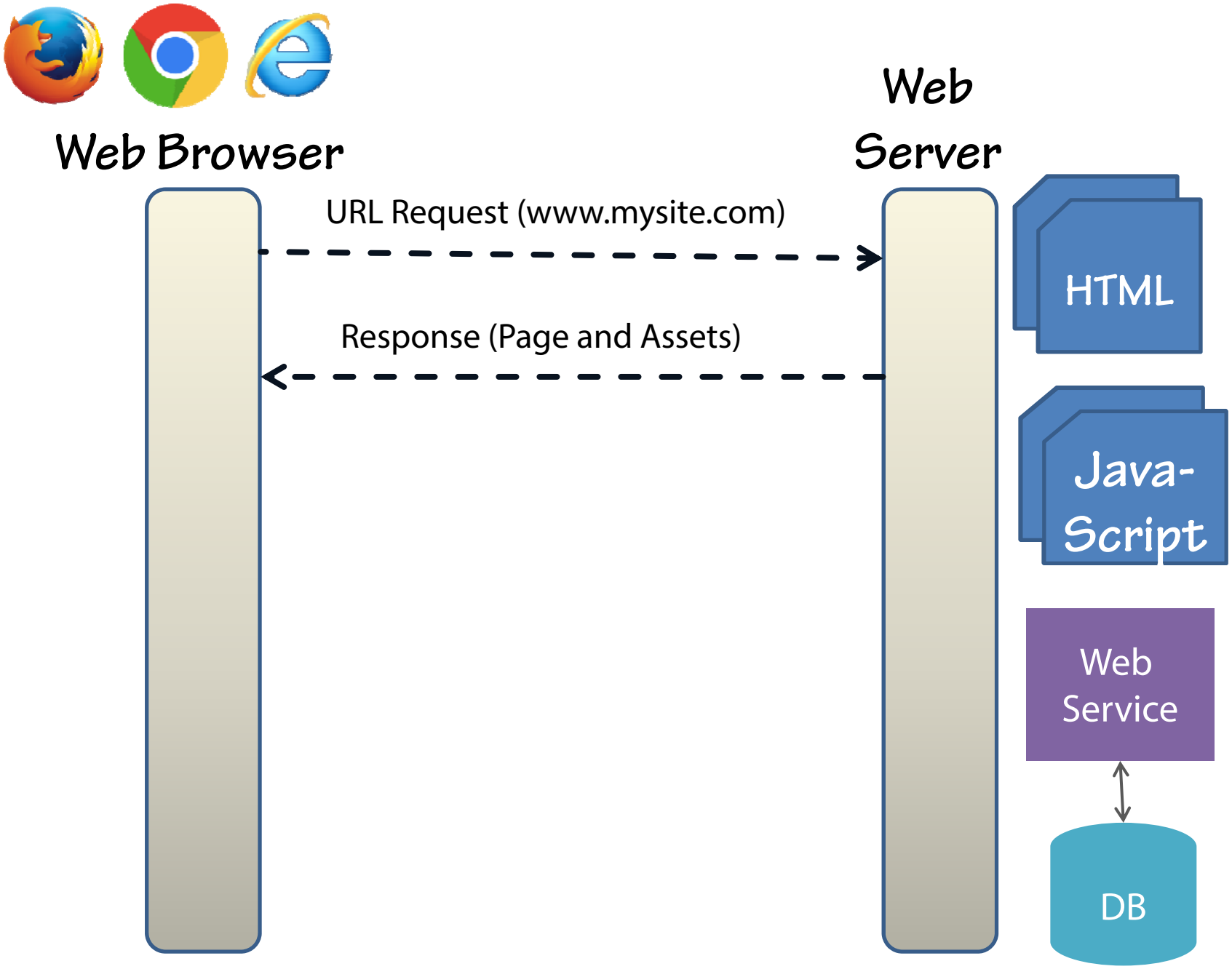
Accessing Data

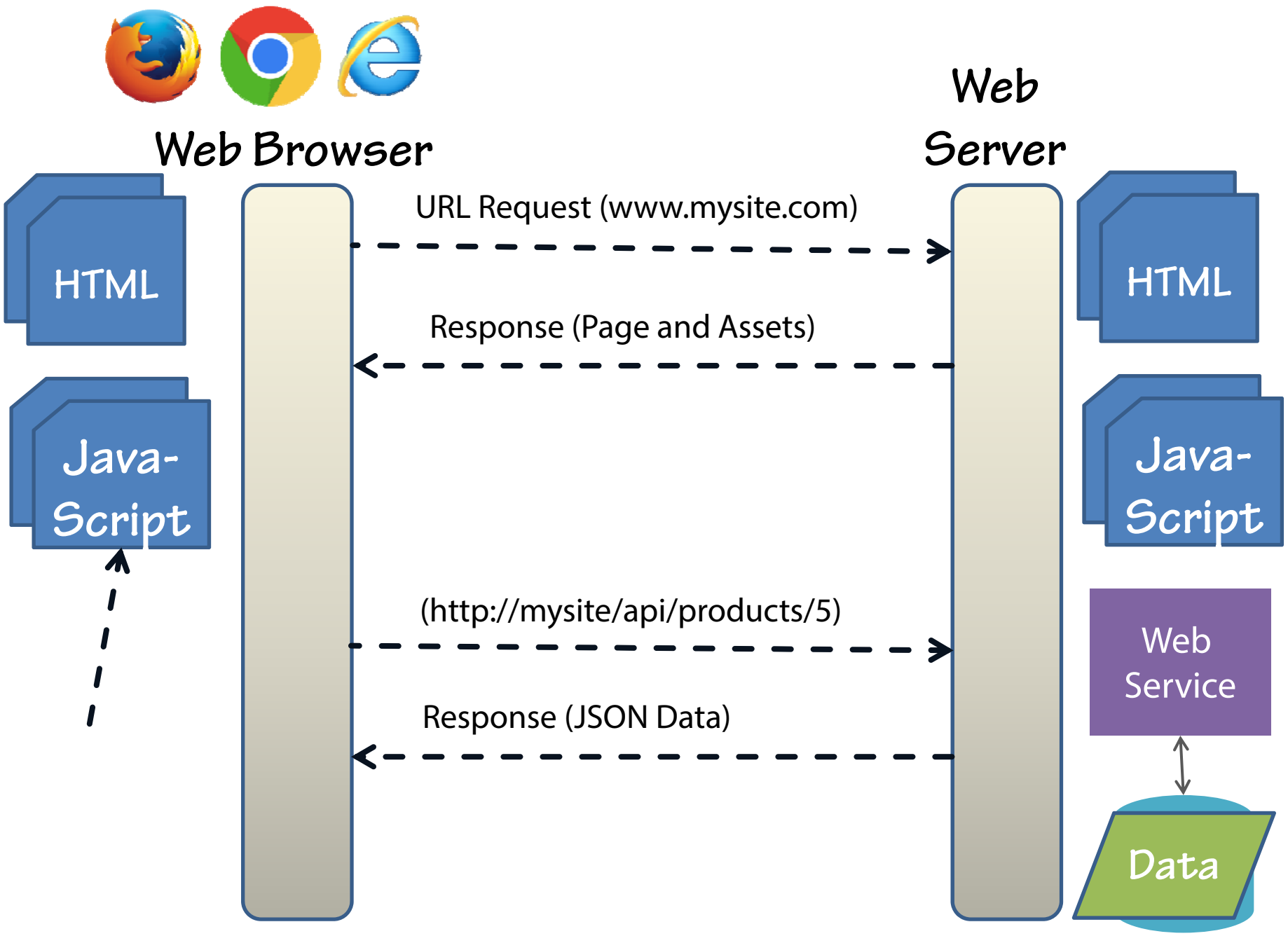
Deborah Kurata
http://blogs.msmvps.com/deborahk/
@DeborahKurata
deborahk@insteptech.com

