Angular

One of the world’s leading Frontend Framework

WHAT IS ANGULAR?

Angular is a full featured JavaScript framework created & maintained by Google and is used for building front-end applications or the front-end of a full stack applications.

Angular is very popular in large enterprise.

A NOTE ON VERSIONS

AngularJS was released in 2010. It is not recommended and should be updated to **Angular**.

Angular refers to version 2+. Right now we are on version 8 but 2-8 is the same framework with a few changes mostly under the hood.

WHY USE ANGULAR?

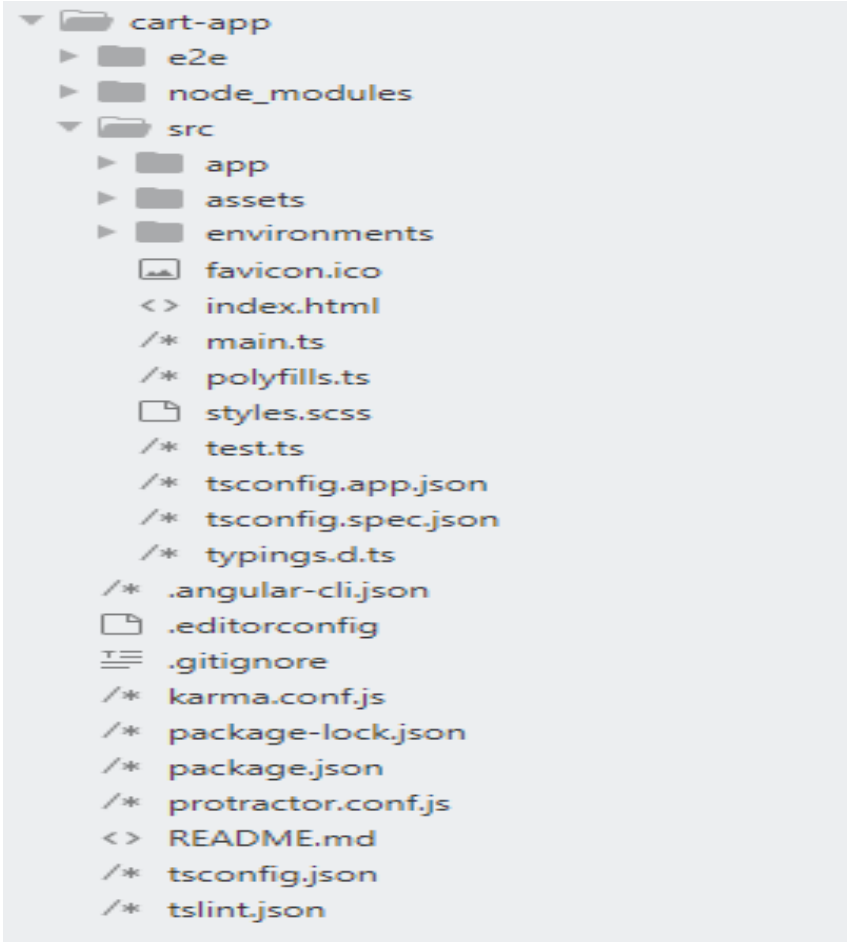
- Organized front-end structure (Components, Modules, Services)
- Extremely powerful and full featured
- All-in-one solution (Routing, HTTP, RxJS, etc.)
- Build powerful SPA apps
- MVC – Model, View, Controller design pattern
- TypeScript
- Fantastic CLI

THE ANGULAR WAY

- Uses TypeScript for static types (variables, functions, params)
- Component based (Like other frameworks)
- Uses “services” to share data/functionality between components
- Concept of “modules” (root module, forms module, http module)
- Uses RxJS “observables” for async operations
- Steep learning curve relative to other frameworks

Angular Shopping Cart

- “src” folder is the root folder
- App folder contains all the components, services and modules.
- In this app we have the following components
  - Admin
  - Cart
  - Navbar
  - Product
  - Store
- In this app only main app module is used.



