JavaScript Frameworks

Outline

- Node.js
- Angular.js

Node.js

1. Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine.

2. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.

3. Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionality.

4. Modules handle file system I/O, networking (DNS, HTTP, TCP, TLS/SSL, or UDP), binary data (buffers), cryptography functions, data streams and other core functions.

Basic functionality

HTTP

- To use the HTTP server and client one must require('http').
- The HTTP interfaces in Node.js are designed to support many features of the protocol which have been traditionally difficult to use. In particular, large, possibly chunk-encoded, messages. The interface is careful to never buffer entire requests or responses--the user is able to stream data.

http.createServer([requestListener])

- Returns a new instance of http.Server.
- The requestListener is a function which is automatically added to the 'request' event.

http.request(options[, callback])

- Node.js maintains several connections per server to make HTTP requests. This function allows one to transparently issue requests.
- options can be an object or a string. If options is a string, it is automatically parsed with url.parse().

http.get(options[, callback])

- Since most requests are GET requests without bodies, Node.js provides this convenience method. The only difference between this method and http.request() is that it sets the method to GET and calls req.end() automatically.

Basic functionality

File System

- File I/O is provided by simple wrappers around standard POSIX functions. To use this module do require('fs'). All the methods have asynchronous and synchronous forms..

fs.readFile(file[, options], callback)

- Asynchronously reads the entire contents of a file.
- The callback is passed two arguments (err, data), where data is the contents of the file.
- If no encoding is specified, then the raw buffer is returned.

fs.readFileSync(file[, options])

- Synchronous version of fs.readFile. Returns the contents of the file.
- If the encoding option is specified then this function returns a string. Otherwise it returns a buffer.
- .

Basic functionality

Buffer

- Prior to the introduction of TypedArray in ECMAScript 2015 (ES6), the JavaScript language had no mechanism for reading or manipulating streams of binary data. The Buffer class was introduced as part of the Node.js API to make it possible to interact with octet streams in the context of things like TCP streams and file system operations.

- The Buffer class is a global within Node.js, making it unlikely that one would need to ever use require('buffer').

Buffers and Character Encodings

- Buffers are commonly used to represent sequences of encoded characters such as UTF8, UCS2, Base64 or even Hex-encoded data. It is possible to convert back and forth between Buffers and ordinary JavaScript string objects by using an explicit encoding method.

Example usage – Buffer Class

buffer.js:

```
const buf = new Buffer('hello world', 'ascii');

console.log(buf.toString('hex'));
  // prints: 68656c6c6f20776f726c64
console.log(buf.toString('base64'));
  // prints: aGVsbG8gd29ybGQ=
```

Example usage – Read File

fs-readFile.js:

```
var fs = require('fs');
fs.readFile('./intro.txt', function (err, data) {
  if (err) throw err;
  console.log(data);
});
```

Intro.txt:

```
JsApp.US is a hosting platform for node.js  applications.
It is setup to be a platform to coddle to quick, weekend hack like projects.
```

Run command:

