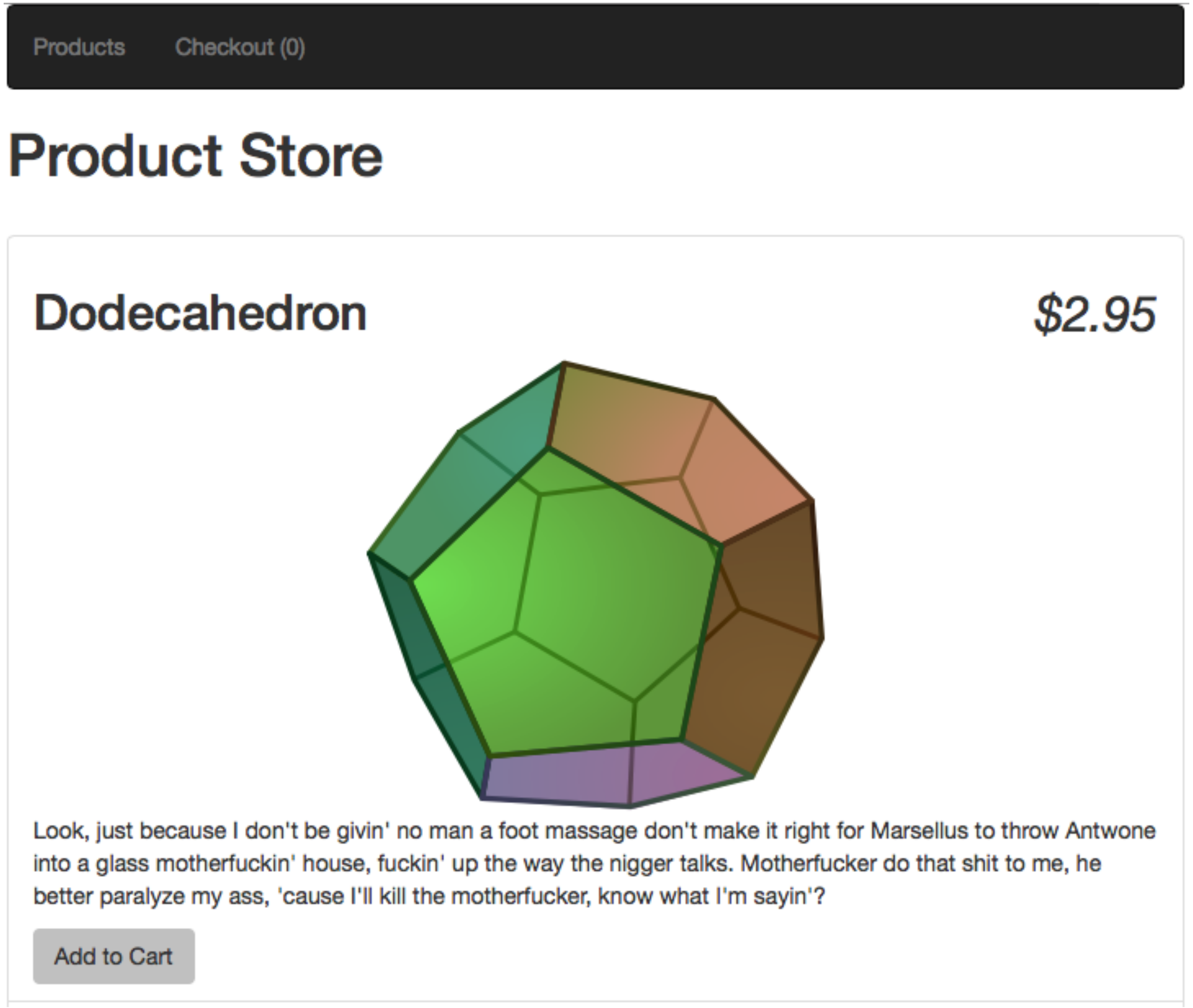
AngularJS Workshop

Let’s build a shop

Goal



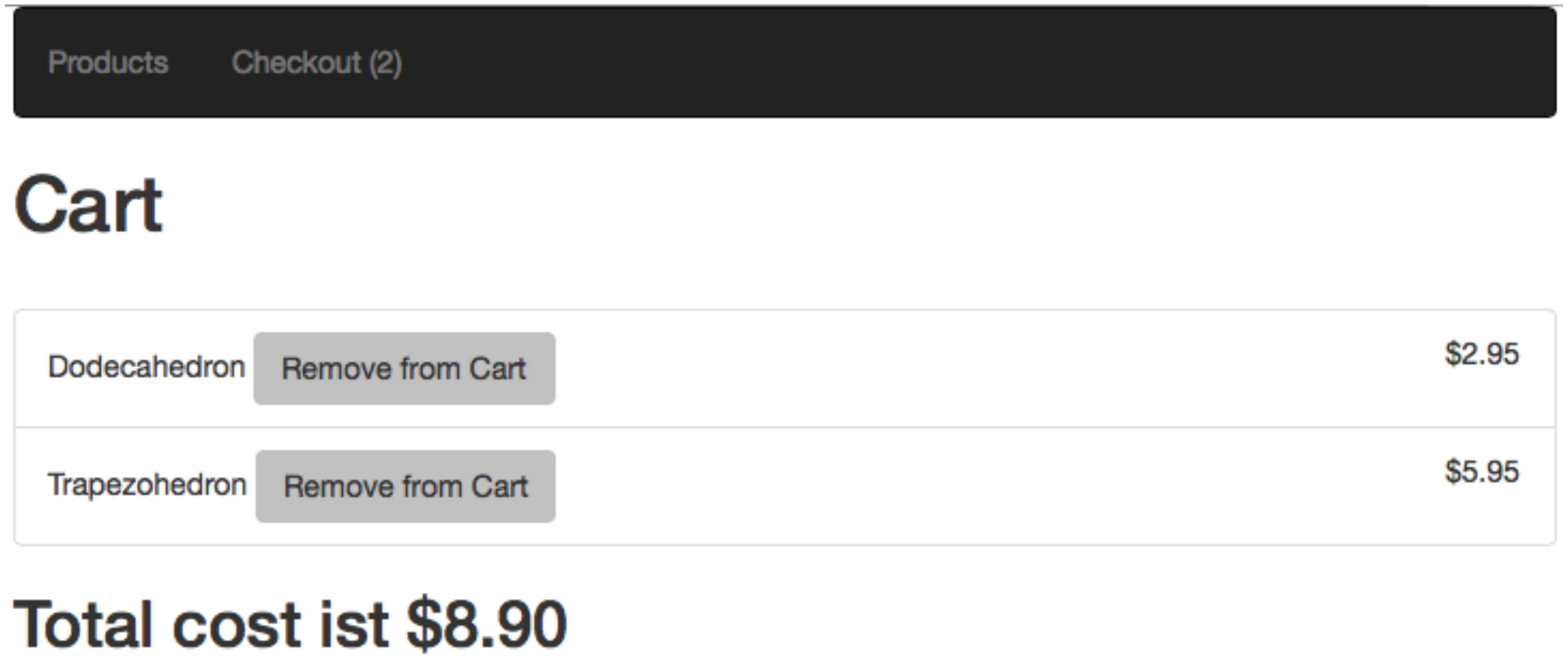
Product Store

Dodecahedron

$2.95

Look, just because I don't be givin' no man a foot massage don't make it right for Marsellus to throw Antwone into a glass motherfuckin' house, fuckin' up the way the nigger talks. Motherfucker do that shit to me, he better paralyze my ass, 'cause I'll kill the motherfucker, know what I'm sayin'?