STORE AND PRODUCT COMPONENTS

- The routes for store and product components are defined in **app.routing.ts**

```
const routes: Routes = [
    { path: '', component: StoreComponent },
    { path: 'store', component: StoreComponent },
    { path: 'cart', component: CartComponent },
    { path: 'product/:id', component: ProductComponent },
    { path: 'admin', component: AdminComponent },
    { path: 'blank', component: BlankComponent }
];
```

- Services: Angular services are singleton objects that are accessible and reusable throughout multiple components.
  - They can be either built in or user created
  - One of the built-in services used is **PagerService**

CONTINUE

- **Life-cycle hooks:** The components have their whole life-cycle managed by Angular, from creation to destruction. And it provides us access to life-cycle hooks, which allows us to act in key moments during the component’s life-cycle.
- One of the life-cycle hooks used is **ngOnInit()** where any initialization is performed shortly after the component is created.
- In this case, list of products are displayed depending on the page size.
- Data about the products is stored in **products.json** file.

STORE COMPONENT CODE

```
ngOnInit() {

  this.config = this.configService.config();
  this.setSaleImage();
  this.prods = this.configService.getProducts();
  this.prods.forEach((data) => {
    // if (data.carousel) {
        this.slides.push(data);
    // }
  });

  this.zebra = '';

  this.allItems = this.prods.filter(todo => ((todo.categoryname === this.params) || (this.params === '')) );
  // initialize to page 1
  this.setPage(1);
  window.scrollTo(0, 0);
```

CONTINUE

```
setPage(page: number) {
    if (page < 1 || page > this.pager.totalPages) {
        return;
    }

    let _pageSize: number;
    _pageSize = this.config.PAGE_SIZE;

    // get pager object from service
    this.pager = this.pagerService.getPager(this.allItems.length, page, _pageSize);

    // get current page of items
    this.pagedItems = this.allItems.slice(this.pager.startIndex, this.pager.endIndex + 1);

    window.scrollTo(0, 0);
}
```

```
    // return object with all pager properties required
    return {
        totalItems: totalItems,
        currentPage: currentPage,
        pageSize: pageSize,
        totalPages: totalPages,
        startPage: startPage,
        endPage: endPage,
        startIndex: startIndex,
        endIndex: endIndex,
        pages: pages
    };
}
```

