Angular JS services and path

Services

In AngularJS, a service is a function, or object, that is available for, and limited to, your AngularJS application.

AngularJS has about 30 built-in services.

$timeout

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope,
$timeout) {
    $scope.myHeader = "Hello World!";
    $timeout(function () {
        $scope.myHeader = "How are you today?";
    }, 2000);
});
```

$interval

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $interval) {
    $scope.theTime = new Date().toLocaleTimeString();
    $interval(function () {
        $scope.theTime = new Date().toLocaleTimeString();
    }, 1000);
});
```

$location

```
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $location) {
    $scope.myUrl = $location.absUrl();
});
```

$http

```
<div ng-app="myApp" ng-controller="myCtrl">

<p>Today's welcome message is:</p>
<h1>{{myWelcome}}</h1>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
    $http.get("welcome.html")
    .then(function(response) {
        $scope.myWelcome = response.data;
    });
});
</script>
```

welcome.html:

```
<h1> This is today’s greeting </h1>
```

Methods

The .get method is a shortcut method of the $http service. There are several shortcut methods:

.delete()
.get()
.head()
.jsonp() − deprecated
.patch()
.post()
.put()

RESTful API HTTP methods (wikipedia)

RESTful API HTTP methods

Resource	GET	PUT	POST	DELETE
Collection URI, such as `http://example.com/resources`	**List** the URIs and perhaps other details of the collection's members.	**Replace** the entire collection with another collection.	**Create** a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.[17]	**Delete** the entire collection.
Element URI, such as `http://example.com/resources/item17`	**Retrieve** a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	**Replace** the addressed member of the collection, or if it doesn't exist, **create** it.	Not generally used. Treat the addressed member as a collection in its own right and **create** a new entry in it.[17]	**Delete** the addressed member of the collection.

HTTP Patch

The HTTP PATCH request method applies partial modifications to a resource.

The HTTP PUT method only allows complete replacement of a document.

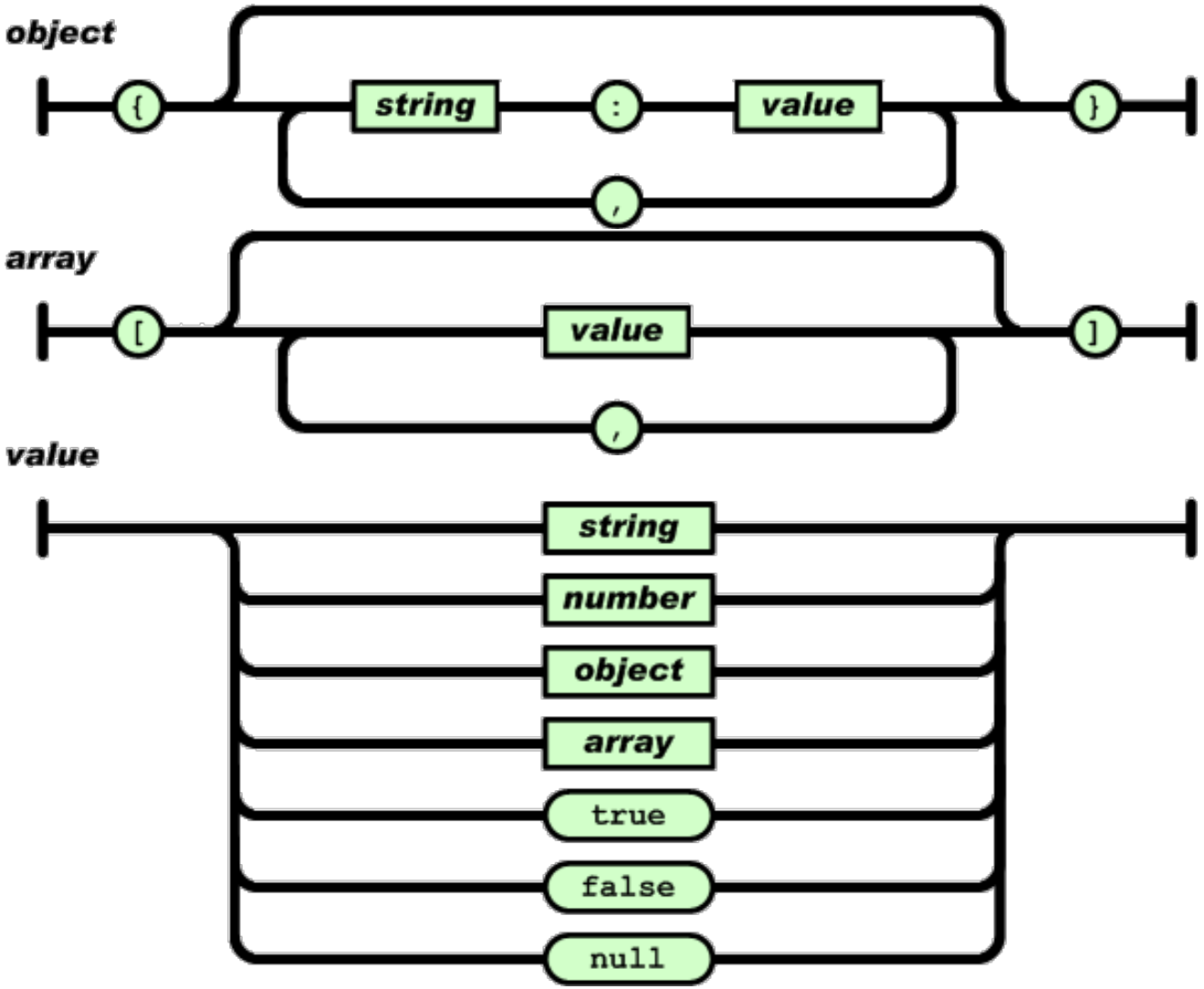
Unlike PUT, PATCH is not idempotent, meaning successive identical patch requests may have different effects. However, it is possible to issue PATCH requests in such a way as to be idempotent.