pluralsight
hardcore dev and IT training

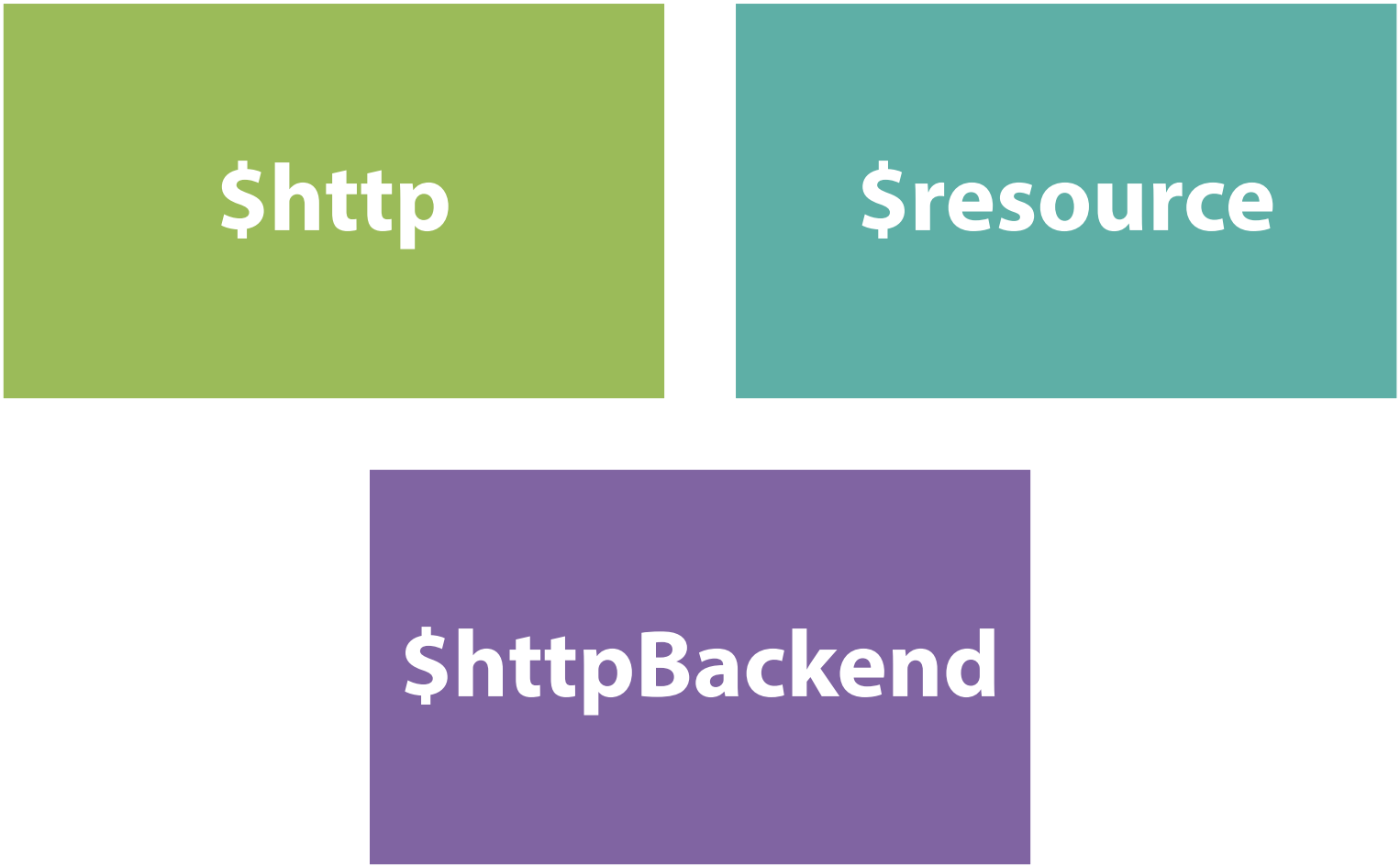
Module Objectives

- Understand how Data is Retrieved
- Build a Reusable Data Access Component
- Evaluate Existing Angular Services for Calling a Web Service
- Modify the Controller to Retrieve Data
- Fake it without the Web Service


Web Browser
Web Server
URL Request (www.mysite.com)
Response (Page and Assets)
HTML
Java-Script
Web Service
DB


Web Browser
Web Server
HTML
Java-Script
URL Request (www.mysite.com)
Response (Page and Assets)
(http://mysite/api/products/5)
Response (JSON Data)
HTML
Java-Script
Web Service
Data

Built-In Angular Services for Calling a Web Service



$http

Facilitates communication with remote Web servers

Call is asynchronous

When the request is complete:

- The callback function is executed
- The response object is passed in

```
$http.get("/api/products/")
     .then(function(response) {
          vm.products = response.data;
        });
```

$resource

Angular factory which creates a resource object

REST (Representational State Transfer)

