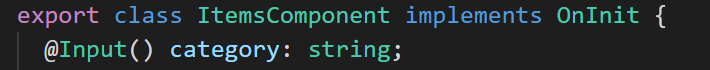
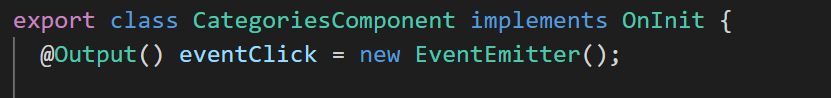
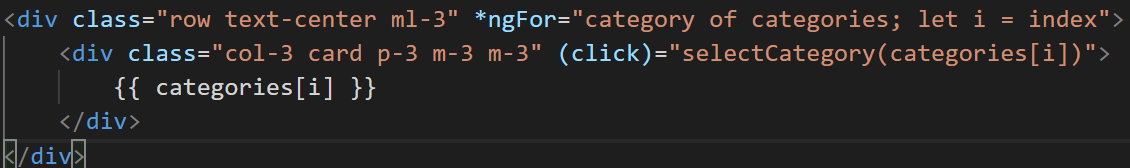
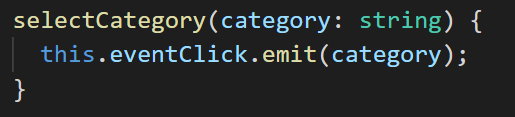
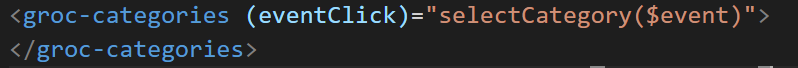
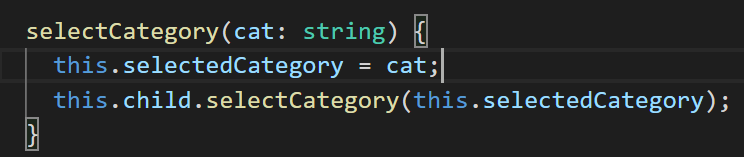
GROCERIES GUIDE

* Create a new Angular Project with the Angular (NG) CLI
  + ng new groceries
* Create as a best practice a component directory inside src folder (this also can be done for services, interfaces, directives, etc.)
  + mkdir src/app/components/
  + cd src/app/components
* Inside the created dir, generate a component named "categories" with the NG CLI. You can add prefix to overwrite the default **app**-component-name
  + ng g c categories  --prefix="groc"
  + ng g c items --prefix="groc"
* You may install Bootstrap that imports responsive design css classes like *row, text-center, col-\* (where \*'s sum up 12), etc.*
  + ng add @ng-bootstrap/ng-bootstrap
* Open VS Code to start editing the Typescript and HTML template files
  + code .
  + In categories.component.html
    - <p>Artículos que tengo que comprar:</p>
    - <div class="row text-center ml-3" \*ngFor="category of categories; let i = index">
    - <div class="col-3 card p-3 m-3 m-3">
    - {{ categories[i] }}
    - </div>
    - </div>

In [items.component.html](http://groc-items.component.html)

* + - <p>Seleccione los items comprados:</p>
    - <div class="row text-center ml-3" \*ngFor="items of selectedCategory">
    - <div class="col-3 card p-3 m-3 m-3">
    - {{ items.name }}
    - </div>
    - </div>
* Adding cats and items to the app component
  + Notice that groc-categories and groc-items components will become children of app-component
  + In [app.component.html](http://app.component.html)
    - <groc-categories (eventClick)="selectCategory($event)">
    - </groc-categories>
    - <groc-items [category]="selectedCategory"></groc-items>
* Review the code to see how @Input works
  + Notice how we set an Input variable from the app.component.html template file. Basically we send the **selectedCategory** variable from the parent to the Input variable **category** located in the child component groc-items
    - <groc-items [category]="selectedCategory"></groc-items>
  + And see how we catch this value in the items.component.ts Typescript file
    - 
  + At the same time, we reflect the value directly by doing interpolation (one-way binding) **with {{ category }}** in the items.component.html template file
    - 
    - Take a note on the \*ngIf angular directive which hides or shows an HTML element based on a condition. In this case, the <p> element will be shown only if selectedCategory
* Review how @Output works
  + First, we review the groc-categories child component, which uses two features from Angular Core library to implement actions that send data “out” to the parent: @Output will refer to the property or variable that is being sent and EventEmitter is an event binding handler
    - 
  + If we take a look in the groc-categories html template, we see that there is an event binding (in angular it goes between parenthesis inside an HTML element) that calls a method defined in groc-categories Typescript file, triggered by a **click.** 
    - 
      * Take a note on \*ngFor, which is an Angular directive to iterate through a collection (in this case, the categories array). In each iteration, the HTML elements contained inside the parent div are displayed *n* times, where *n* is the length of the categories array. Also notice there is a control autoincremented variable provided by angular named index that we can use inside our children divs.
  + The method being called by the click event uses the EventEmitter object to send data to the parent app
    - 
  + Now, see how the parent catches the @Output property via event binding with (eventClick), and how it gets the data emitted from the child by using $event
    - 
  + With this event binding, a method in the parent app Typescript file is also called: selectCategory, which ultimately receives the $event data as parameter, i.e. the category name we emitted from groc-categories
* Finally, review how @ViewChild works
  + Basically, ViewChild lets you access to the methods within a child directly from the parent. In this case the reference is to the groc-items child elements
    - In app.component.ts
      * 
  + When we explore selectCategory method, we see this
    - 
* Test it with ng serve
* So, summarizing all the interaction above, this is what the application does
  + The categories component html is displayed, allowing the user to choose one category in the groceries list
  + When the user clicks on a category, an Output variable with the category name is emitted to the parent app, which in turn
    - Defines the category input property
    - Calls a method in the child component items
  + When the selectedCategory array in the Items component TS file is filled with whatever the app component sent, then the Items html is displayed, showing only the JSON objects/items associated with the clicked category
    - This is done by using a Javascript Filter function
      * 

THE END (for now)