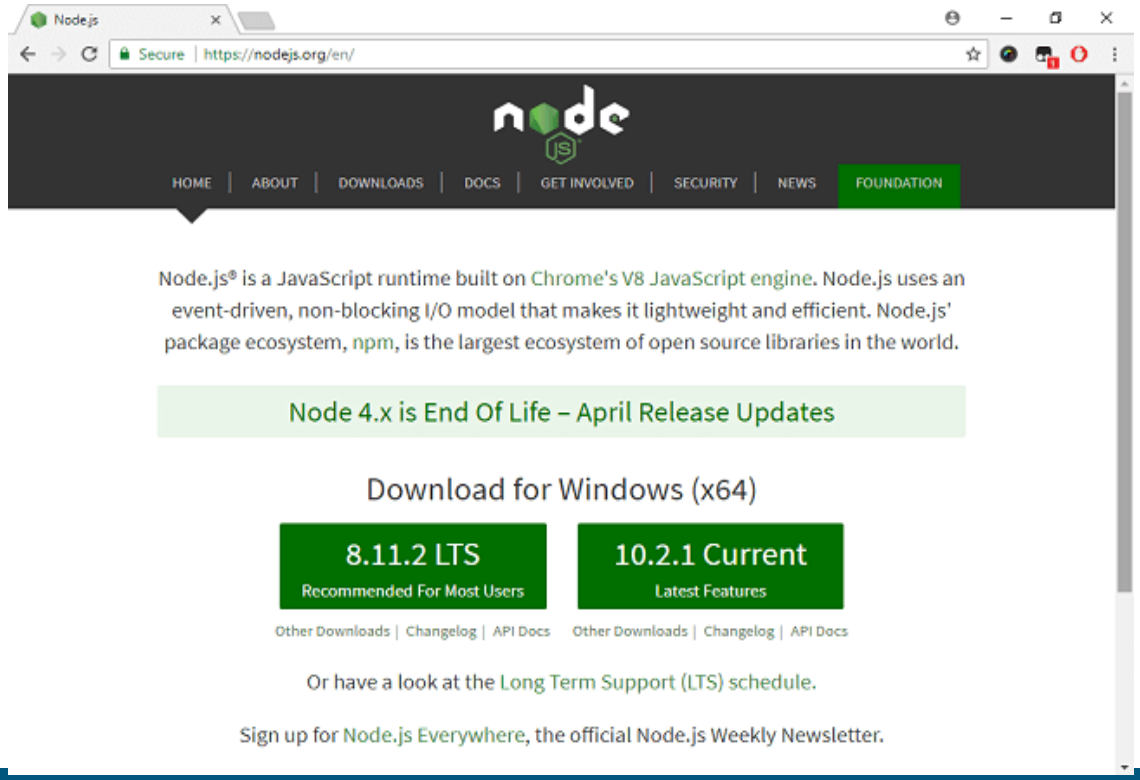



Setup Development
Environment

Development Environment

- **Node**
- Npm
- Angular CLI
- Text-editor - Visual Studio code

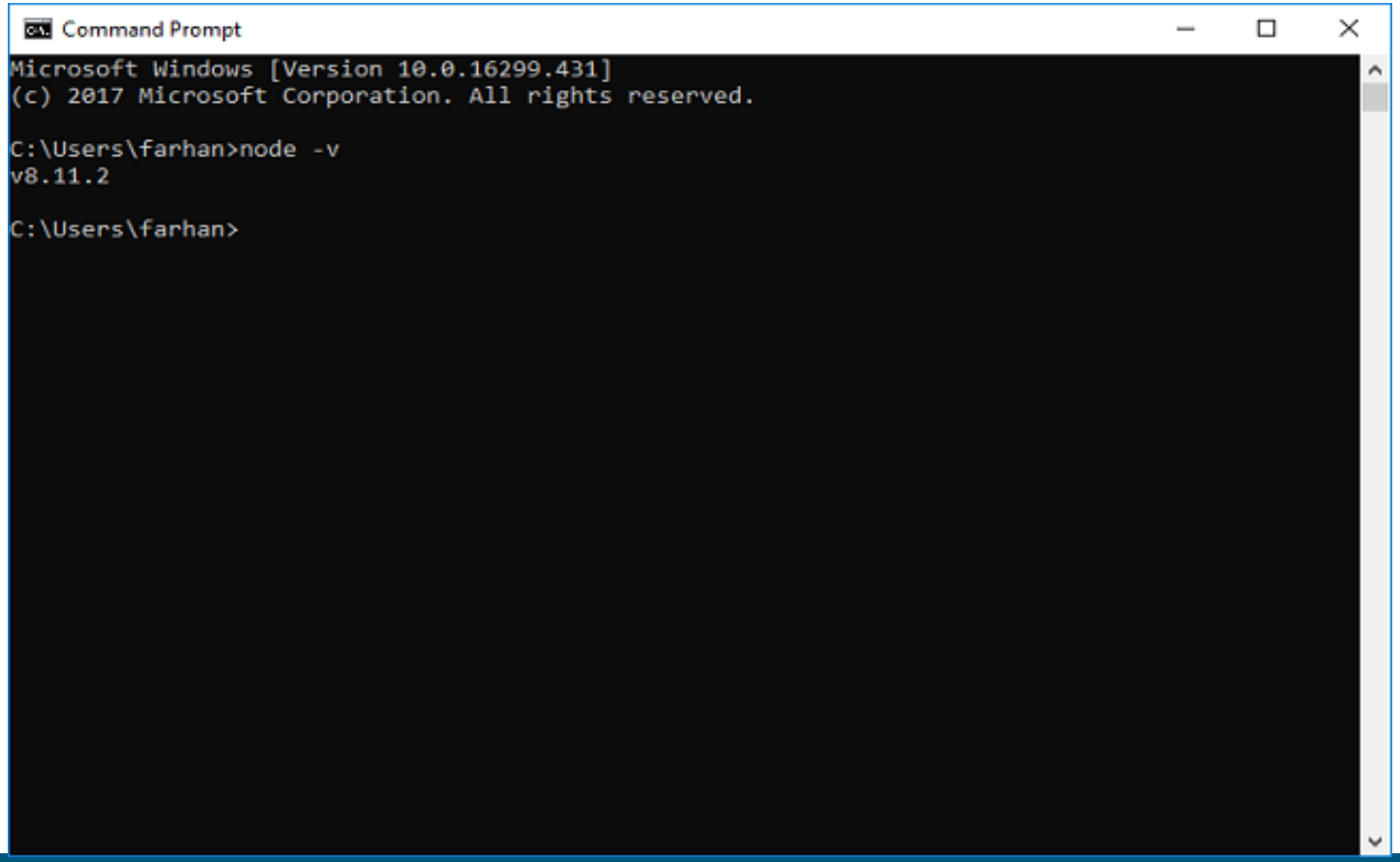
Step 1: Open a browser type *https://nodejs.org/en/* download, and install node js based on your window bit.