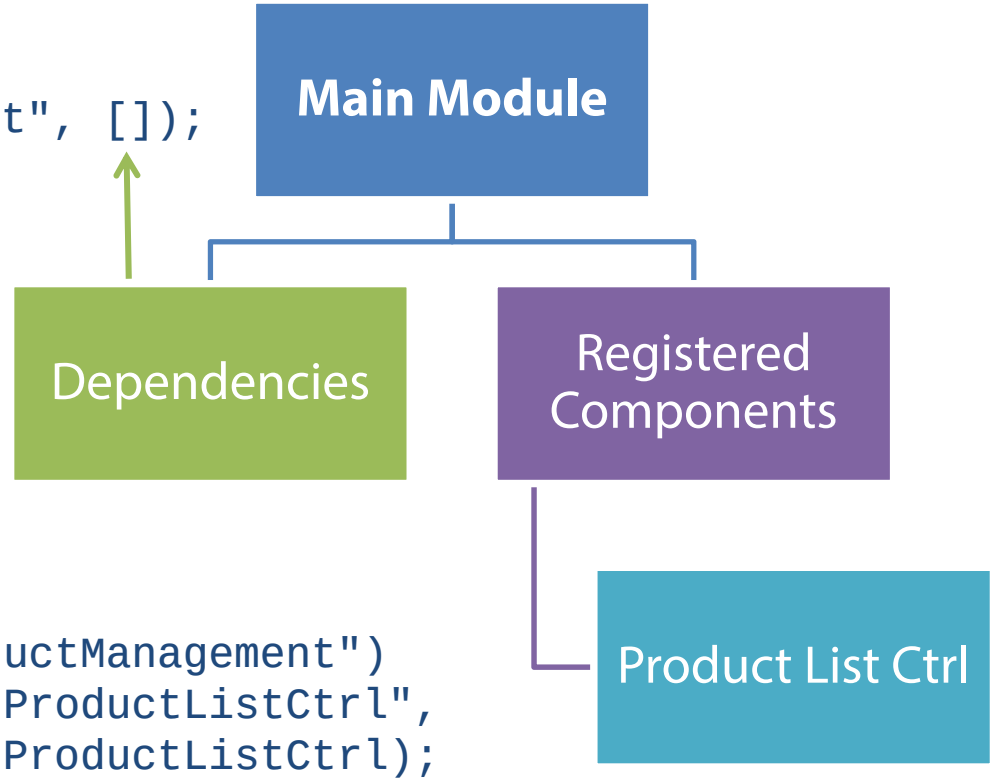
Abstracts away complexity

```
function productResource($resource) {
  return $resource("/api/products/:productId")
        }
```

Method Name	URL	Description
get	/api/products/5	Retrieves the specified product
query	/api/products	Retrieves all of the products
save	/api/products	Saves modifications to the product

Current Module

```
angular
 .module("productManagement", []);
```




```
angular
 .module("productManagement")
 .controller("ProductListCtrl",
             ProductListCtrl);
```

Hard-Coded Data

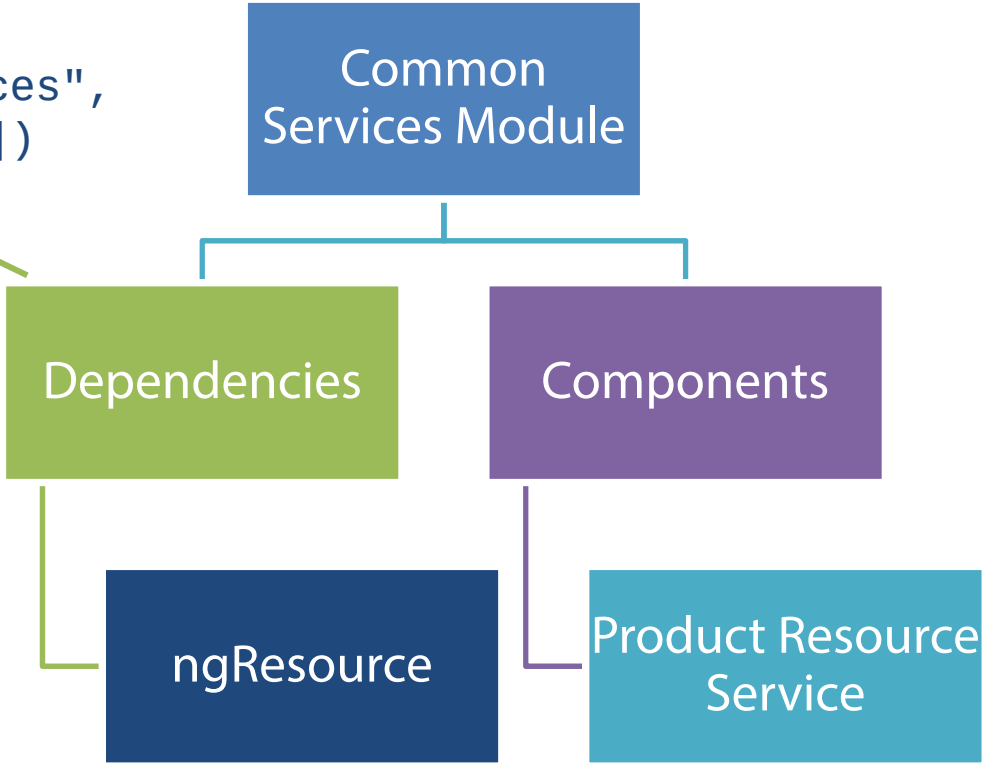
```
function ProductListCtrl() {
    var vm = this;

    vm.products = [
        {" productId": 1,
            "productName": "Leaf Rake",
            "productCode": "GDN-0011",
            "releaseDate": "March 19, 2009",
            "description": "Leaf rake with 48-inch handle.",
            "price": "19.95",
            "category": "garden",
            "tags": [ "leaf", "tool" ],
            "imageUrl": "http://openclipart.org/image/.../Leaf_Rake.png"
        },
        {"productId": 5...}];

    vm.showImage = false;
    vm.toggleImage = function () {
        vm.showImage = !vm.showImage;
    }
}
```

