jQuery and AngularJS

WS Slides #3

Content

- Alternative languages to JS
- Bower
- jQuery
- AngularJS
- Jasmine

JS has too many problems …

Some alternative languages (compiled to JavaScript, we don't use only for knowledge)

- CoffeScript
- TypeScript
- Dart
- …

Bower

Bower works by fetching and installing packages from all over, taking care of hunting, finding, downloading, and saving the stuff you're looking for. Bower keeps track of these packages in a manifest file, bower.json. Bower was created at Twitter.

Summary: It a client side package manager (download bundles of CSS, JS and manage dependencies between bundles)

- Dependant on other package manager npm (Node package manager). Must install (nice command: npm config list)
- Then install Bower (see source samples README-files or WWW)
- Let Bower install Bootstrap, jQuery, AngularJS, … in web app resources folder (will create folder bower_components)

NOTE: We normally only need the dist-folder from the downloaded "stuff", sometimes the package contains a lot more … just delete not needed subfolders.

- jquery source code has error intro.js file is missing a bit, which is found in outro.js …!
- .. delete src to get rid of

jQuery

1 2 3

```
$("#myList").children().each(
  function(index, element) {
    ...
    $(this).css("color", "red");
  });
```

jQuery is a JavaScript library by John Resig

- Cross browser: Trying to eliminate differences
  - No need for : "use strict", jQuery will handle
- Free, Open Source, very good documentation
- jQuery API represented by jQuery-object (shorthand $)

Features

- Better DOM, DOM Events and CSS API's
- Better AJAX API
- Effects, animations

jQuery philosophy

1. Find some DOM nodes using CSS selectors. Nodes will be "wrapped" in the jQuery object (objects enhanced with jQuery methods)
2. Do something with them using jQuery methods (instead of native JS API i.e. DOM). Chain multiple method calls to process the set of nodes
3. Don't need to explicitly traverse the set. Use implicit iteration (each()-method) to get final result. Possible to get "previous" set back

Slide Code Dissection

- jQuery will find all children to element with id "myList" and traverse the list
- Anonymous callback method passed to each(). Will execute for each child element, will set color for each child
- $(this) is the wrapped child element ("this" only is the current DOM element)

jQuery Elements

Defer execution

```
$(function () { … });
```

Find

```
var ul = $("#myList");
var children = $("#myList").children();
var parent = $("#myList").parent();
```

Add

```
$("<input id='btnClose' type='button'
                        value='Save'/>").appendTo('#popUp');
```

Modify

```
$('div.second').replaceWith('<h2>New heading</h2>');
```

Delete

```
$('.productTableBody tr').remove();
```

Find and manipulate elements

- To defer execution until DOM is ready wrap code in $ (code run when DOM ready, use to set up listeners etc.)
- Use CSS selectors to find element(s) then jQuery methods to change, add, remove, …

jQuery Attributes, CSS and Input

Attribute

```
$('#greatphoto').attr('alt');  // Get
$('#greatphoto').attr('alt', 'Beijing Brush
Seller');     // Set
```

CSS

```
$('#mydiv').css('color') // Get
$('#mydiv').css('color', 'green')  // Set
```

Get some user input

```
var value = $("#myInput").val();
```

Find and manipulate attributes, styles and input.

jQuery Events

Existing node

```
$(function(){
   $("#btn1").on("click", listener);
});
```

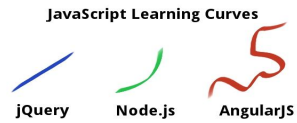
Non-existing node

```
$(function(){
  $(document).on("dblclick","#myBtn", listener);
 });
```

Event names similar to native Event API

- Skip leading on
- Setup listeners when DOM finished using $(function() { … })
- NOTE: No parentheses for listener (it's the function reference we want not executing it)

AngularJS (v 1.0)