Development Environment

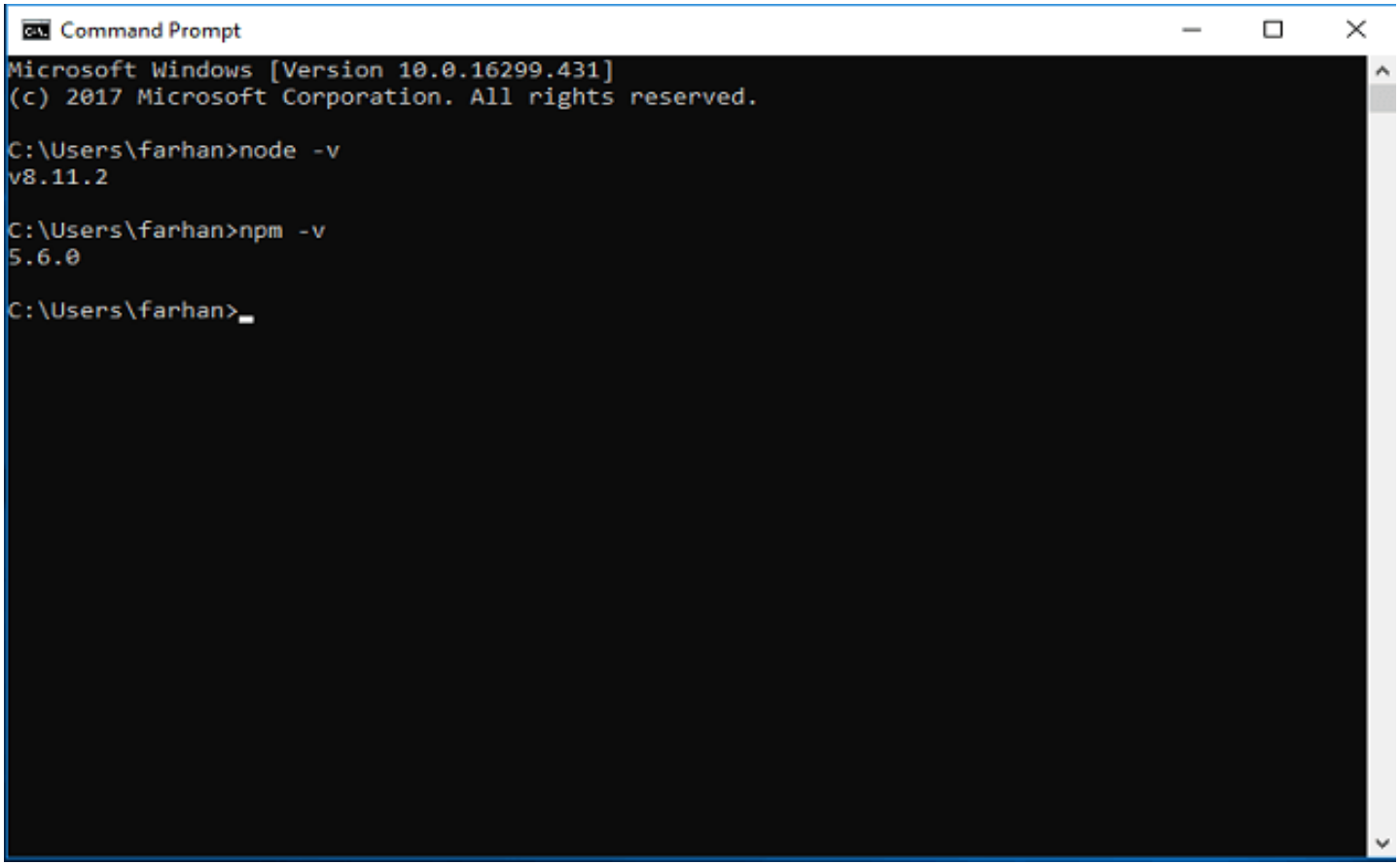
Step 2

After installing node js, open command prompt type *node –v* to check installed version of node js.