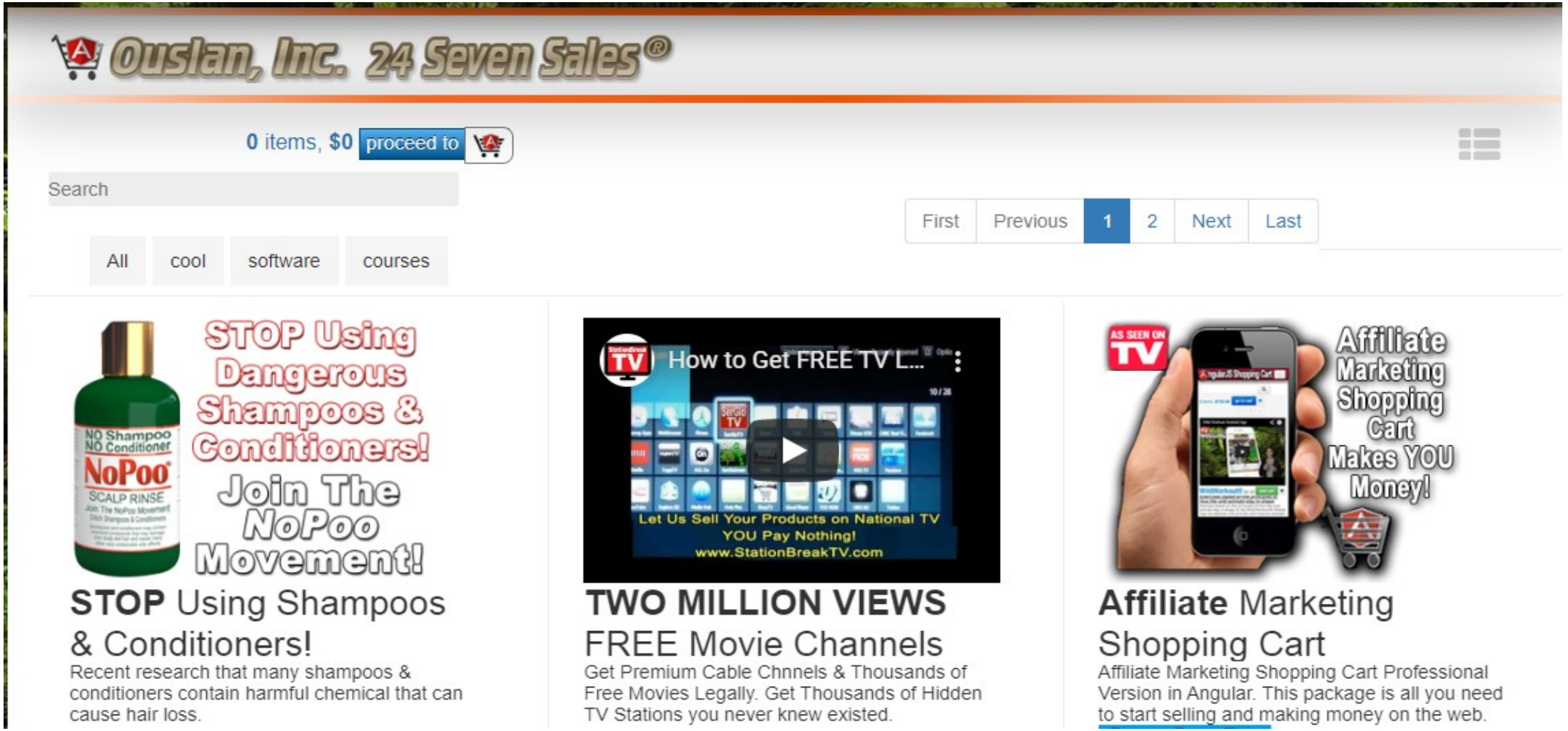
reenshot of getPager() method in pager-service.ts.

```
search(search) {
  this.zebra = search.toString();
  this.searchTerm = search;
  this.getProducts();
}

filterCategory(prop: any, value: any) {
  this.zebra = '';
  this.params = value;
  this.getProducts();
}
```

Screenshot of search() and filterCategory() methods in store.component.ts.