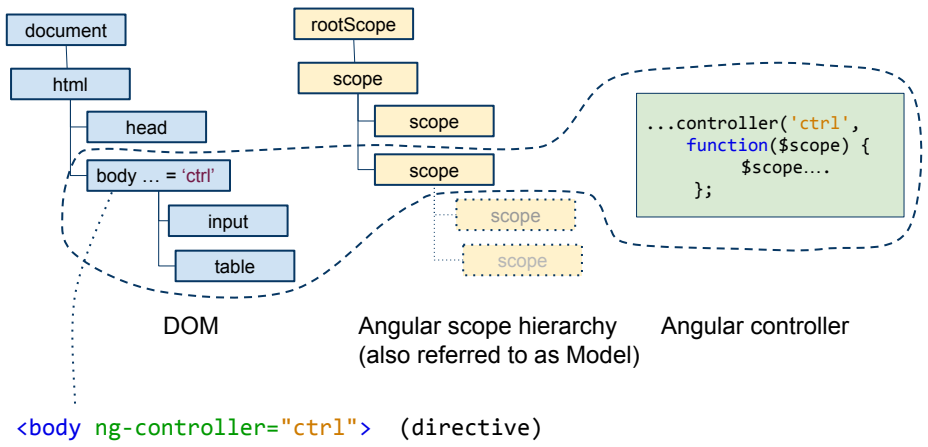
AngularJS v 1.0

- JavaScript client side MVC-similar framework developed by Google.
- We use 1.x : Conceptual Overview . Sadly AngularJS 2.x will be totally different from AngularJS 1.x and NOT backward compatible!
- Developer docs
- How to learn Angular
- Dependency on jQuery

Basically

- An MVC-like structure
- Templates, HTML enhanced with Angular attributes (smarter HTML)
- "Data binding", JS code and HTML page automatically in sync (NO, or little, need for explicit DOM manipulation code)
- Event Handling
- Modules
- Dependency injection (automatic wiring of application)
- … and much more …

Directive, Scope and Control

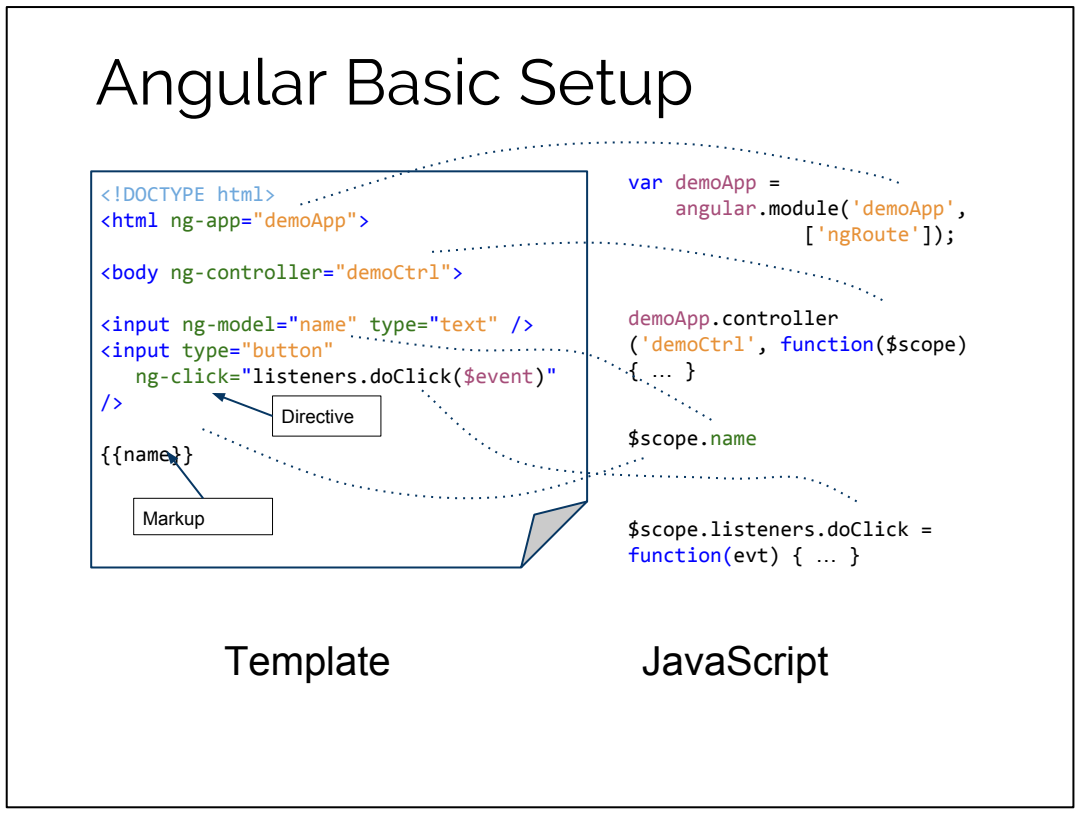


To use Angular we enhance HTML with Angular directives (often attributes)

- The enhanced HTML is called a template
- After DOM is constructed Angular will process the DOM (Compiling the DOM) thereby:
  - Building a hierarchy of $scopes "mirroring" the DOM structure (some directives create scopes some do not)
    - $scopes will hold data common to some DOM node and an Angular controller
      - $scope parameter automatically supplied by Angular
    - $scopes are distinct programming scopes
      - Possible to access parent scope (not used)
  - Setting up an event system between DOM-node and $scope
    - If data in $scope changed (in JS code), change propagated to DOM
    - If DOM changed propagated to $scope (accessible in some variable in scope)
    - This is called (two way) databinding