Goal



Cart

Dodecahedron	Remove from Cart	$2.95
Trapezohedron	Remove from Cart	$5.95

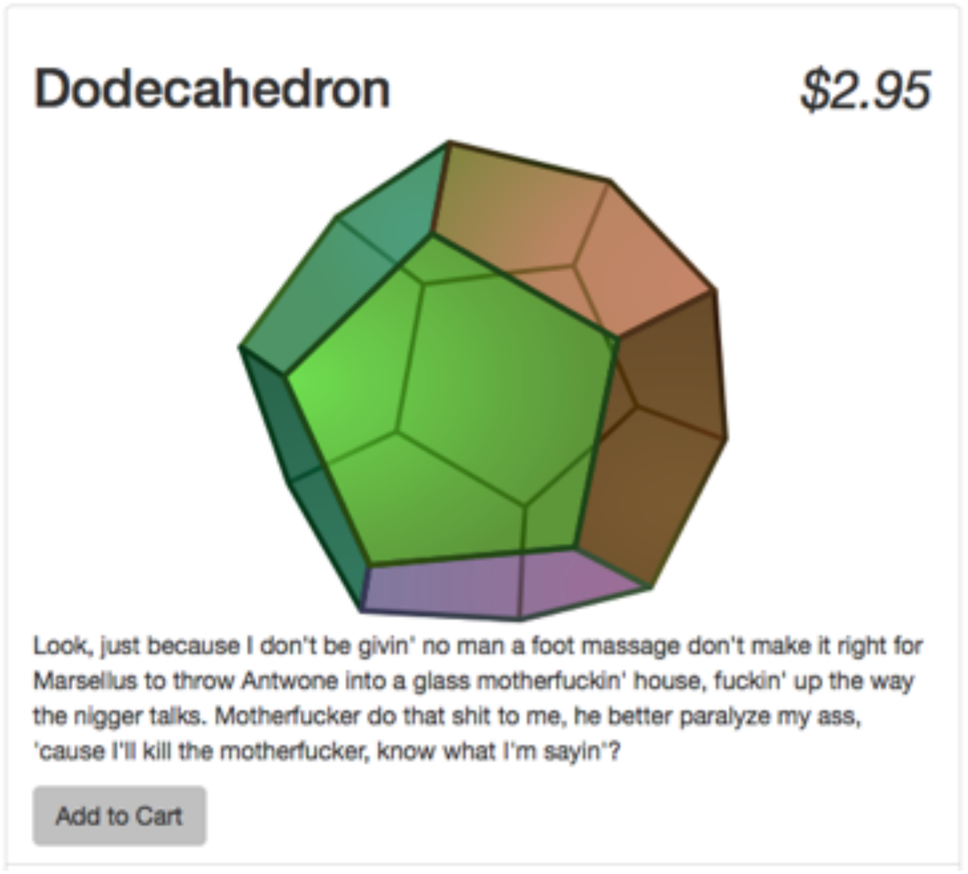
Total cost ist $8.90

Step 1: Controllers

- Look at index.html and app.js
- create a module called *store*
- create a controller called *StoreController*
- connect *store* and *StoreController* to the view

Step 2: Repeaters

- Display the products from the given array in your shop
- use the *ng-repeat* directive & *$scope* to pass it to the view
- only show the product if *canPurchase* is true
  - **Hint** use the filter called *filter*
- helpfull bootstrap classes:
  - list-group
  - list-group-item
  - pull-right