Development Environment

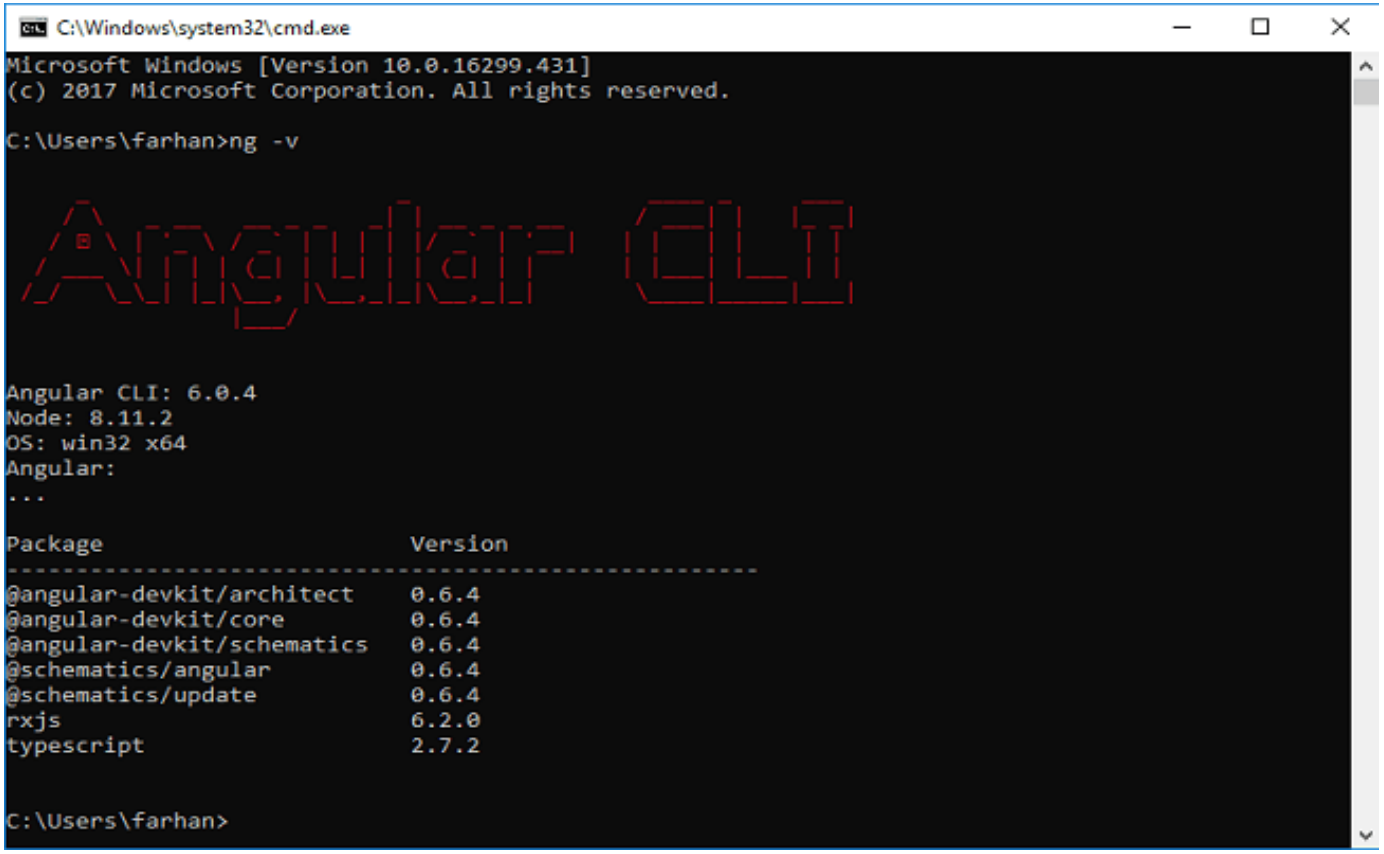
Type *npm –v* to check npm version. If both the commands show their respective versions it means node is installed successfully.