PATCH (like PUT) may have side-effects on other resources.

Full version

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
    $http({
        method : "GET",
        url : "welcome.htm"
    }).then(function mySuccess(response) {
        $scope.myWelcome = response.data;
    }, function myError(response) {
        $scope.myWelcome = response.statusText;
    });
});
```

JSON


object
{
string
:
value
}
,
array
[
value
]
,
value
string
number
object
array
true
false
null

Example


```
{
 "records":[
{"Name":"Alfreds Futterkiste","City":"Berlin","Country":"Germany"},
{"Name":"Ana Emparedados","City":"México D.F.","Country":"Mexico"},
{"Name":"Antonio Taquería","City":"México D.F.","Country":"Mexico"},
{"Name":"Around the Horn","City":"London","Country":"UK"},
{"Name":"Comércio Mineiro","City":"São Paulo","Country":"Brazil"}
 ]
 }
```

Jason and the Argonauts

Argo (Parsing JSON in Java)

```
{
  "name": "Black Lace",
  "sales": 110921,
  "totalRoyalties": 10223.82,
  "singles": [
    "Superman", "Agadoo"
  ]
}
```

```
String secondSingle = new JdomParser().parse(jsonText)
    .getStringValue("singles", 1);
```

http://argo.sourceforge.net/index.html

Parsing JSON in JavaScript

```
var text = '{ "name":"John", "birth":"1986-12-14", "city":"New York"}';

var obj = JSON.parse(text, function (key, value) {
  if (key == "birth") {
    return new Date(value);
  } else {
    return value;
  }
});

document.getElementById("demo").innerHTML = obj.name + ", " +
obj.birth;
```

Parsing JSON in JavaScript - simpler

```
var text = '{ "name":"John", "birth":"1986-12-14", "city":"New York"}';
var obj = JSON.parse(text);
obj.birth = new Date(obj.birth);

document.getElementById("demo").innerHTML = obj.name + ", " +
obj.birth;
```

Creating your own services

AngularJS services are singletons

A typical use of service is sharing data or functionalities among different controllers.

There are two ays to access them:

service() and factory()

Two ways

```
angular.module("myApp")
    .service("somma1", function() {
        this.somma = function(a,b) { return a + b};
    });

angular.module("myApp")
    .factory("somma2", function() {
        return function(a, b) { return a + b;}
    });
```

```
angular.module("myApp")
    .controller("myController",
    function($scope, somma1, somma2) {
        $scope.x = somma1.somma(1,2);
        $scope.y = somma2(1,2)
    });
```

An example

```
app.service('hexafy', function() {
    this.myFunc = function (x) {
        return x.toString(16);
    }
});

app.controller('myCtrl', function($scope, hexafy) {
    $scope.hex = hexafy.myFunc(255);
});
```

ng-path

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/
angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular-
route.js"></script>

<body ng-app="myApp”>
<p><a href="#/!">Main</a></p>
<a href="#!red">Red</a>
<a href="#!green">Green</a>
<a href="#!blue">Blue</a>
<div ng-view></div>

<p>Click on the links to navigate to
"red.html", "green.html", "blue.html",
 or back to "main.html"</p>
</body>
</html>
```

```
<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
        $routeProvider
        .when("/", {
            template : ”h1>main with local template</h1>"
            })
        .when("/red", {
            templateUrl : "red.html"
            })
        .when("/green", {
            templateUrl : "green.html"
            })
        .when("/blue", {
            templateUrl : "blue.html"
            });
        });
</script>
```

red.html:

```
<h1>this is the "red" page"</h1>
```

green.html:

```
<h1>this is the ”green" page"</h1>
```

blue.html:

```
<h1>this is the ”blue" page"</h1>
```

Output


localhost/public/ajstests/try_ng_routing.html#!/
Main
Red Green Blue
main with local template
Click on the links to navigate to "red.html", "green.html", "blue.html", or back to "main.html"
localhost/public/ajstests/try_ng_routing.html#!/green
Main
Red Green Blue
this is the "green" page"
Click on the links to navigate to "red.html", "green.html", "blue.html", or back to "main.html"

You can add an “.otherwise” clause

You can add behaviours, by adding controllers to the “.when”