New Module

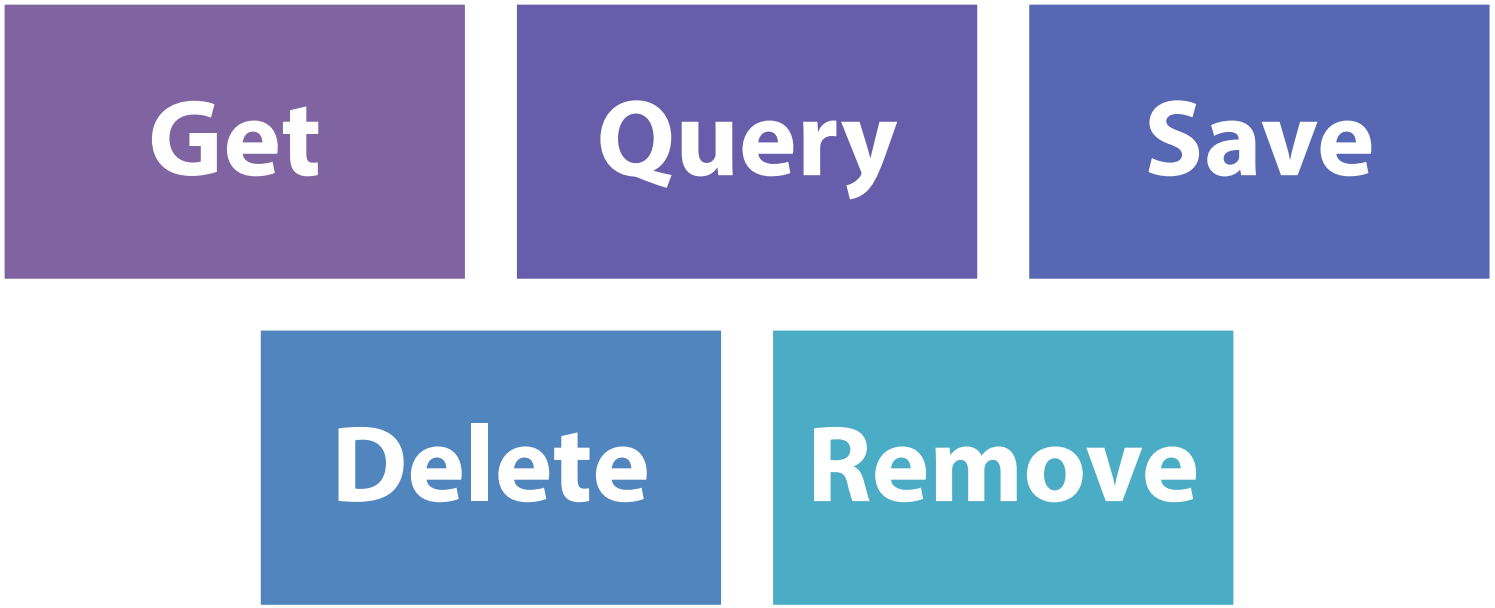
```
angular
 .module("common.services",
         ["ngResource"])
```




```
angular
 .module("common.services")
 .factory("productResource",
          ["$resource",
          productResource])
```

Using the Resource Object

```
function productResource($resource) {
    return $resource("/api/products/:productId")
}
```




```
productResource.query(function (data) {
   vm.products = data;
});
```