$ nodejs fs-readFile.js

```
ubuntu@ip-172-31-12-48:~/nodejs$ nodejs fs-readFile.js
CSCI571 focuses on the phenomenon known as the World Wide Web (WWW or Web).
It's focus is to present many of the core technologies that the Web is based upon.
```
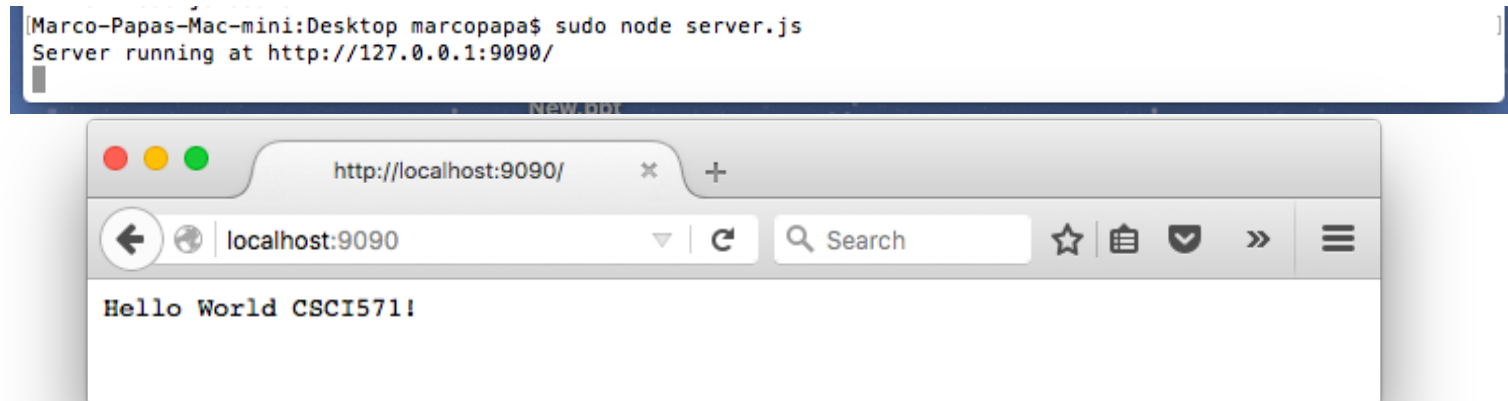
Example usage – Run Web Server

server.js:

```
const http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello World CSCI571!\n');
}).listen(9090, '127.0.0.1', function () {
  console.log(`Server running at http://localhost:9090/`);});
```

Run command:

$ sudo node server.js

```
[Marco-Papas-Mac-mini:Desktop marcopapa$ sudo node server.js
Server running at http://127.0.0.1:9090/
```

http://localhost:9090/

localhost:9090   Search

Hello World CSCI571!

Node.js on AWS

- Create Ubuntu Micro EC32 instance
- ssh to the AWS ubuntu server
- Download node.js using 'wget'
- Execute the binaries, make, and install via the command:
  ./configure && make && sudo make install

Node.js on AWS

- AWS SDK for JavaScript in Node.js

- Provides JavaScript objects for AWS services including Amazon S3, Amazon EC2, DynamoDB, Amazon Elastic Beanstalk and many more.

- Single, downloadable package includes the AWS JavaScript Library and documentation

- See: https://aws.amazon.com/sdk-for-node-js/
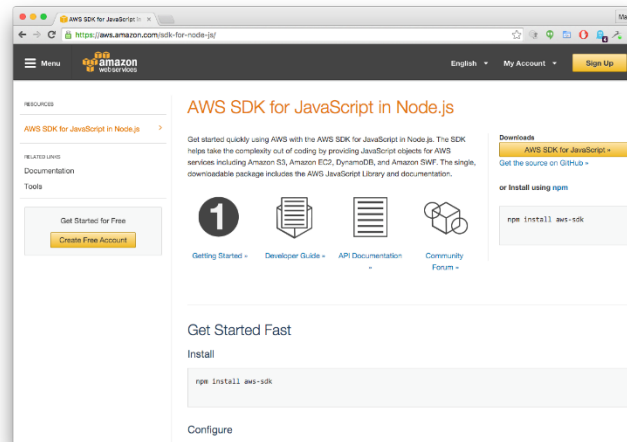
Node.js on Google Cloud Platform
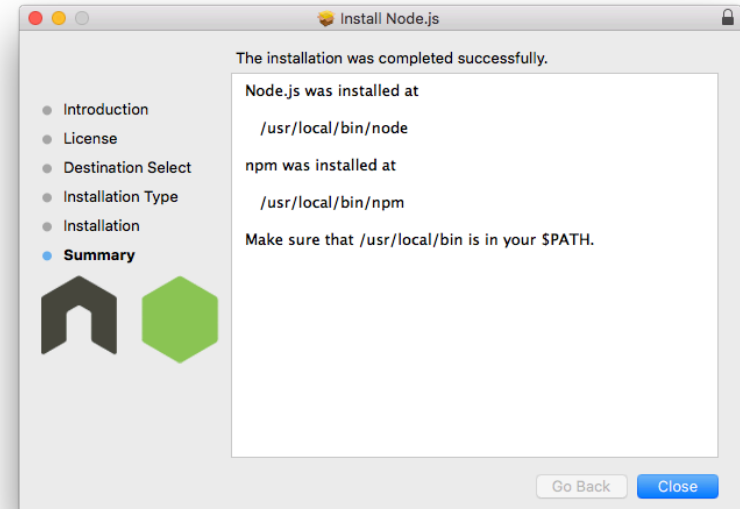
- Node.js app deployed using Google App Engine Managed VMs
- Scales to serve millions of requests
- Supports structures and binary data, authentication, logging, events
- See: https://cloud.google.com/nodejs

