Angular ngx for Beginners

SoCal Code Camp @UCSD

Ogun TIGLI
otigli@gmail.com

What is it?

Angular is a client side web application framework that addresses the challenges of the single page application development process.

Angular 2+ is the new version for this popular framework which comes with many fundamental changes. This session will be an introduction to the ngx* for the beginners.

***** New generation of Angular is sometimes referred as ***Angular 2+*** or ***ngx***.

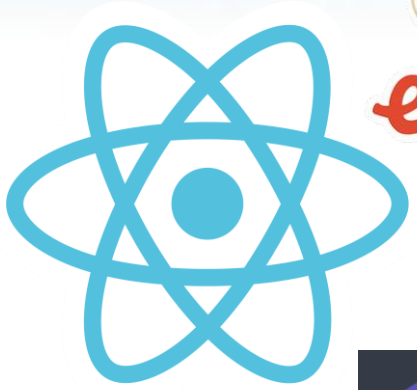
Currently in v4.2.4*

There is no version 3: Due to a misalignment in the router package's version, Angular Team decided to go straight to the version 4.

- Released on 21 June 2017

Why Angular?

VueJS



Elm

RIOT

aurelia

CycleJS

Mithril

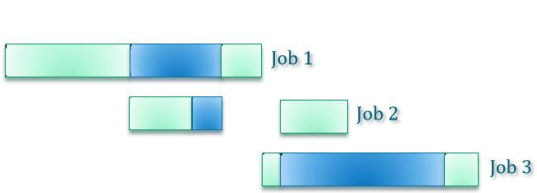
METEOR

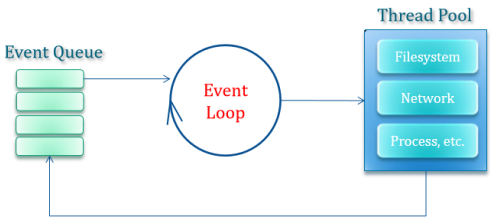
Angular & TypeScript

Angular is built with features of ES6 and ES7, Web Components in mind and written in TypeScript which is a ***super set*** of JavaScript.

As we know, JavaScript is a dynamic language:

* High-level, untyped
* Multi-paradigm, functional & object-based
* Single threaded
* Asynchronous