SCREENSHOT OF WEBSITE WITH STORE COMPONENT



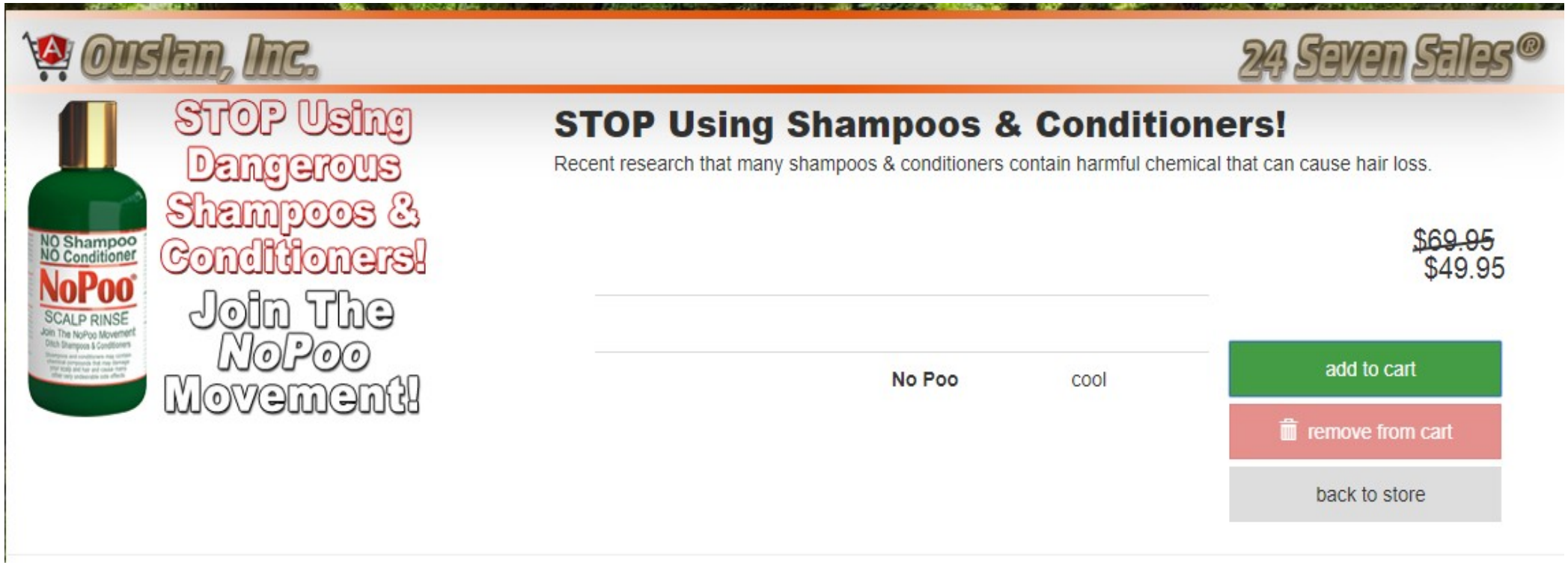
PRODUCT COMPONENT

- When the user clicks on the particular product, it is directed to **product.component.ts**
- The routing link is **/product/:item.sku**
- **getProducts()** filters out the particular product
- The details of the product are then displayed to the page

```
getProducts() {
  this.prods = (this.configService.getProducts()).filter(prods => ((prods.sku === this.params) || (this.params === '')) );
  window.scrollTo(0, 0);
  // this.dataObservableService
```

```
// get URL parameters via route
this.sub = this.route
  .params
  .subscribe(params => {
    this.params = params['id'];
    this.getProducts();
});
```

SCREENSHOT OF THE WEBSITE WITH PRODUCT COMPONENT



SHOPPING CART

CART SERVICE CODE

- **ADDING AND REMOVING ITEMS:**
- addItem(sku, name, price, quantity)
- This method adds or removes items from the cart.
- If the cart already contains items with the given sku, then the quantity of that item is modified. If the quantity reaches zero, the item is automatically removed from the cart.
- If the cart does not contain items with the given sku, then a new item is created and added to the cart using the specified sku, name, price, and quantity.
- After the cart has been updated, it is automatically saved to local storage.

```
addItem(sku, productname, unitprice,
    saleprice, showsale, quantity, sh, faux) {
    const _return = true;
    if (!faux) {
        let found = false;
        for (let i = 0; i < this.items.length && !found; i++) {
            const item = this.items[i];
            if (item.sku === sku) {
                found = true;
                item.quantity = this.toNumber(item.quantity + quantity);
                if (item.quantity <= 0) {
                    this.items.splice(i, 1);
                }
            }
        }
        // if item wasn't already in cart, add it now
        if (!found) {
            const item = new this.cartItem(sku, productname, unitprice, saleprice, showsale, quantity, sh, faux);
            this.items.push(item);
        }
        // save changes
        this.saveItems();
    } else {
        alert('This product is shown for demonstration purposes only!');
    }
    return _return;
}
```