Copyright © Marco Papa 2017
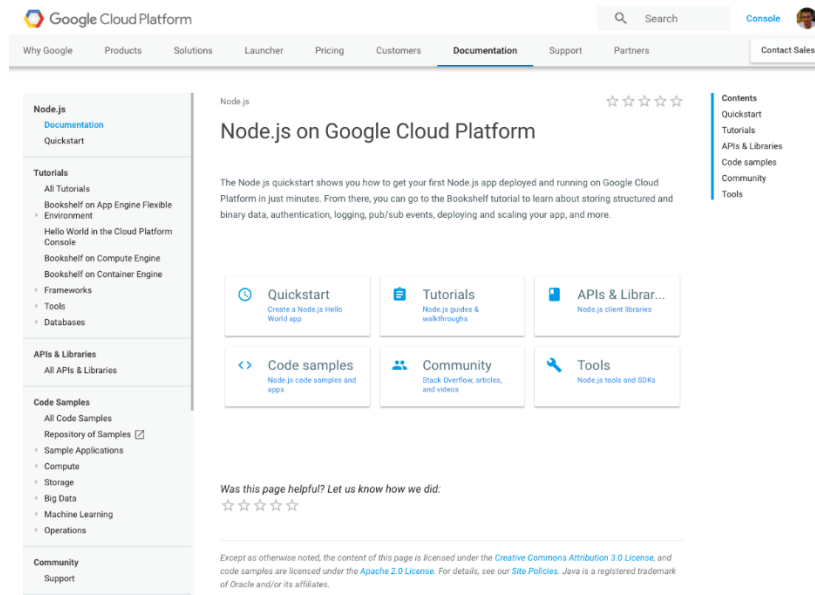
Node.js on OS X and Windows

- Download OS X package at https://nodejs.org/en/
- Learn Node.js on you local MacBook or Windows PC
- Latest version is 7.0
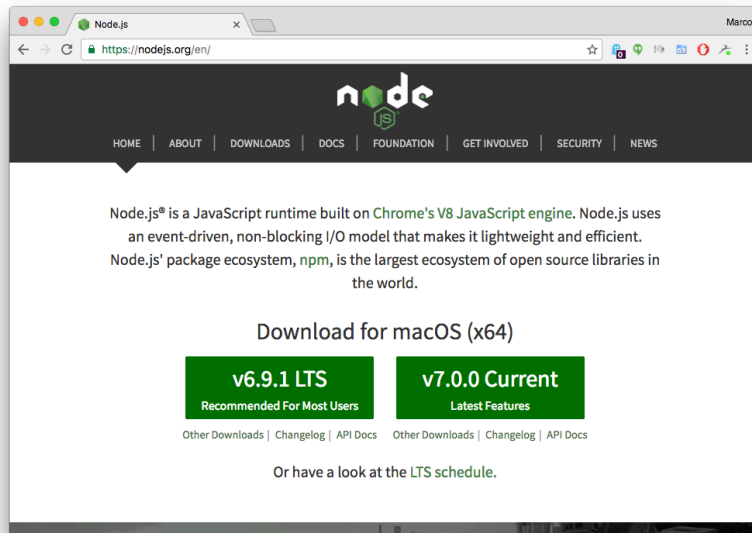
Related URLs

- **Node.js website:** https://nodejs.org/

- **Node.js on Github:** https://github.com/nodejs/node

- **NPM:** https://www.npmjs.com/

- **Learn Node.js in terminal:** https://github.com/workshopper/learnyounode

- **Tools:**

    http://expressjs.com/

    http://gruntjs.com/

Angular.js

Why AngularJS?

