AngularJS

http://pds25.egloos.com/pds/201208/09/76/Libera-02-Sanctus_(Based_On_Canon_By_Pachelbel).swf

왜 angularJS 인가?

- HTML은 스태틱 도큐먼트를 선언하는데 훌륭한 언어이다.
- 하지만, 웹어플리케이션에서 다이나믹 뷰를 구현하기는 힘들다.
- angularJS는 어플리케이션을 위해 HTML 구문을 확장하여, 풍부한 표현력, 가독성을 제공하고 빠르게 개발할 수 있다.

Alternatives

- 다른 프레임워크는 HTML, CSS 그리고 자바스크립트를 추상화 하거나, 돔을 다루기위한 필수적인 방법들을 제공하여 HTML의 단점을 처리한다.
- 이 중 어느것도 HTML이 다이나믹 뷰를 위해 디자인되지 않아서 생기는 root problem을 거론하지 않는다.

Extensibility

- angularJS는 어플리케이션 개발을 위한 잘 짜여진 툴셋이다.
- 다른 라이브러리와 함께 잘 동작하고 쉽게 확장가능하다.
- 특별한 개발 플로우나, 요소들을 위해 모든 요소를 수정/대체 가능하다.

The Basics

```
<!doctype html>
<html ng-app>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.3/angular.min.js"></script>
  </head>
  <body>
    <div>
      <label>Name:</label>
      <input type="text" ng-model="yourName" placeholder="Enter a name here">
      <hr>
      <h1>Hello {{yourName}}!</h1>
    </div>
  </body>
</html>
```

- ng-app
  angularJS가 적용될 위치, 예제는 모든 document를 가르킨다.
  angular.js가 로드되면 root로 ngApp directive를 보게 된다.
- ng-model
  angularJS 폼/모델 선언.
  컨트롤이 어떻게든 변하면 모델의 데이터를 업데이트한다.
  마찬가지로, 모델의 데이터가 업데이트 될 경우 컨트롤을 업데이트한다.
- {{}} (binding expression)
  데이터 바인딩.
  모델의 데이터가 변화할 경우 자동으로 업데이트 된다.

Bootstrapping

- bootstrap

automatic Initialization

- DOMContentLoaded 이벤트 or angular.js가 실행되고 document.readyState 가 'complete'가 되면...
- 초기화 작업
  1. directive로 선언된 모듈 로드
  2. 어플리케이션 injector 생성
     a. dependency injection 생성
     b. application root scope 생성
  3. DOM 컴파일(ngApp root element)
- 초기화 작업 이후
  - 모델을 변경시키는 브라우져 이벤트 옵저빙(마우스 이벤트, 키 플레스, http responce)

- 변경된 모델과 관련된 모든 바인딩을 업데이트

manual Initialization

- 문서참고

Add Some Control

Data Binding

- 데이터 바인등은 모델이 변할 때, 뷰 업데이트를 자동으로 해준다.
- 마찬가지로 뷰가 변하면 모델도 업데이트 자동으로 된다.
- 이게 끝내주는 이유는 개발자가 해야될 리스트에서 돔조작을 생각하지 않아도 되기 때문이다.

controller

- 컨트롤러는 돔 엘리먼트의 밑에서 일어나는 행위이다.
- AngularJS는 일반적인 돔조작을 위한 보일러플레이트 코드, 콜백 등록, 모델 변화를 체크할 필요없이, 단순하고 가독성 좋은 형태로 돔 조작을 표현할 수 있다.

Plain Javascript

- 다른 프래임워크와 달리, proprietary types에서 상속할 필요가 없다.
- 단순히 모델을 accessors 메소드로 감싸면 끝!
- 겁내 단순하고 친숙한 자바스크립트면 됨.
- 테스트, 유지보수, 재사용이 쉽고, 보일러플레이트 코드로 부터 해방될 수 있음.

example

index.html

```
<!doctype html>
<html ng-app>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.3/angular.min.js"></script>
    <script src="todo.js"></script>
    <link rel="stylesheet" href="todo.css">
  </head>
  <body>
    <h2>Todo</h2>
    <div ng-controller="TodoCtrl">
      <span>{{remaining()}} of {{todos.length}} remaining</span>
      [ <a href="" ng-click="archive()">archive</a> ]
      <ul class="unstyled">
        <li ng-repeat="todo in todos">
          <input type="checkbox" ng-model="todo.done">
          <span class="done-{{todo.done}}">{{todo.text}}</span>
        </li>
      </ul>
      <form ng-submit="addTodo()">
        <input type="text" ng-model="todoText"  size="30"
               placeholder="add new todo here">
        <input class="btn-primary" type="submit" value="add">
      </form>
    </div>
  </body>
</html>
```