- **GETTING TOTAL PRICE OF CART ITEMS:**
- getTotalPrice([sku])
- This method gets the total price (unit price * quantity) for one or all items in the cart.
- If the sku is provided, then the method returns the price of items with that sku. It the sku is omitted, then the method returns the total price of all items in the cart.

```
// get the total price for all items currently in the cart
getTotalPrice(sku) {
    let total: any;
    total = 0;
    for (let i = 0; i < this.items.length; i++) {
        const item = this.items[i];
        if (sku === null || item.sku === sku) {
            if (item.showsale) {
                total += this.toNumber(item.quantity * item.saleprice);
            } else {
                total += this.toNumber(item.quantity * item.unitprice);
            }
        }
    }
    return total;
}
```

- **GETTING TOTAL QUANTITY OF CART ITEMS:**
- getTotalCount([sku])
- This method gets the quantity of items or a given type or for all items in the cart.
- If the sku is provided, then the method returns the quantity of items with that sku. It the sku is omitted, then the method returns the quantity of all items in the cart.

```
// get the total price for all items currently in the cart
getTotalCount(sku) {
    let count: any;
    count = 0;
    for (let i = 0; i < this.items.length; i++) {
        const item = this.items[i];
        if (sku === null || item.sku === sku) {
            count += this.toNumber(item.quantity);
        }
    }
    return count;
}
```

- **CLEARING THE CART:**
- clearItems()
- This method clears the cart by removing all items. It also saves the empty cart to local storage.

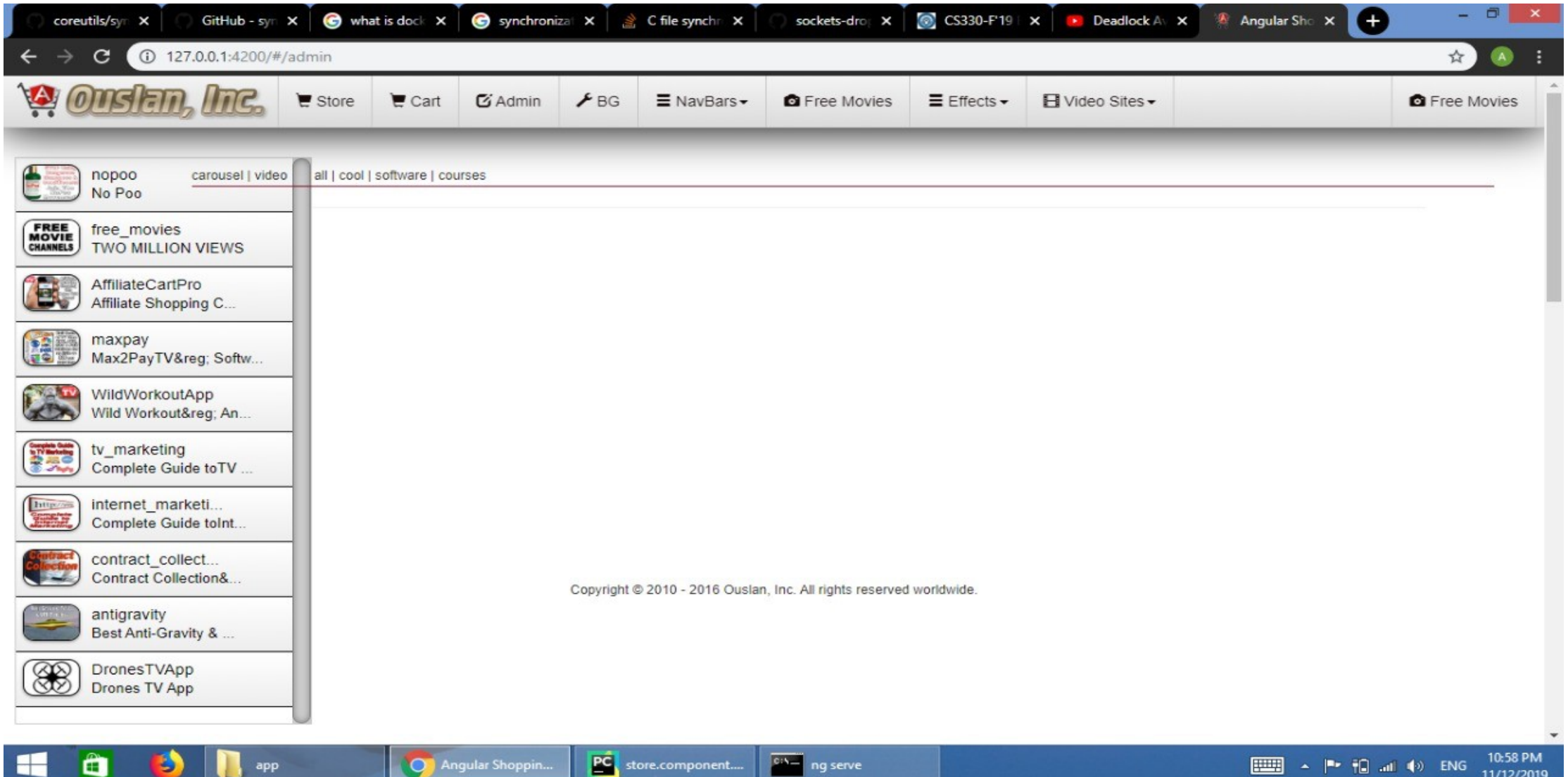
```
// clear the cart
clearItems() {
    this.items = [];
    this.saveItems();
}
```