Development Environment

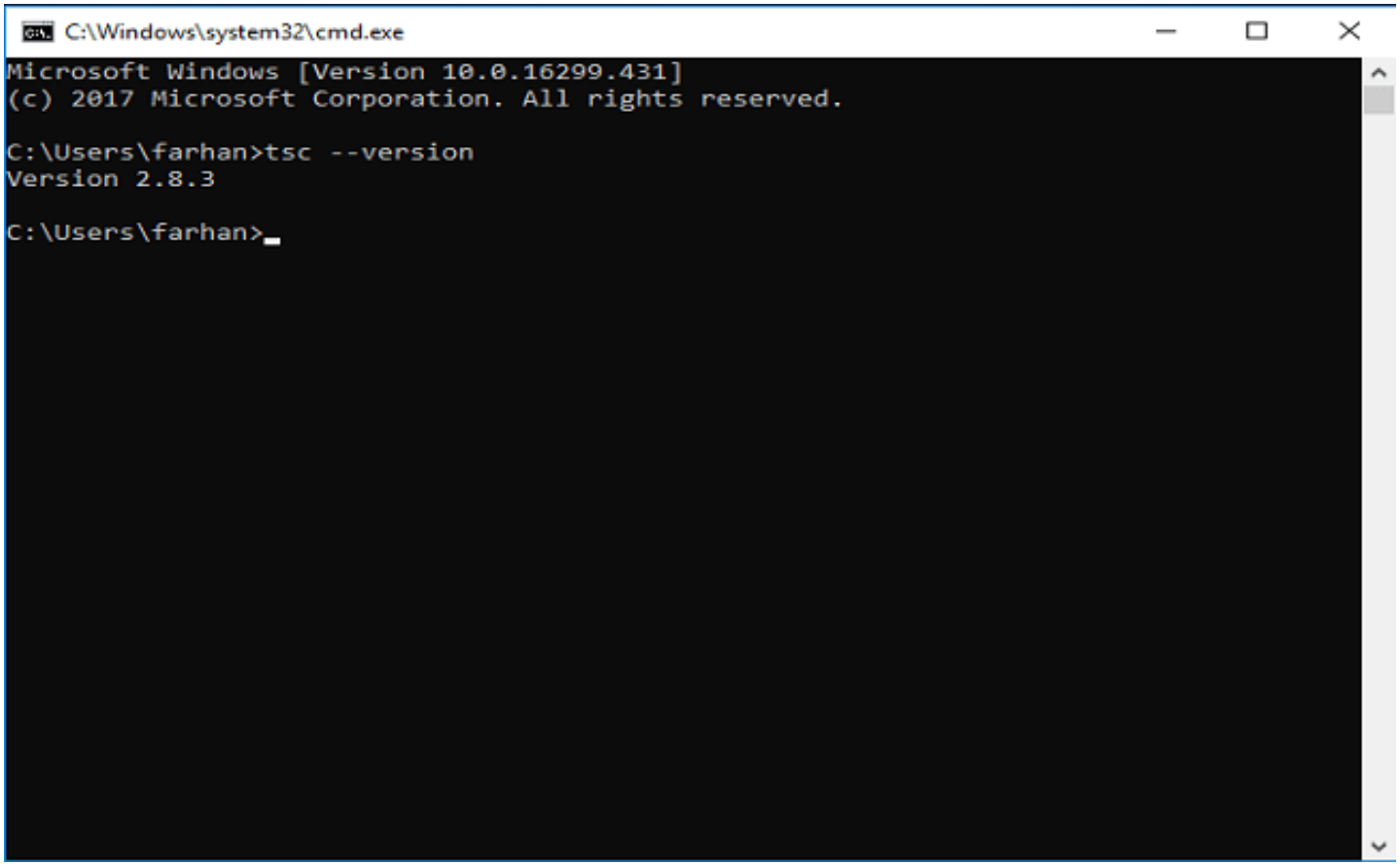
Step 3: Open browser type *https://cli.angular.io/* check command to install Angular CLI.

Step 4: Open command prompt, type command **npm install –g @angular/cli.** It will install CLI globally where g represents globally. To check Angular CLI installed type command ng –v



Development Environment

Step 5: Open command prompt type npm install –g typescript, it will install TypeScript in your system. Now type tsc –version in command prompt to check the version of typescript installed.



Development Environment

Step 6

Open browser, type *https://code.visualstudio.com/* download, and install visual code editor.

Setup local env: https://angular.io/guide/setup-local

if you have EACCESS errors please use Sudo: https://docs.npmjs.com/resolving-eacces-permissions-errors-when-installing-packages-globally

Angular tutorial:

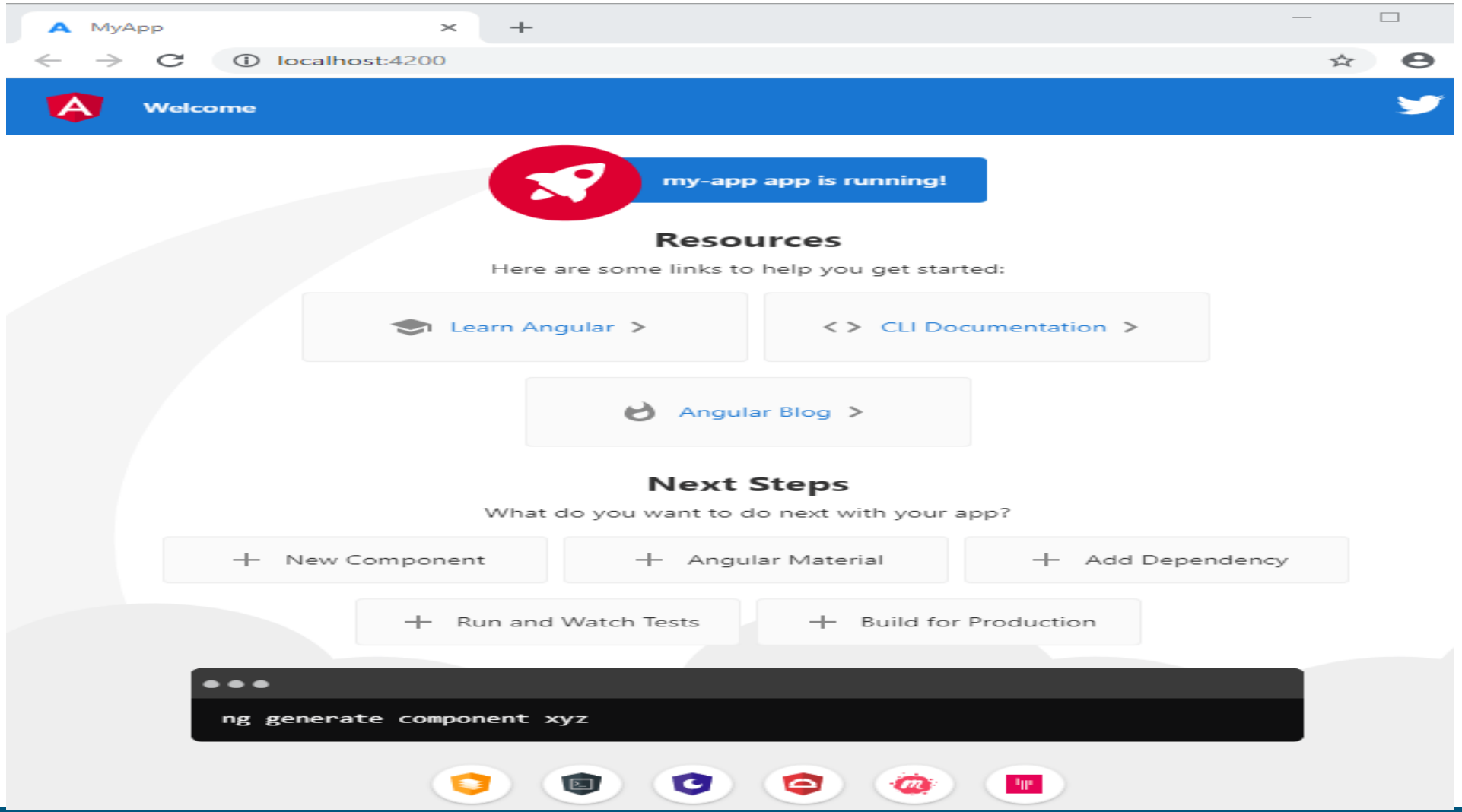
https://angular.io/tutorial
https://angular.io/tutorial/toh-pt0
https://github.com/johnpapa/angular-tour-of-heroes

Open Visual Studio and click on Terminal

Create new projects: https://angular.io/guide/setup-local

```
npm install -g @angular/cli
ng new my-app
cd my-app
ng serve --open
```

MY-APP Page


MyApp
localhost:4200
Welcome
my-app app is running!
Resources
Here are some links to help you get started:
Learn Angular >
CLI Documentation >
Angular Blog >
Next Steps
What do you want to do next with your app?
New Component
Angular Material
Add Dependency
Run and Watch Tests
Build for Production
ng generate component xyz

Basic Examples

Basic exercise:

- Create an input field
  - `<input id='text'/>`
- Create a button
  - `<button class='btn' (click)='show()'>Show</button>`
- Call a method on button click
  - `show(){`
    `console.log(this.email);`
    `}`
- Create a drop down
  - `<select >`
    `<option *ngFor="let item of frameworks">{{item}}</option>`
    `</select>`

`frameworks=['Angular', 'React', 'Vue']`

Basic Examples

- Example with *ngIf

```
        </div>
         <div *ngIf='currentUser'> Hello {{user.name}}
        </div>
      currentUser: boolean = false;
```

- Example with *ngFor

```
        <select >
         <option  *ngFor="let item of frameworks">{{item}}</option>
        </select>
    }
```

Basic Examples

- One way binding
  - <input [value] = 'email' />

email = ‘me@example.com’;

- Two way binding
  - <input [(ngModel)]= 'email' />

```
email;
show(){
  console.log(this.email);
}
```

Assignments

- Search for a country and display country with its capital.
  - create an input box and a button
  - display the result on button click
  - for example: search for Italy and display: Italy: Rome
- Populate all countries names, capitals and populations on button click
  - using *ngFor populate list of countries

List of countries will be send separately

Assignments

Create a new application:

in Terminal type **ng new myApp,** go to myApp(**cd myApp**)

ng new myApp
cd myApp
ng serve --open

You should see page in localhost.

Lets delete all content in app.component.html and save. The page automatically is refreshed.

Continue to install all dependencies we need for app. Run in Terminal:

npm install, npm install bootstrap, npm install @ng-bootstrap/ng-bootstrap.