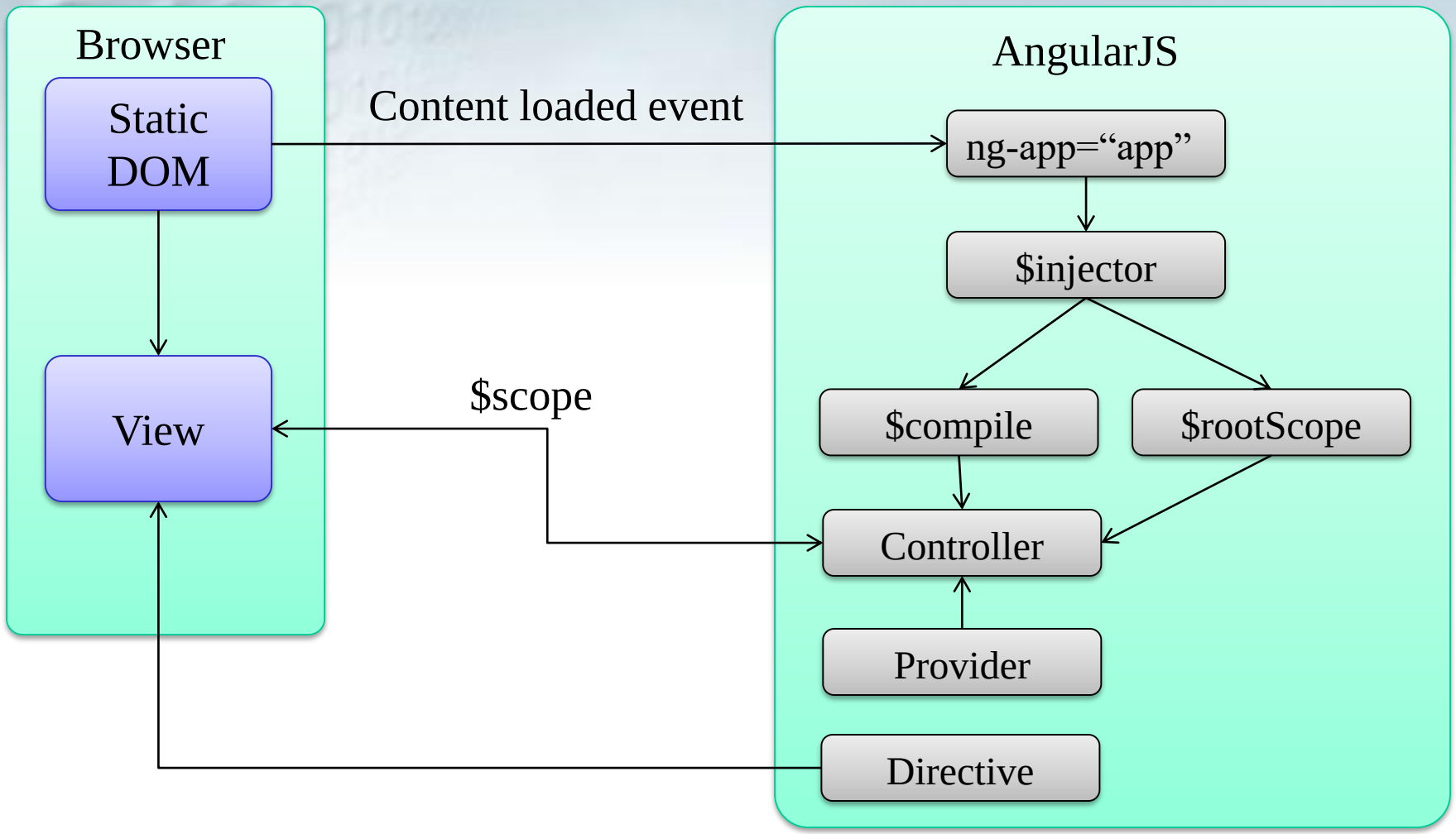


Remember Angular 1?

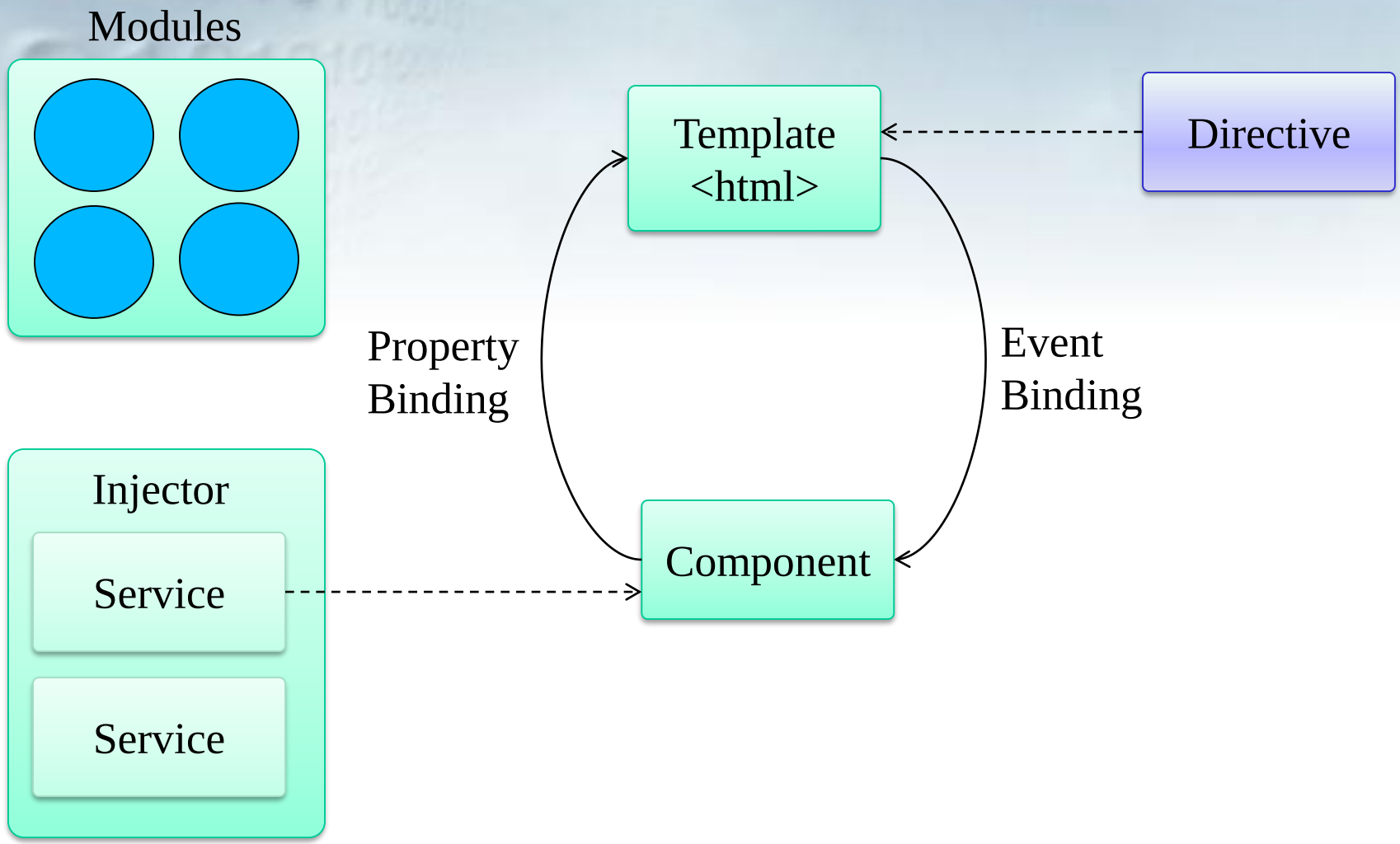
```
<html lang="en" ng-app="phonecatApp">
<body ng-controller="PhoneListCtrl">
  <ul>
    <li ng-repeat="phone in phones">
      <span>{{phone.name}}</span>
      <p>{{phone.snippet}}</p>
    </li>
  </ul>
  <script src="js/angular.js"></script>
  <script src="js/app.js"></script>
</body>
</html>
```