NOTE: Objects with leading $ in names are Angular native (don't use $ in your names)

- Confusing because jQuery uses $ …



Angular handles a lot in the background (problem is to find out what and how)

Dissection (simplified). See also code sample

- ng-app*) is a directive to auto-bootstrap an AngularJS application. The value for the directive is the module to load
- Our module is "demoApp". Our module has a dependency on Angular ngRoute -module (in code we have access to the angular-object)
  - An application consists of modules
- ng-controller attaches a controller object to the element/node (and subelements/nodes).
- We attach our "demoCtrl"-control to the <body>. The controller has a scope (to store data and functions)
- ng-model defines data in the scope. We define a variable "name". Name will be a n attribute of the scope.
- In the scope we also create a listener. We connect the button to the listener using ng-click directive
- Markup is a simple way to access the properties of the $scope (display value of $scope.name).

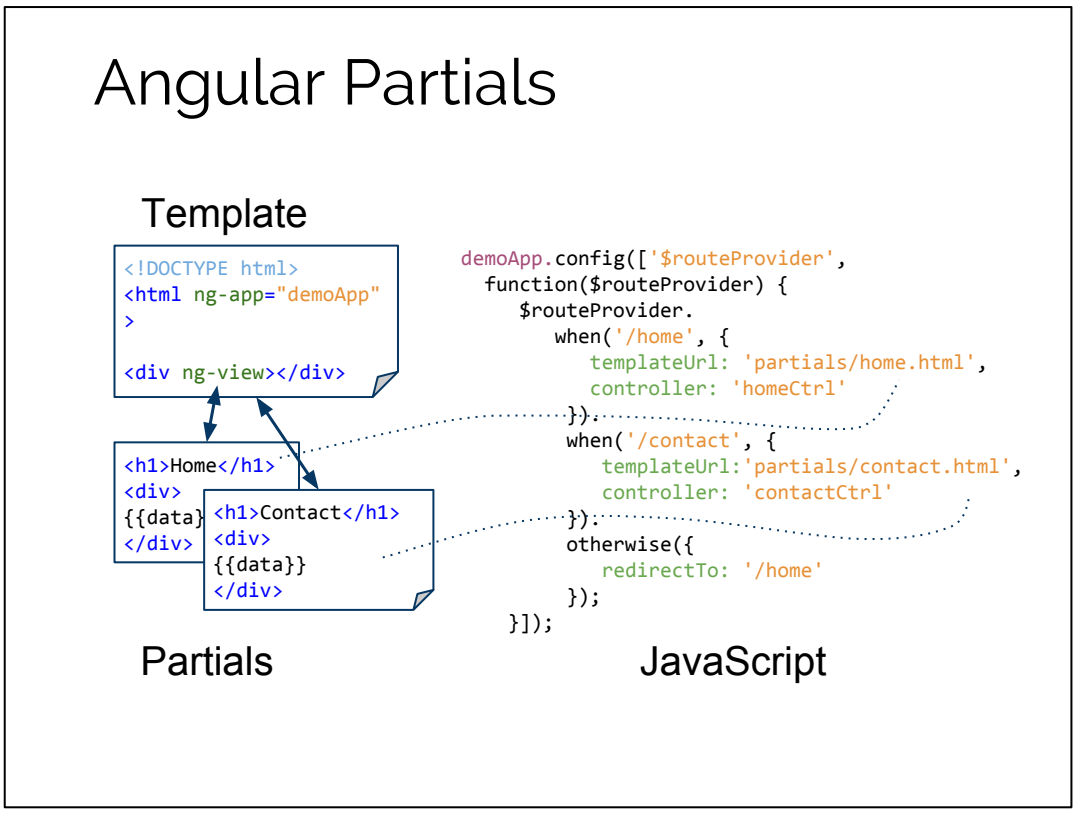
The synchronization between the $scope data and the displayed DOM is automatically handled by Angular (both ways!)

- Data binding works if data manipulated from inside Angular. If not have to force update using $apply(function(){…})

NOTE: No explicit DOM manipulation needed!

NOTE: Spelling!

*) ngApp and ng-app refers to the same directive, the former is the Angular API name the latter is how it's written in HTML



Partials are used for modular pages (similar to JSP). See code samples

Dissection (simplified) See also code samples

- ng-view, is a marker where to put the partial content.
- What will be inserted is configured using the config-method (from Angular). Config will connect URLs with partials and controllers (each partial will have it's own controller)
- Config uses dependency Injection to get a built in Angular object, $routeProvider used to specify the above.