In project file folder click on **index.html** and insert these libraries**:**

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.12.1/css/all.css" crossorigin="anonymous">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" crossorigin="anonymous">

Assignments

Check **package.json** file for dependencies :

```
"dependencies": {
  "@angular/animations": "~11.2.5",
  "@angular/common": "~11.2.5",
  "@angular/compiler": "~11.2.5",
  "@angular/core": "~11.2.5",
  "@angular/forms": "~11.2.5",
  "@angular/platform-browser": "~11.2.5",
  "@angular/platform-browser-dynamic": "~11.2.5",
  "@angular/router": "~11.2.5",
 "@ng-bootstrap/ng-bootstrap": "^9.1.0",
  "bootstrap": "^4.6.0",
  "rxjs": "~6.6.0",
  "tslib": "^2.0.0",
  "zone.js": "~0.11.3"
},
```

Assignments

1. Create Header and Footer and components
   a. ng g c header
   b. ng g c footer
   c. ng g c england
   d. ng g c france
   e. ng g c spain
   f. ng g c greece
   g. style header and footer(color it)
2. Have a buttons or hyperlinks (countries: France, Greece, England or any).
3. Each item in the header navigates to page with some pictures about the country
4. Have some content in Footer.(Careers, About Us and etc)
5. Design each page with some text or pictures.

Template

england | france | Greece | Spain | Account

dynamic content

app.component.html

Include selectors in app.component.html to run application main page

```
<header></header>
<div class='content' role='main'>
 <div class='dynamic-container col-md-10 col-xl-10'>
   <router-outlet></router-outlet>
 </div>
</div>
<footer></footer>
```

<router-outlet> informs Angular to update the application view with the component for the selected route.

app.component.css

Styling dynamic container:

```
.dynamic-container{
   margin-top: 35px;
   height: calc(100vh-70px);
   overflow-y: auto;
   overflow-x: hidden;
   padding: 50px 20px 20px 50px;
}
```

Create England template

Add pictures in the **england.component.html**:

```
<div class='imgContainer' >
<img  class='img' src='/assets/images/england/england-bridge.jpg' />
<img  class='img' src='/assets/images/england/england-parliament.jpg' />
<img  class='img' src='/assets/images/england/england-stonehedge.jpg' />
</div>
```

And styling in **england.component.css**:

```
.img{
   height: 350px;
   width: 350px;
   padding-left: 5px;
}
```

Create Account page

First, we need to create an account component:

Type in Terminal: **ng g c account**

In **account.component.ts** need to import FormBuilder and Validators modules:

import { FormBuilder,Validators } from '@angular/forms';

Lets start to create template in **account.component.html:**

Account template(html)

```
<p>account works!</p>
<div>
<form [formGroup]="userForm" (ngSubmit)="onSubmit1()">

  <p>Sign up</p>

  <div>
    <div>
      <input type="text" formControlName="firstName"  class="form-control input" placeholder="First name">
    </div>
    <div>
      <input type="text" formControlName="lastName" class="form-control input " placeholder="Last name">
    </div>
  </div>

  <input type="email" formControlName="email"  class="form-control  input" placeholder="E-mail">
  <div>
    <button  type="submit">Sign in</button>
  </div>
 </form>
 <div >
  <ngb-alert  *ngIf="showMessage" type="danger" >{{ invalidEntry }}</ngb-alert>
  <!-- <div class= "invalid-text" *ngIf="showMessage">
    {{invalidEntry}}
  </div> -->
 </div>
</div>
```

Account page CSS

```
.input{
   width: 15em;
   height: 2em;
   margin-bottom: 1em;
}
 .invalid-text{
     color:red;
 }
```


Catalog UI

Catalog UI

Catalog UI is a single page application.