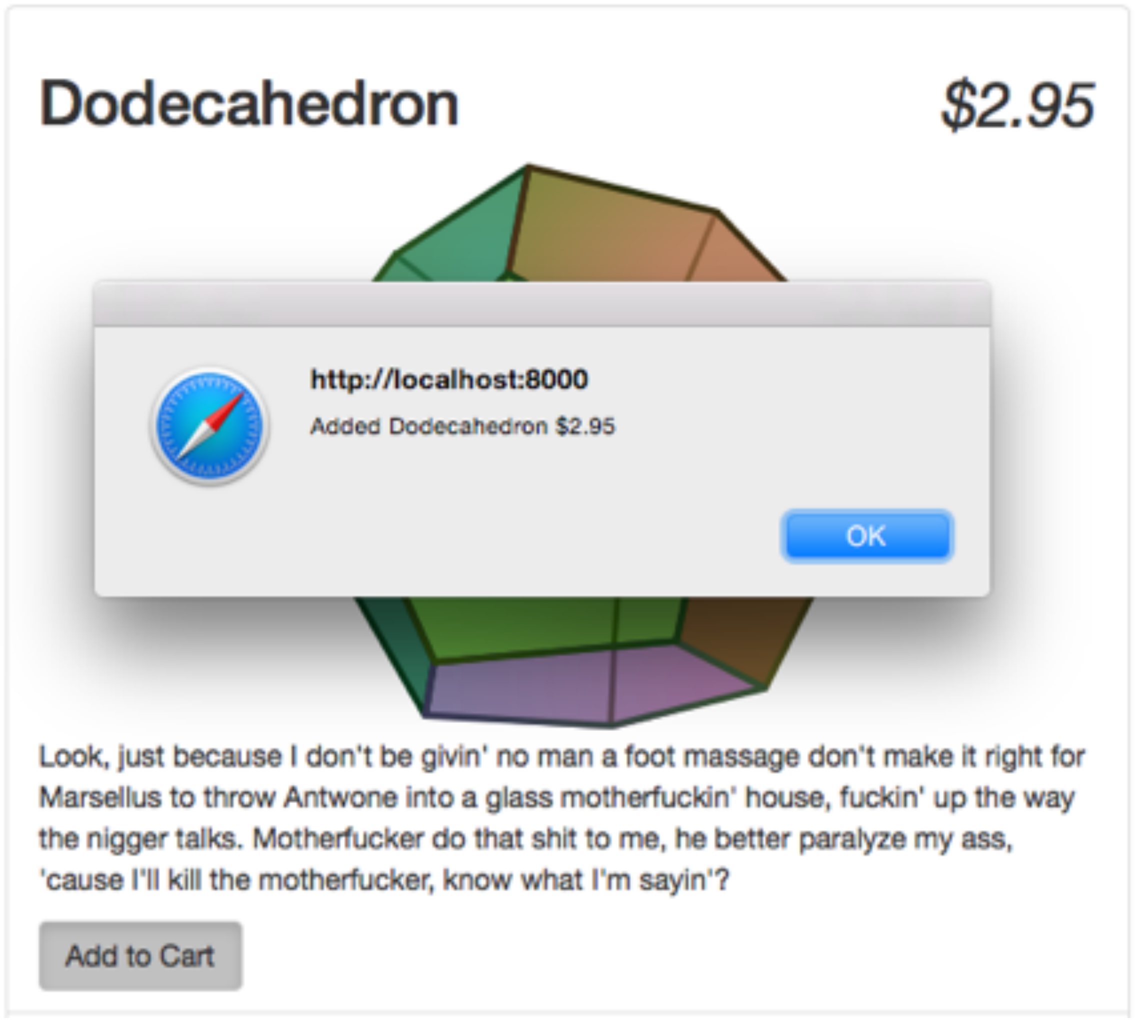


Dodecahedron *$2.95*

Look, just because I don't be givin' no man a foot massage don't make it right for Marsellus to throw Antwone into a glass motherfuckin' house, fuckin' up the way the nigger talks. Motherfucker do that shit to me, he better paralyze my ass, 'cause I'll kill the motherfucker, know what I'm sayin'?

Step 3: Actions & filters

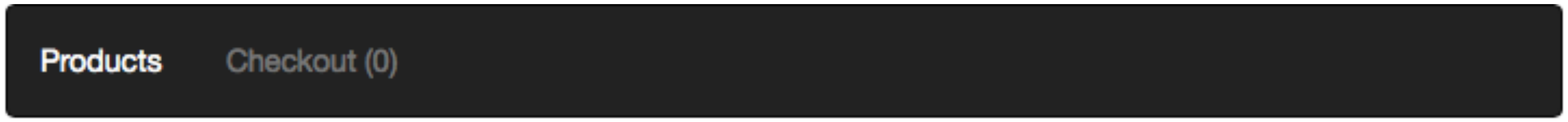
- When you click on „Add to Cart“ it should alert the name & price of the product
- format the price with the filter „currency“:
    - once in the view (top-right)
    - once in the on-click function for the alert message



Added Dodecahedron $2.95

Look, just because I don't be givin' no man a foot massage don't make it right for Marsellus to throw Antwone into a glass motherfuckin' house, fuckin' up the way the nigger talks. Motherfucker do that shit to me, he better paralyze my ass, 'cause I'll kill the motherfucker, know what I'm sayin'?

Step 4: build a navigation



- use *ui-router*
- create 2 html templates
    - *partials/store.html* (move your product list here)
    - *partials/checkout.html*
- configure 2 routes for these 2 templates
    - */products*
    - */checkout*
- **Hints on the next page**

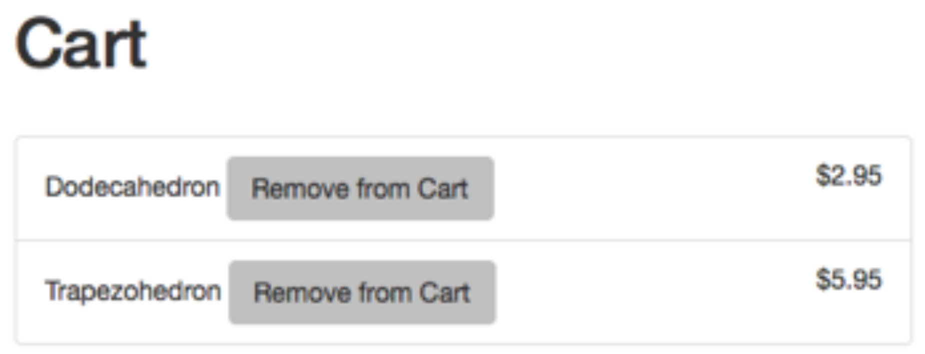
Step 4: hints

- this is where the templates from partials/*.html will be included
    - *<div ui-view></div>*
- Don’t forget to require ui-router in your module
    - *….. module(‚store‘, [‚ui.router‘]);*
- Configure ui-router inside an *app.config* block
    - *app.config(function($stateProvider, $urlRouterProvider) { …. something … });*