todo.js

```
function TodoCtrl($scope) {
  $scope.todos = [
    {text:'learn angular', done:true},
    {text:'build an angular app', done:false}];

  $scope.addTodo = function() {
    $scope.todos.push({text:$scope.todoText, done:false});
    $scope.todoText = '';
  };

  $scope.remaining = function() {
    var count = 0;
    angular.forEach($scope.todos, function(todo) {
      count += todo.done ? 0 : 1;
    });
    return count;
  };

  $scope.archive = function() {
    var oldTodos = $scope.todos;
    $scope.todos = [];
    angular.forEach(oldTodos, function(todo) {
      if (!todo.done) $scope.todos.push(todo);
    });
  };
}
```

todo.css

```
.done-true {
  text-decoration: line-through;
  color: grey;
}
```

View & Template

Model & Controller

$scopre

- prototypical descendant of the root
- 템플릿, 데이터 모델, 컨트롤러의 정보를 가지고 있으며, 모델과 뷰를 싱크
- watch, digest, emit, on, broadcast 같은 메서드를 들고 있음
- http://docs.angularjs.org/api/ng.$rootScope.Scope

Wire up a Backend

Deep Linking

- deep link를 이용하여 북마킹, 이메일 링크로 앱의 특정 페이지로 바로 보낼 수 있다.
- Round trip 앱은 이러한 기능을 자동으로 처리해 주지만, Ajax 앱은 직접구현 해줘야 한다.
- angularJS는 이러한 deep link의 유익한 점이 잘 반영되어 있다.

Form Validation

- 클라이언트 측의 폼 발리데이션은 끝내주는 사용자 경험을 제공하는데 있어서 중요한 부분이다.
- AngularJs는 발리데이션 룰을 자바스크립트 코딩없이 선언할수 있게 해준다.
- 코딩은 적게, 적용은 후딱!

Server Communication

- angularJS는 고도화된 XHR 서비스를 제공하며, 코드를 드라마틱하게 간단히 짤수 있다.
- XHR을 한번 감싸서, 예외처리, 프로미스(promises)를 제공한다.
- 프로미스를 이용해 어싱크페턴으로 데이터를 반환하는 코드를 간단하게 짤 수 있다.
- 이는 반환값이 어싱크하게 들어와도, 프로퍼티를 싱크로스럽게 받을 수 있다.

example

index.html

```
<!doctype html>
<html ng-app="project">
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.3/angular.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.3/angular-resource.min.js">
    </script>
    <script src="https://cdn.firebase.com/v0/firebase.js"></script>
    <script src="http://firebase.github.io/angularFire/angularFire.js"></script>
    <script src="project.js"></script>
  </head>
  <body>
    <h2>JavaScript Projects</h2>
    <div ng-view></div>
  </body>
</html>
```

project.js

```
angular.module('project', ['ngRoute', 'firebase'])

.value('fbURL', 'https://angularjs-projects.firebaseio.com/')

.factory('Projects', function(angularFireCollection, fbURL) {
  return angularFireCollection(fbURL);
})
```

```
.config(function($routeProvider) {
  $routeProvider
    .when('/', {
      controller:'ListCtrl',
      templateUrl:'list.html'
    })
    .when('/edit/:projectId', {
      controller:'EditCtrl',
      templateUrl:'detail.html'
    })
    .when('/new', {
      controller:'CreateCtrl',
      templateUrl:'detail.html'
    })
    .otherwise({
      redirectTo:'/'
    });
})

.controller('ListCtrl', function($scope, Projects) {
  $scope.projects = Projects;
})

.controller('CreateCtrl', function($scope, $location, $timeout, Projects) {
  $scope.save = function() {
    Projects.add($scope.project, function() {
      $timeout(function() { $location.path('/'); });
    });
  };
})

.controller('EditCtrl',
  function($scope, $location, $routeParams, angularFire, fbURL) {

    var projectUrl = fbURL + $routeParams.projectId;
    var bindToProject = angularFire(projectUrl, $scope, 'remote', {});

    bindToProject.then(function() {

      $scope.project = angular.copy($scope.remote);
      $scope.project.$id = $routeParams.projectId;

      $scope.isClean = function() {
        return angular.equals($scope.remote, $scope.project);
      }

      $scope.destroy = function() {
        $scope.remote = null;
        $location.path('/');
      };

      $scope.save = function() {
        $scope.remote = angular.copy($scope.project);
        $location.path('/');
      };
```

```
    });
});
```

list.html

```
<input type="text" ng-model="search" class="search-query" placeholder="Search">
<table>
  <thead>
  <tr>
    <th>Project</th>
    <th>Description</th>
    <th><a href="#/new"><i class="icon-plus-sign"></i></a></th>
  </tr>
  </thead>
  <tbody>
  <tr ng-repeat="project in projects | filter:search | orderBy:'name'">
    <td><a href="{{project.site}}" target="_blank">{{project.name}}</a></td>
    <td>{{project.description}}</td>
    <td>
      <a href="#/edit/{{project.$id}}"><i class="icon-pencil"></i></a>
    </td>
  </tr>
  </tbody>
</table>
```