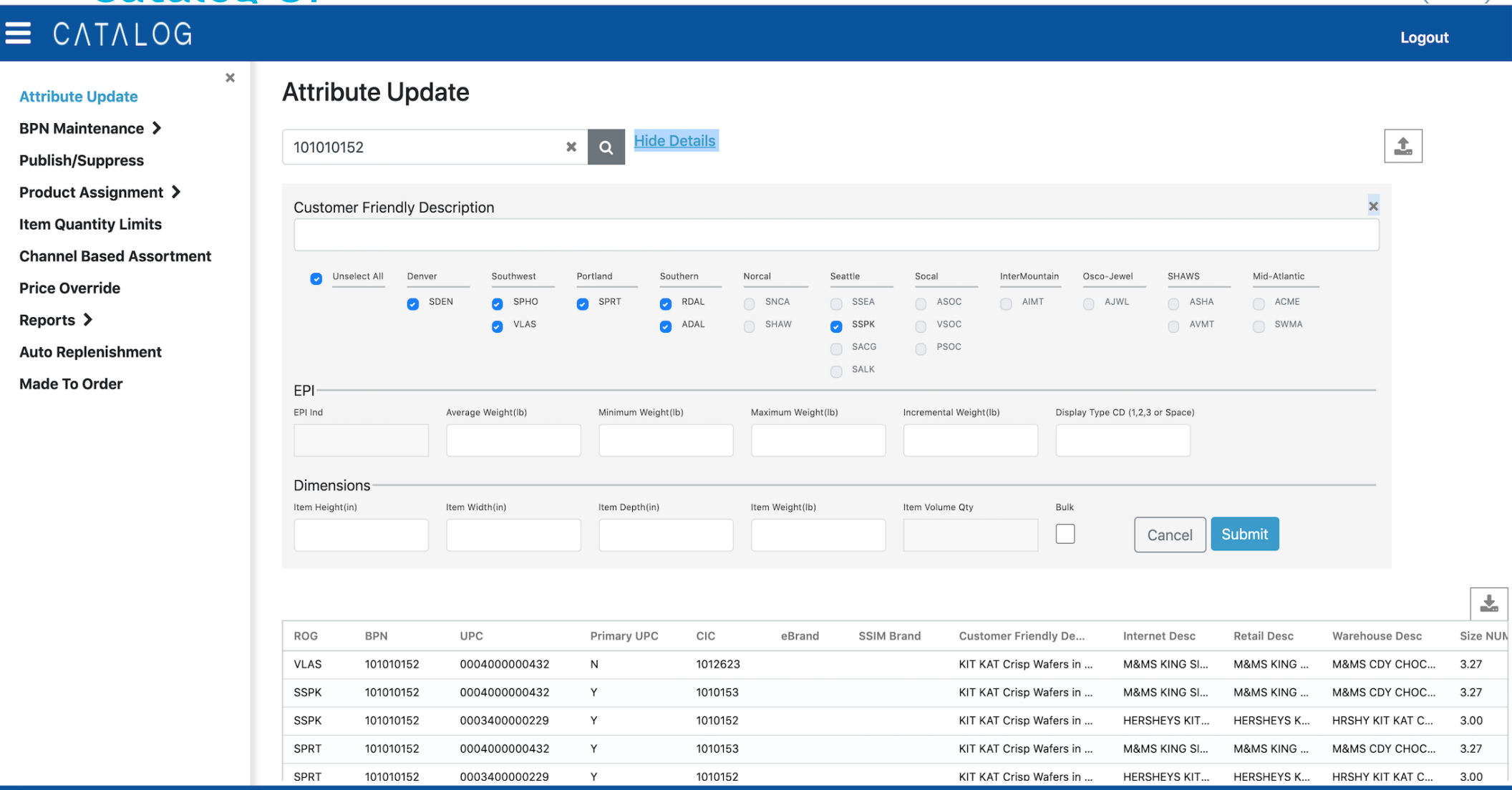
Single Page Applications are web **applications** that load a **single** HTML **page** and only a part of the **page** instead of the entire **page** gets updated with every click of the mouse.

As the client clicks certain parts on the webpage, only the required part of the information is fetched from the server and the page is rewritten dynamically. With certain clicks we call api using GET, POST , PUT and DELETE methods. For example when user search for product we use GET call.

When user updates the data we call PUT by sending payload and get successful response back.

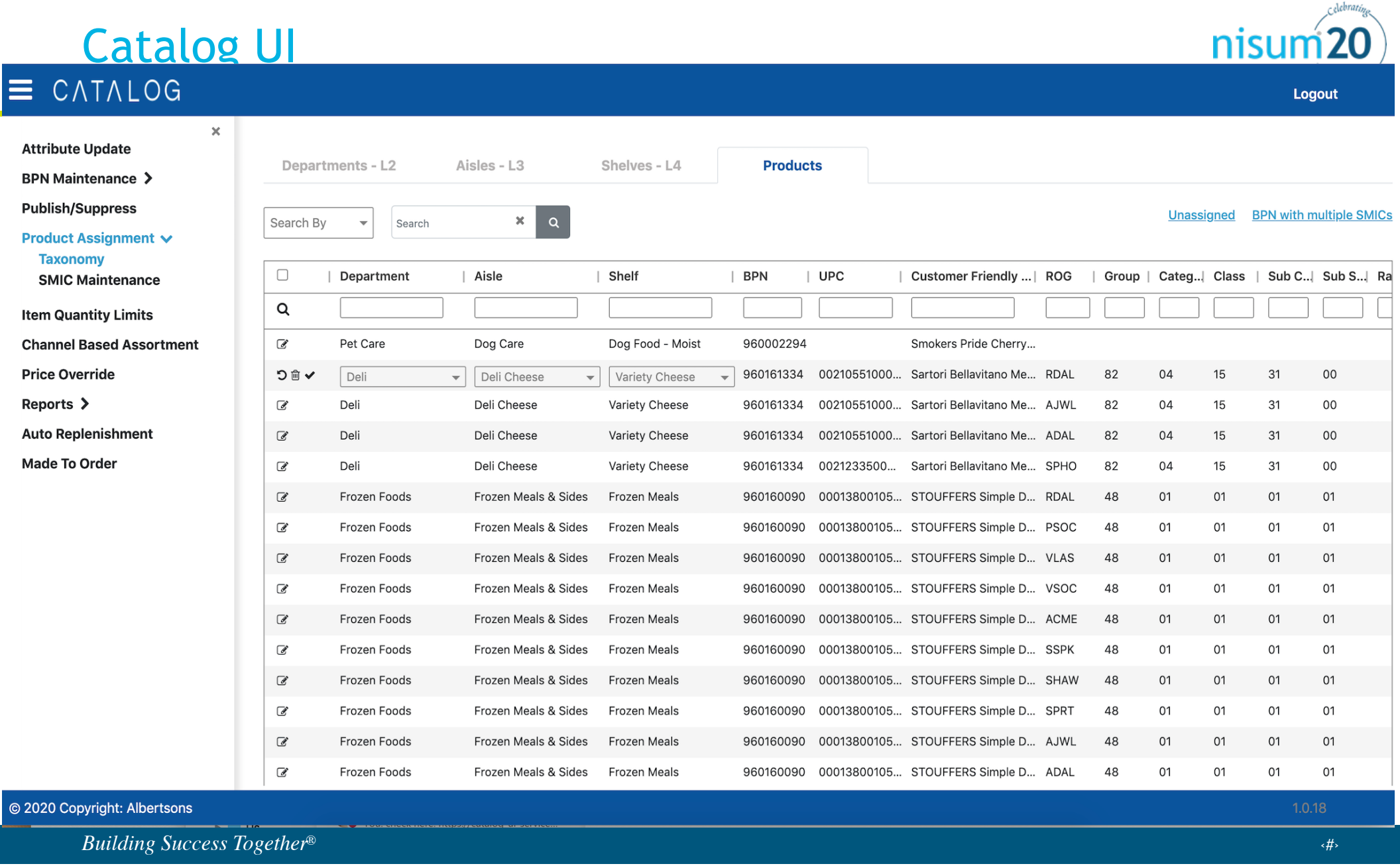
With every api call we send authenticated token to the server side.

