What is AngularJS
AngularJS is a Javascript MVC framework created by Google to build properly architectured and maintenable web applications.

Why AngularJS?

- Defines numerous ways to organize web application at client side.
- Enhances HTML by attaching directives, custom tags, attributes, expressions, templates within HTML.
- Encourage TDD
- Encourage MVC/MVVM design pattern   (Model View ViewModel)
- Code Reuse
- Good for Single Page Apps (SPA)
- Cool Features
    - Declarative HTML approach
    - Easy Data Binding : Two way Data Binding
    - Reusable Components
    - MVC/MVVM Design Pattern
    - Dependency Injection
    - End to end Integration Testing / Unit Testing
    - Routing
    - Templating
    - Modules
    - Services
    - Expressions
    - Filters
    - Directives
    - Form Validation
    - $scope, $http, $routeProvider…

HTML Compiler
(https://docs.angularjs.org/guide/compiler)
Angular's HTML compiler allows the developer to teach the browser new HTML syntax. The compiler allows you to attach behavior to any HTML element or attribute and even create new HTML elements or attributes with custom behavior. Angular calls these behavior extensions directives.
Compiler is an angular service which traverses the DOM looking for attributes. The compilation process happens in two phases.
Compile: traverse the DOM and collect all of the directives. The result is a linking function.
Link: combine the directives with a scope and produce a live view. Any changes in the scope model are reflected in the view, and any user interactions with the view are reflected in the scope model. This makes the scope model the single source of truth.

Modules
The structure of your application.
In AngularJS applications are structured in modules.
Inside module you create:

- controllers
- services
- filters
- directives

Use ng-app to tell AngularJS that this part of your application will be managed by module

```
var app = angular.module('budget', []);
```

Controllers
Data provider for your view
To provide data for the view, the controller can be injected with the scope of the view.

```
app.controller('MainCtrl', function ($scope) {
...
});
```

Scope
Scope is an object that refers to the application model.
Scope is used to link the controller and the views.
It is an execution context for expressions.
Scopes are arranged in hierarchical structure which mimic the DOM structure of the application (scope can have child scopes which can see the content of parent scopes).
Scopes can watch expressions and propagate events.
Actually the ViewModel of MVVM.

```
$scope
```

Bootstrap

```
// index.html
<!doctype html>
<html ng-app="budget">

<head>
    <title>Hello AngularJS</title>
    <script
src="http://code.angularjs.org/1.2.13/angular.min.js"></script>
    <script src="controllers.js"></script>
</head>
```

```
<body ng-controller="MainCtrl">

</body>
</html>

//controllers.js
var app = angular.module('budget', []);

app.controller('MainCtrl', function ($scope) {

});
```

Expression
Expressions are JavaScript-like code snippets that are usually placed in bindings such as {{ expression }}

```
  1+2={{1+2}}
```

Data Binding
Connects models and views

```
//controllers.js
$scope.sometext = "witam serdecznie";

// index.html
{{sometext}} <br />
<input type="text" ng-model="sometext" value="{{sometext}}">
```

Directive
The directives can be placed in element names, attributes, class names, as well as comments. Directives are a way to teach HTML new tricks.
A directive is just a function which executes when the compiler encounters it in the DOM.
Directive extends HTML to structure your application.

```
// controllers.js
$scope.users = [
        {name: "Ala", gender: "female", description: "Ala desc"},
        {name: "Olek", gender: "male", description: "Olek desc"}
    ];
```

ng-repeat

```
      <div ng-repeat="user in users">
        <div>
            <h3>{{user.name}} ({{user.gender}})</h3>
            {{user.description}}  <input type="text"
ng-model="user.description" placeholder="Enter a name here">
        </div>
    </div>
```

It iterates on a collection in the scope to create the DOM

ng-show

```
   <div ng-repeat="user in users" ng-show="user.gender == 'female'">
        <div>
            <h3>{{user.name}} ({{user.gender}})</h3>
            {{user.description}}  <input type="text"
ng-model="user.description" placeholder="Enter a name here">
        </div>
    </div>
```

ng-switch (check in documentation :))

Custom directives
There are a bit complex but here is a simple example:

```
// controllers.js
app.directive('mySuperCustomDirective', function(){
    return {
        template: 'Nie taki diabel straszny'
    };
});


// index.html
<div my-super-custom-directive ></div>
```

Filters
Change the ways the expressions are displayed
Usage: {{expression | filter}}

```
   <div>
        <label>Name:</label>
        <input type="text" ng-model="name" placeholder="Enter a name
here">
        <hr>
        <h1>Hello {{name | uppercase}}!</h1>
    </div>
```

Scope $watch
Angular provides a tool that observe the changes on the data of the scope from the controller
With the operation $watch a controlller can add a listener on an expression of the scope.

```
  $scope.$watch('users', function(newValue, oldValue){
        console.log("Old: ");
        console.log(oldValue);
        console.log("New:");
        console.log(newValue);
    }, true);
```

There is also possible broadcasting an events
https://docs.angularjs.org/api/ng/type/$rootScope.Scope

Forms
Form and controls provide validation services, so that the user can be notified of invalid input. This provides a better user experience, because the user gets instant feedback on how to correct the error.

```
  <form name="myForm" novalidate>  <!-- novalidate is used to disable
browser's native form validation -->
            <input type="text" name="user" ng-model="user" required>
            <span style="color:red" ng-show="myForm.user.$dirty &&
myForm.user.$invalid"><span
ng-show="myForm.user.$error.required">Username is
required.</span></span>
            <input type="submit" ng-disabled="myForm.user.$dirty &&
myForm.user.$invalid">
    </form>
```

Documentation
https://docs.angularjs.org/api/