NOTE: When running code samples use server else possibly cross origin problems

Angular Services

List as dependency in app.js

```
var personReg = angular.module('PersonReg', [
'PersonRegService'... ]);
```

The Services (services.js)

```
var personRegService = angular.module('PersonRegService', [] );
personRegService.factory('PersonRegProxy', ['$http',
     function($http) {
       var url = 'http://localhost:8080/rest_backend...';

     return {
       findAll: function() {
           return $http.get(url);
       },
       findRange: function(first, count) {
              return $http.get(url + "/range?fst=" + first +
           "&count=" + count);
       }
       ...
}]);
```

Angular services are substitutable objects that are wired together using dependency injection (DI). You can use services to organize and share code across your app.

- A service is a singleton
- Any user of it will get the same reference

Dissection of code

- We have named the module (for potentially many services) PersonRegService
  - Module has a service named PersonRegProxy, the name to be used to inject an object into a controller
- We register a dependency on the module when creating the application (the module), se slide
- Calling the factory-method of the module will return the anonymous object (this service also uses the $http object, injected by Angular). Factory is a provider

Location

Location

```
function($scope, $location) {
        $scope.navigate = function(url) {
            $location.path(url);
        };
    }
```

The $location service exposes the current URL in the browser address bar

- Used for navigation
- Watch and observe the URL
- Change the URL
- Injected

Angular Observer Pattern

Observer Pattern

```
personRegControllers.controller('PersonListCtrl', ['$scope', … ,
    function($scope, PersonsRegProxy) {
        ...
        $scope.currentPage = 0;
        $scope.someValue = 45;
        ...
        $scope.$watch('currentPage', function() {
                getRange();
        });
        function getRange() {
             $scope.someValue = 99;
        }
    }]);
```

In Page

```
<button class="btn btn-default" ng-disabled="currentPage === 0"
           ng-click="currentPage = currentPage - 1"> Previous
</button>
```

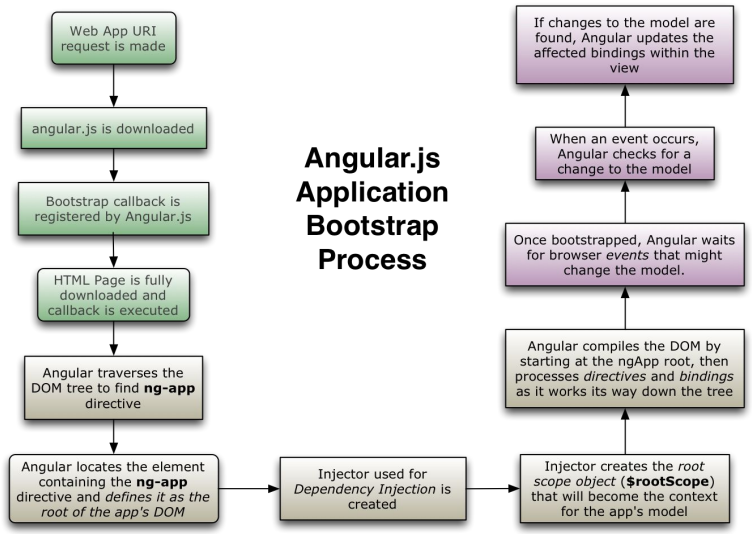
A lot can be handled by Angular but sometimes need explicit observer

- Add a $watch for some variable in $scope (using name of variable)

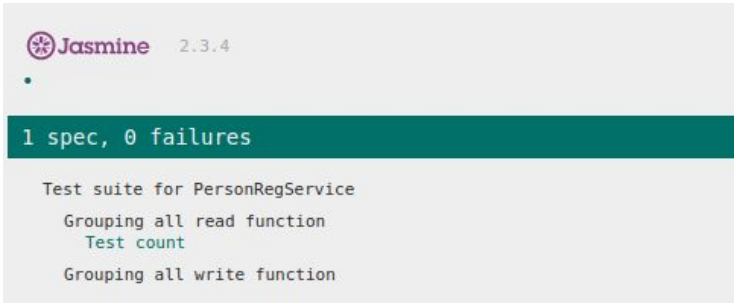
Assume some action (in actual scope) in page (ng-click)

- When $scope.currentPage's value changed in page getRange() called.

Angular Bootstrap



JavaScript Testing



JS development heavily dependent on testing, normally need large test suites

- Many possibilities we (AngularJS) use Jasmine, similar to Junit
- See workshop skeleton code for use