* From https://docs.angularjs.org/tutorial/

Angular 1 Architecture



Angular 2+ Architecture



Using Components

Components are the main building blocks for the UI

Angular 1

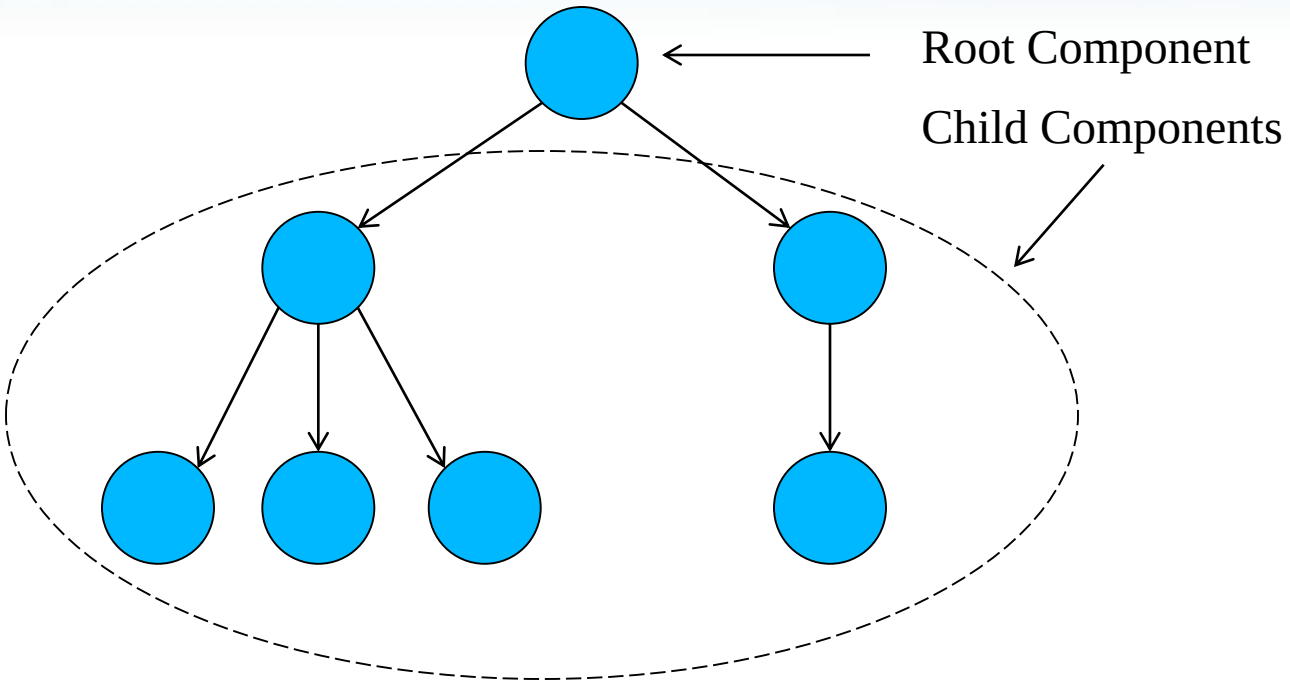
```
angular.module('app', ['some.component']);
```