- **CHECKOUT OPTIONS:**
- checkoutParameters(serviceName, merchantID, [options])
- This method defines a set of checkout parameters.
- The serviceName parameter defines the name of the payment provider to use, either “PayPal” or “Google”.
- The merchantID parameter specifies the merchant account associated with the service.
- The options parameter defines additional provider-specific fields. For example, we used this parameter to specify custom shipping methods

```
// checkout parameters (one per supported payment service)
//
checkoutParameters(serviceName, merchantID, options) {
    this.serviceName = serviceName;
    this.merchantID = merchantID;
    this.options = options;
    return this;
}
```

Admin, Navbar and Product Components

Admin:



Code:

admin.component.html

```
<div id="sidebar" auto-active class="scrollbar">
    <ul id="nav" data-role="listview" data-inset="true" class="ui-listview zgeneral" onMouseOver="this.style.cursor='pointer'">
        <li *ngFor="let product of allItems | orderBy:'sortorder'" onMouseOver="this.style.cursor='pointer'"
            class="sidebarlist" data-icon="false" data-role="listview"
            style="width:100%;display:block;">
            <a class="rsslistfetch-link" (click)="loadProduct(product)">
                <img class="rounded-img" src="{{config.PRODUCTS_FOLDER}}/{{product.imagename}}">
                <div class="ellipsis">   <span class="zebra1">{{ product.sku | limitTo:16 }}</span></div>
                <div class="ellipsis">   <span class="zebra2">{{ product.productname | html2Text | limitTo:20 }}</span></div>
            </a>
        </li>
        <!--<div style="height:1200px;"></div>-->
    </ul>
</div>

<div id="content">
    <div>
        <span class="carouselFilterClass" onMouseOver="this.style.cursor='pointer'" filterTextClass (click)="carouselFilter()">carousel</
        span> |
        <span class="showvideoFilterClass" onMouseOver="this.style.cursor='pointer'" (click)="showvideoFilter()">video</span>
             
        <span onMouseOver="this.style.cursor='pointer'" (click)="setFilter('all')">all</span>
        <span *ngFor="let product of prods | unique: 'categoryname'" style="white-space:nowrap !important;">
            | <span onMouseOver="this.style.cursor='pointer'" (click)="setFilter(product.categoryname)">{{product.categoryname}}</span>
        </span>
    </div>
    <div style="border-top: 1px solid #8f0222;margin:1px 0 2px 0 !important;"></div>



    <!--////////////////////////////////////////////-->
    <div *ngFor="let product of items">
        <table cellpadding="0" cellspacing="0" style="width:100%;max-height:140px !important;">
            <tr style="padding: 0 0 0 0;max-height:140px !important;">
                <td style="vertical-align: top; text-align: left;width: 250px !important;max-width: 250px !important;">
                    <div class="store_img_video" style="max-height: 140px !important;">
```