detail.html

```
<form name="myForm">
  <div class="control-group" ng-class="{error: myForm.name.$invalid}">
    <label>Name</label>
    <input type="text" name="name" ng-model="project.name" required>
    <span ng-show="myForm.name.$error.required" class="help-inline">
        Required</span>
  </div>

  <div class="control-group" ng-class="{error: myForm.site.$invalid}">
    <label>Website</label>
    <input type="url" name="site" ng-model="project.site" required>
    <span ng-show="myForm.site.$error.required" class="help-inline">
        Required</span>
    <span ng-show="myForm.site.$error.url" class="help-inline">
        Not a URL</span>
  </div>

  <label>Description</label>
  <textarea name="description" ng-model="project.description"></textarea>

  <br>
  <a href="#/" class="btn">Cancel</a>
  <button ng-click="save()" ng-disabled="isClean() || myForm.$invalid"
          class="btn btn-primary">Save</button>
  <button ng-click="destroy()"
          ng-show="project.$id" class="btn btn-danger">Delete</button>
</form>
```

- ng-view
    - 파티셜 페이지 or 뷰 생성 directive

firebase

https://www.firebase.com/?utm_medium=web&utm_source=angularFire

Create Components

Directives

- Directives는 오직 angularJS에서만 사용가능한 파워풀한 피쳐입니다.
- 새론운 HTML 문법을 개발하여 앱에서 사용할 수 있습니다.

Reusable Components

- directives를 사용해서 재사용 가능한 컴퍼넌트를 생성할 수 있습니다.
- 새로운 컴퍼넌트를 생성함으로써, 복잡한 마크업, 행동을 감출 수 있습니다.
- 이로서, 어플리케이션의 행동과 룩을 구분하여 각자에 집중할 수 있습니다.

Localization

- 앱에서 심각한 문제중 하나는 현지화입니다.
- angularJS는 로케일 인식 필터 및 형태소 directives는 모든 지역에서 응용 프로그램을 사용할 수 있도록 빌딩 블록을 제공합니다.

index.html

```
<!doctype html>
<html ng-app="app">
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.3/angular.min.js"></script>
    <script src="components.js"></script>
    <script src="app.js"></script>
  </head>
  <body>
    <tabs>
      <pane title="Localization">
        Date: {{ '2012-04-01' | date:'fullDate' }} <br>
        Currency: {{ 123456 | currency }} <br>
        Number: {{ 98765.4321 | number }} <br>
      </pane>
      <pane title="Pluralization">
        <div ng-controller="BeerCounter">
          <div ng-repeat="beerCount in beers">
            <ng-pluralize count="beerCount" when="beerForms"></ng-pluralize>
          </div>
        </div>
      </pane>
    </tabs>
  </body>
</html>
```

components.js

```
angular.module('components', [])

 .directive('tabs', function() {
  return {
    restrict: 'E',
    transclude: true,
    scope: {},
    controller: function($scope, $element) {
      var panes = $scope.panes = [];

      $scope.select = function(pane) {
        angular.forEach(panes, function(pane) {
          pane.selected = false;
        });
        pane.selected = true;
      }

      this.addPane = function(pane) {
        if (panes.length == 0) $scope.select(pane);
        panes.push(pane);
      }
    },
    template:
      '<div class="tabbable">' +
        '<ul class="nav nav-tabs">' +
          '<li ng-repeat="pane in panes" ng-class="{active:pane.selected}">'+
            '<a href="" ng-click="select(pane)">{{pane.title}}</a>' +
          '</li>' +
        '</ul>' +
        '<div class="tab-content" ng-transclude></div>' +
      '</div>',
    replace: true
  };
 })

 .directive('pane', function() {
  return {
    require: '^tabs',
    restrict: 'E',
    transclude: true,
    scope: { title: '@' },
    link: function(scope, element, attrs, tabsCtrl) {
      tabsCtrl.addPane(scope);
    },
    template:
      '<div class="tab-pane" ng-class="{active: selected}" ng-transclude>' +
      '</div>',
    replace: true
  };
 })
```

app.js

```
angular.module('app', ['components'])

.controller('BeerCounter', function($scope, $locale) {
  $scope.beers = [0, 1, 2, 3, 4, 5, 6];
  if ($locale.id == 'en-us') {
    $scope.beerForms = {
      0: 'no beers',
      one: '{} beer',
      other: '{} beers'
    };
  } else {
    $scope.beerForms = {
      0: 'žiadne pivo',
      one: '{} pivo',
      few: '{} pivá',
      other: '{} pív'
    };
  }
});
```

Dependency Injector

Directives

- http://docs.angularjs.org/guide/directive
- http://docs.angularjs.org/api/ng.$compile

reference

- http://docs.angularjs.org