Angular 2+

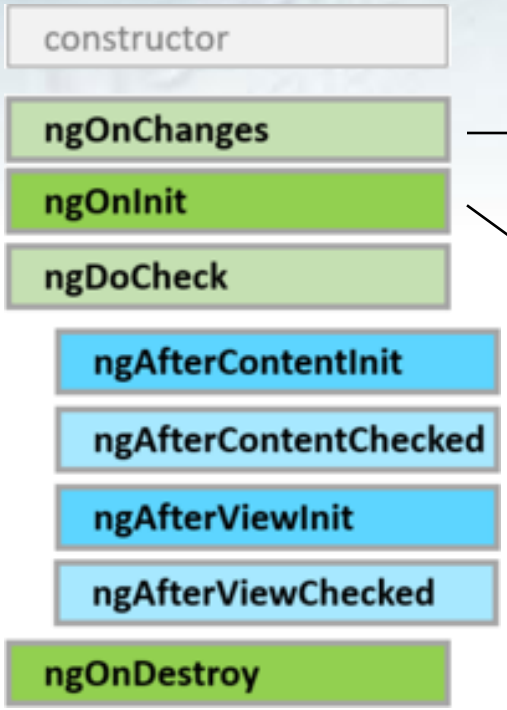
```
import {Component} from '@angular/core';
import {SomeComponent} from './some.component';
```

Components in Angular 2+

- Angular 2+ app is a tree of components
- It should have one root component (main component)
- Component is the execution context for the template



Angular 2+ Lifecycle

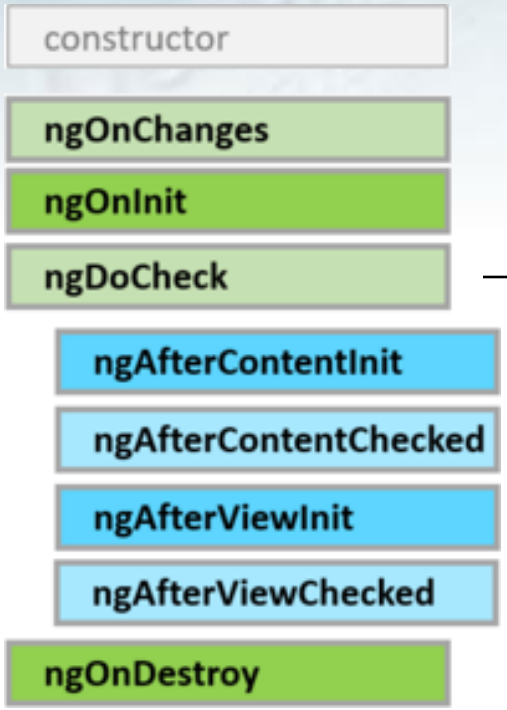


Respond when Angular (re)sets data-bound input properties. The method receives a SimpleChanges object of current and previous property values.
Called before ngOnInit() and whenever one or more data-bound input properties change.

Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties.
Called once, after the first ngOnChanges()

From angular.io/guide/lifecycle-hooks

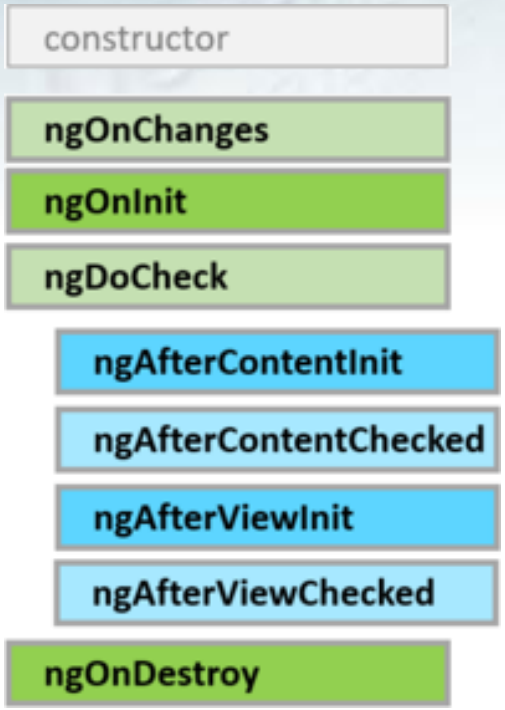
Angular 2+ Lifecycle



Detect and act upon changes that Angular can't or won't detect on its own.
Called during every change detection run, immediately after ngOnChanges() and ngOnInit().