Catalog UI



1.0.18

Catalog UI


CATALOG
Logout
Attribute Update
BPN Maintenance
Publish/Suppress
Product Assignment
Taxonomy
SMIC Maintenance
Item Quantity Limits
Channel Based Assortment
Price Override
Reports
Auto Replenishment
Made To Order
Departments - L2
Aisles - L3
Shelves - L4
Products
Search By
Search
Unassigned
BPN with multiple SMICs
Department | Aisle | Shelf | BPN | UPC | Customer Friendly ... | ROG | Group | Categ... | Class | Sub C... | Sub S... | Ra
Pet Care | Dog Care | Dog Food - Moist | 960002294 | | Smokers Pride Cherry...
Deli | Deli Cheese | Variety Cheese | 960161334 | 00210551000... | Sartori Bellavitano Me... | RDAL | 82 | 04 | 15 | 31 | 00
Deli | Deli Cheese | Variety Cheese | 960161334 | 00210551000... | Sartori Bellavitano Me... | AJWL | 82 | 04 | 15 | 31 | 00
Deli | Deli Cheese | Variety Cheese | 960161334 | 00210551000... | Sartori Bellavitano Me... | ADAL | 82 | 04 | 15 | 31 | 00
Deli | Deli Cheese | Variety Cheese | 960161334 | 0021233500... | Sartori Bellavitano Me... | SPHO | 82 | 04 | 15 | 31 | 00
Frozen Foods | Frozen Meals & Sides | Frozen Meals | 960160090 | 00013800105... | STOUFFERS Simple D... | RDAL | 48 | 01 | 01 | 01 | 01
Frozen Foods | Frozen Meals & Sides | Frozen Meals | 960160090 | 00013800105... | STOUFFERS Simple D... | PSOC | 48 | 01 | 01 | 01 | 01
Frozen Foods | Frozen Meals & Sides | Frozen Meals | 960160090 | 00013800105... | STOUFFERS Simple D... | VLAS | 48 | 01 | 01 | 01 | 01
Frozen Foods | Frozen Meals & Sides | Frozen Meals | 960160090 | 00013800105... | STOUFFERS Simple D... | VSOC | 48 | 01 | 01 | 01 | 01
Frozen Foods | Frozen Meals & Sides | Frozen Meals | 960160090 | 00013800105... | STOUFFERS Simple D... | ACME | 48 | 01 | 01 | 01 | 01
Frozen Foods | Frozen Meals & Sides | Frozen Meals | 960160090 | 00013800105... | STOUFFERS Simple D... | SSPK | 48 | 01 | 01 | 01 | 01
Frozen Foods | Frozen Meals & Sides | Frozen Meals | 960160090 | 00013800105... | STOUFFERS Simple D... | SHAW | 48 | 01 | 01 | 01 | 01
Frozen Foods | Frozen Meals & Sides | Frozen Meals | 960160090 | 00013800105... | STOUFFERS Simple D... | SPRT | 48 | 01 | 01 | 01 | 01
Frozen Foods | Frozen Meals & Sides | Frozen Meals | 960160090 | 00013800105... | STOUFFERS Simple D... | AJWL | 48 | 01 | 01 | 01 | 01
Frozen Foods | Frozen Meals & Sides | Frozen Meals | 960160090 | 00013800105... | STOUFFERS Simple D... | ADAL | 48 | 01 | 01 | 01 | 01
© 2020 Copyright: Albertsons
1.0.18

Nisum Digital Services

Armine Grigoryan

Building Success Together