Faking the Web Service: $httpBackend

Angular's fake HTTP backend implementation

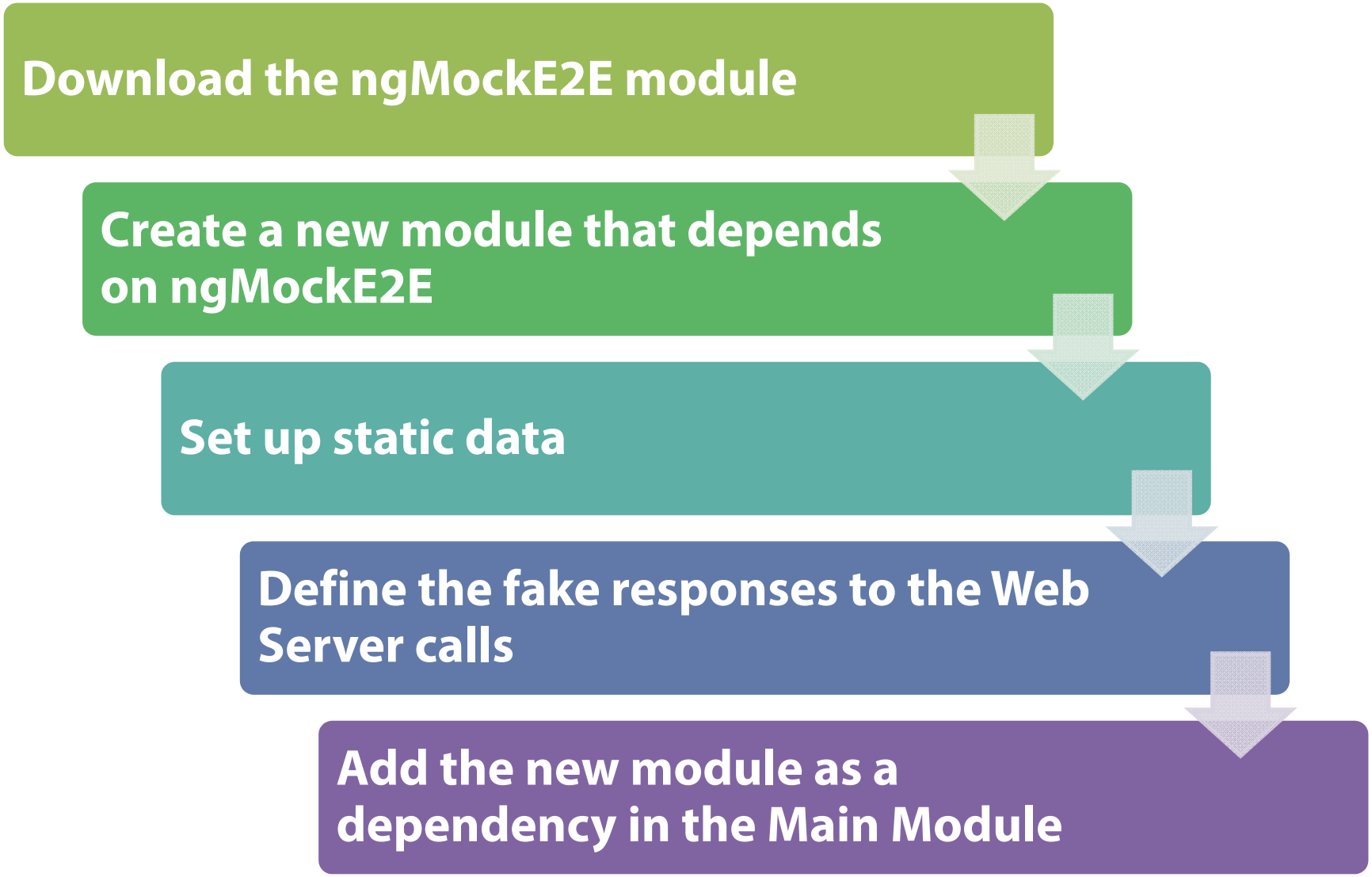
Mocks the calls to the Web Service

Returns static data to the application

Two implementations:

- **ngMock**: for unit testing applications
- **ngMockE2E**: for end-to-end testing or backend-less development

Steps to Mocking the Web Server



Module Objectives

Understand how Data is Retrieved

Build a Reusable Data Access Component

Evaluate Existing Angular Services for Calling a Web Service

Modify the Controller to Retrieve Data

Fake it without the Web Service

Angular
Line of Business Applications