From angular.io/guide/lifecycle-hooks

Angular 2+ Lifecycle



External content projected into the view (once)

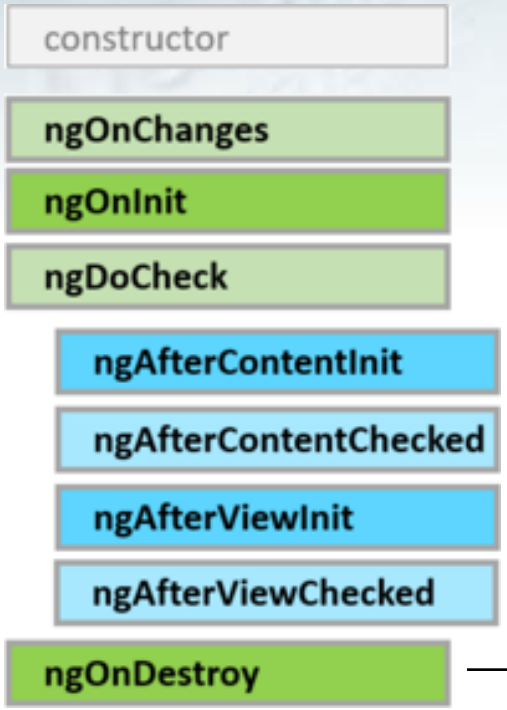
Content projected into the component (every time)

Component views and child views initialized (once)

Checked after the view initialized (every time)

From angular.io/guide/lifecycle-hooks

Angular 2+ Lifecycle



Cleanup just before Angular destroys the directive/component. Unsubscribe Observables and detach event handlers to avoid memory leaks.
Called just before Angular destroys the directive/component.

From angular.io/guide/lifecycle-hooks

Event Bindings

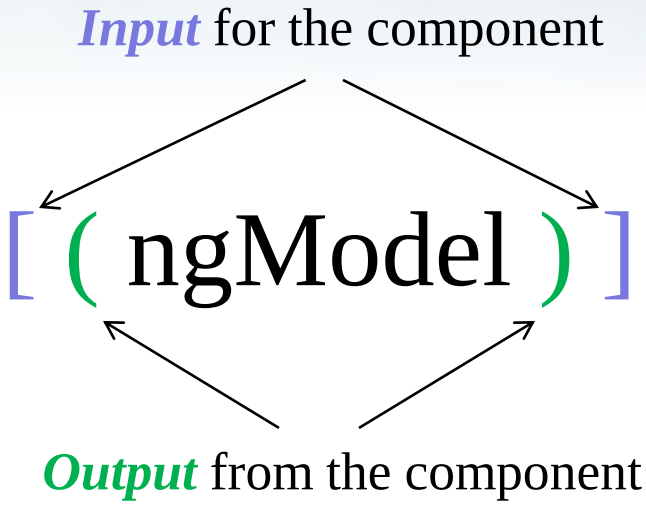
- <button (click)="“handleClick1($event)”>OK1</button>
- <button (^click)="“handleClick2($event)”>OK2</button>
  - Bubble up the event
- <button on-click="“handleClick3($event)”>OK3</button>
  - Alternative notation
- Other events are also supported like
  - doubleclick
  - keydown, keyup
  - mouseenter, mouseover, mouseleave
  - etc.

Property Binding

- Data binding
  - <some-comp [data]="myData"></some-comp>
- Class binding
  - <div [class]='some-class'>Content
- Attribute binding
  - <span [attr.someattr]='val'>Info
- Style binding
  - <div [style.color]='highlighted ? 'yellow' : 'blue' />

Two-way Data Binding

<some-comp [(ngModel)]="some.data" />



Input & Output Variables

```
<todo-item [todo]='currentTodo'
           (deleted)='handleDeletion($event)' />
```

```
export class TodoItem {
    @Input() todo: Todo;
    @Output() deleted = new EventEmitter<Todo>();
}
```

Demos

Questions ?

Questions & Answers

PS: Sample code and presentation are provided at the following GitHub page:
https://github.com/omt66/demos-socal-2017