Navbar:

Code:

navbar.component.html

```
<!-- Fixed navbar -->
<nav class="navbar yamm navbar-custom navbar-fixed-top">
  <div class="container-fluid">
    <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        </button>
        <div style="display:inline !important;">
            <a class="your-brand" href="{{config.NAVBAR_LOGO_LINK}}" target="_blank">
                <img src="{{config.NAVBAR_LOGO_IMAGE}}" alt="logo" style="padding-top: 6px !important;" />
            </a>
        </div>
    </div>

    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
      <ul class="nav navbar-nav">
        <!--An elegant way to set the active tab is to use [ngClass]="{ active: isActive('/store')}" in the view as shown below:-->
        <li [ngClass]="{ active: isActive('/store')}" id="store"><a (click)="callRoute('/store')"><span class="glyphicon
        glyphicon-shopping-cart fa-lg"></span> Store</a></li>
        <li [ngClass]="{ active: isActive('/cart')}" id="cart"><a (click)="callRoute('/cart')"><span class="glyphicon
        glyphicon-shopping-cart fa-lg"></span> Cart</a></li>
        <li class="{ active: isActive('/admin-welcome')}" id="admin"><a href="admin.html#/admin"><span class="glyphicon glyphicon-edit fa-lg
        "></span> Admin</a></li>
        <!-- <li (click)="showHideCarousel($event)"><a href="#"><span class="glyphicon glyphicon-transfer fa-lg"></span> Carousel
        </a></li> -->
```

STORE VIEW

- Go to cart
- Page slider
- Search bar
- Filter results
- Pages items
  - Playing embed videos in angular 5

```
checkProducts(prods: Product[]) {
    if (prods) {
        prods.forEach((data) => {
            // Get embed format for tube servers like youtube, vimeo, youku, etc.
            data.link  = this.getVideoEmbed(data.tube, data.videoid);
            data.videopage =
this.sanitizer.bypassSecurityTrustResourceUrl(data.link);
        });
    };
}
```

StoreComponent.html
getProducts()

```
<td style="text-align: left !important;background-color: transparent !important;">
    <span *ngIf="item.link.length > 0 && item.unitprice.toString() === '0'">
        <a href="{{item.link}}" target="_blank"><span class="btn btn-x-blue">{{item.linktext}}</span><
    </span>
    <!--routerLink will handle 'href' whereas (click) doesn't-->
    <a [hidden]="item.buynow" [routerLink]="['/product', item.sku]">
        <div class="btn btn-x-blue">Info</div>
    </a>
    <a [hidden]="!item.buynow" (click)="cartService.buyNow(item)" autoscroll>
        <div class="btn btn-x-danger">BuyNow</div>
    </a>
    <a [hidden]="item.buynow" *ngIf="item.unitprice > 0" (click)="cartService.addItem(item.sku, item.p
        <div class="btn btn-x-blue">+</div>
    </a>
    <button *ngIf="cartService.getTotalCount(item.sku) > 0" class="btn btn-x-blue" type="button" style
        [disabled]="item.quantity < 1"
        (click)="cartService.addItem(item.sku, item.productname, item.unitprice, item.saleprice, item.
</td>
<td [hidden]="item.unitprice.toString() !== '0'"><img src="{{config.APP_IMAGES}}/free.png" /></td>
<td [hidden]="item.unitprice.toString() === '0'" class="content_price" style="margin: 0px 30px 0px 0px
```

Ouslan, Inc. 24 Seven Sales®

0 items, $0 proceed to

Search

First Previous 1 2 Next Last

All cool software courses

STOP Using Shampoos & Conditioners!