Step 5: build a cart Service

- create a Service called *cartService*
  - it holds the current cart items in an array
  - it has 4 methods
    - *addItem(item)*
    - *getItems()*
    - *removeItems(index)*
    - *getSize()*
- connect the „Add to Cart“ button to your service
  - remove the alert()

Step 6: display the cart in checkout



Cart

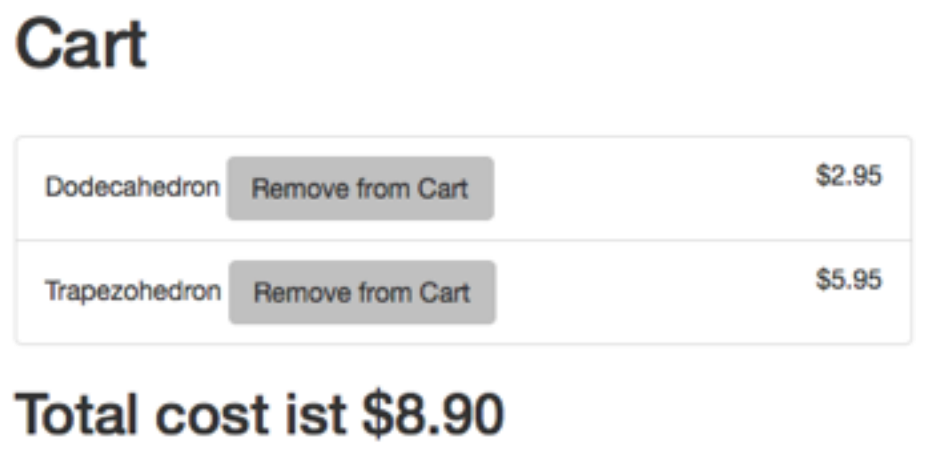
Dodecahedron	Remove from Cart	$2.95
Trapezohedron	Remove from Cart	$5.95

- create a new controller *CheckoutController*
- use the CartService to read out the items
- add a button to remove an item using it's index in the cart

Hint: use *track by $index* in the repeater to avoid errors

Step 7: make it nicer

- write your own filter to calculate the total
    - chain it with the *currency* filter to display the currency
- only show total if there are products in the cart
- show a message if no products are in the cart



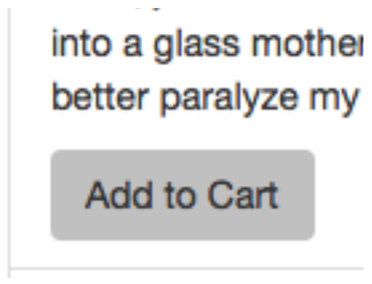
Cart

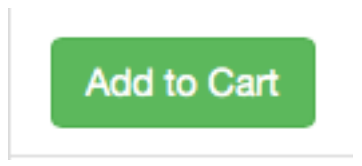
Dodecahedron	Remove from Cart	$2.95
Trapezohedron	Remove from Cart	$5.95

Total cost ist $8.90

Step 8: build a directive

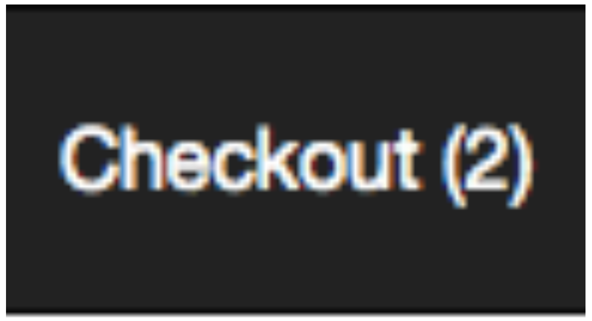
- replace the button with a custom element directive called *addToCart (<add-to-cart>)*
- after adding, the button should become green
- use the cartService from the directive
- use a template inside the directive
- **Hints**
    - you will need to pass the product through an attribute into the directive
    - green = *$element.addClass("btn-success");*





Step 9: communicate between controllers

- when an item is added or removed from the cart, the count in the navigation should update
- create a *NavigationController*
- **Hint:** you will need *$emit $on $rootScope*



Step 10: write e2e tests

- make sure you have a recent node.js installed
- Install the components
  - *npm install -g protractor*
  - *webdriver-manager update*
- Start selenium
  - *webdriver-manager start*
- Look at the test example *test.js*
- Run your tests
  - protractor conf.js
- Write additional tests for your functionality

Optional Tasks

- make your products searchable with an input field
- use real products from the migros api
- unit test your filter, service and directive