Recent research that many shampoos & conditioners contain harmful chemical that can cause hair loss.

TWO MILLION VIEWS FREE Movie Channels

Get Premium Cable Chnnels & Thousands of Free Movies Legally. Get Thousands of Hidden TV Stations you never knew existed.

Affiliate Marketing Shopping Cart

Affiliate Marketing Shopping Cart Professional Version in Angular. This package is all you need to start selling and making money on the web.

CART VIEW

- Empty cart message
- Cart items
- Pricing and shipping (sale totals, shipping total, total amount)
- Back to store/ empty cart buttons
- checkout


Shopping Cart with /
localhost:1234/default.htm#/cart
SAP
SocialMap dash...
BNY Mellon
Invexplorer
Google Analytics
Facebook for W...
Facebook Devel...
Other bookmarks
Angular Store
Thanks for shopping at the Angular Store.
This is your shopping cart. Here you can edit the items, go back to the store, clear the cart, or check out.
Item
Quantity
Price
Apple
2
$24.00
Pineapple
1
$4.00
Total
3
$28.00
back to store
clear cart
check out using PayPal
check out using Google
Check out with PayPal
The safer, easier way to pay
Buy with Google

CART ITEMS

```
<td style="text-align: left !important;background-color: transparent !important;">
    <span *ngIf="item.link.length > 0 && item.unitprice.toString() === '0'">
        <a href="{{item.link}}" target="_blank"><span class="btn btn-x-blue">{{item.linktext}}</span><
    </span>
    <!--routerLink will handle 'href' whereas (click) doesn't-->
    <a [hidden]="item.buynow" [routerLink]="['/product', item.sku]">
        <div class="btn btn-x-blue">Info</div>
    </a>
    <a [hidden]="!item.buynow" (click)="cartService.buyNow(item)" autoscroll>
        <div class="btn btn-x-danger">BuyNow</div>
    </a>
    <a [hidden]="item.buynow" *ngIf="item.unitprice > 0" (click)="cartService.addItem(item.sku, item.p
        <div class="btn btn-x-blue">+</div>
    </a>
    <button *ngIf="cartService.getTotalCount(item.sku) > 0" class="btn btn-x-blue" type="button" style
        [disabled]="item.quantity < 1"
        (click)="cartService.addItem(item.sku, item.productname, item.unitprice, item.saleprice, item.
</td>
<td [hidden]="item.unitprice.toString() !== '0'"><img src="{{config.APP_IMAGES}}/free.png" /></td>
<td [hidden]="item.unitprice.toString() === '0'" class="content_price" style="margin: 0px 30px 0px 0px
```

CART TOTALS

```
<tr class="well">
    <td><b>Sales Total</b></td>
    <td class="tdCenter"><b>{{cartService.getTotalCount(null)}}</b></td>
    <td class="tdRight"><b>{{cartService.getTotalPrice(null) | currency:'USD':true:'1.2-2'}}</b></td>
</tr>
<tr class="well">
    <td><b>Shipping / Handling</b></td>
    <td class="tdCenter"><b>{{cartService.getTotalCount(null)}}</b></td>
    <td class="tdRight"><b>{{cartService.getTotalSH(null) | currency:'USD':true:'1.2-2'}}</b></td>
</tr>
<tr class="well">
    <td><b>Total (Sales + SH)</b></td>
    <td class="tdCenter"><b>{{cartService.getTotalCount(null)}}</b></td>
    <td class="tdRight"><b>{{cartService.getTotalPrice(null) +cartService.getTotalSH(null) | currency:
</tr>
```

CONCLUSION

- Bootstrap, typescript
- Cart supports PayPal and google wallet
- Simple markup
- Views easy to create and understand
- Lack of documentation