- HTML is great for declaring static documents, but it falters when we try to use it for declaring **dynamic views** in web-applications. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop.

Alternatives

- Other frameworks deal with HTML's shortcomings by either abstracting away HTML, CSS, and/or JavaScript or by providing an imperative way for manipulating the DOM. Neither of these address the root problem that HTML was not designed for **dynamic views**.

Extensibility

- AngularJS is a toolset for building the framework most suited to your application development. It is fully extensible and works well with other libraries. Every feature can be modified or replaced to suit your unique development workflow and feature needs. Read on to find out how.

Basic functionality

Control of the app

Data Binding

- Data-binding is an automatic way of updating the view whenever the model changes, as well as updating the model whenever the view changes. This is awesome because it eliminates DOM manipulation from the list of things you have to worry about.

Controller

- Controllers are the behavior behind the DOM elements. AngularJS lets you express the behavior in a clean readable form without the usual boilerplate of updating the DOM, registering callbacks or watching model changes.

Plain JavaScript

- Unlike other frameworks, there is no need to inherit from proprietary types in order to wrap the model in accessors methods. Angular models are plain old JavaScript objects. This makes your code easy to test, maintain, reuse, and again free from boilerplate.

Basic functionality

Wire up a Backend

Deep Linking

- A deep link reflects where the user is in the app, this is useful so users can bookmark and email links to locations within apps. Round trip apps get this automatically, but AJAX apps by their nature do not. AngularJS combines the benefits of deep link with desktop app-like behavior.

Form Validation

- Client-side form validation is an important part of great user experience. AngularJS lets you declare the validation rules of the form without having to write JavaScript code. Write less code, go have beer sooner.

Server Communication

- AngularJS provides built-in services on top of XHR as well as various other backends using third party libraries. Promises further simplify your code by handling asynchronous return of data.

Basic functionality

Create Components

Directives
- Directives is a unique and powerful feature available only in Angular. Directives let you invent new HTML syntax, specific to your application.

Reusable Components
- We use directives to create reusable components. A component allows you to hide complex DOM structure, CSS, and behavior. This lets you focus either on what the application does or how the application looks separately.

Localization
- An important part of serious apps is localization. Angular's locale aware filters and stemming directives give you building blocks to make your application available in all locales.

Introduction

- AngularJS is a complete JavaScript-based open-source front-end web application framework.
- Its mainly maintained by Google and some community of individuals.
- It provides a framework for client-side model–view–controller (MVC) and model–view–viewmodel (MVVM) architectures.
- AngularJS is the frontend part of the **MEAN stack**, consisting of **M**ongoDB database, **E**xpress.js web application server framework, **A**ngular.js itself, and **N**ode.js runtime environment.

Companies that Use Angular JS

There are approximately 12,000 other sites out of 1 million tested in October 2016 that use Angular JS
Companies that use Angular JS [1] - http://libscore.com/?#angular

Goals

AngularJS de-emphasizes explicit DOM manipulation with the goal of improving testability and performance.

- Design goals are
    - It decouples DOM manipulation from application logic
    - It decouple the client side of an application from the server side.
    - It provides structure for building an application
        - Designing the UI
        - Writing the business logic
        - Testing

Goals (cont'd)

- Angular JS framework adapts and extends traditional HTML.
- It supports dynamic content through two-way data-binding
- Two-way data-binding allows for the automatic synchronization of models and views.
- The tasks in angular bootstrapper occur in 3 phases
    - Creation of a new Injector
    - Compilation of the directives that decorate the DOM
    - Linking of all directives to scope

Data Binding

Classical

Angular

One-Way Data Binding

Two-Way Data Binding
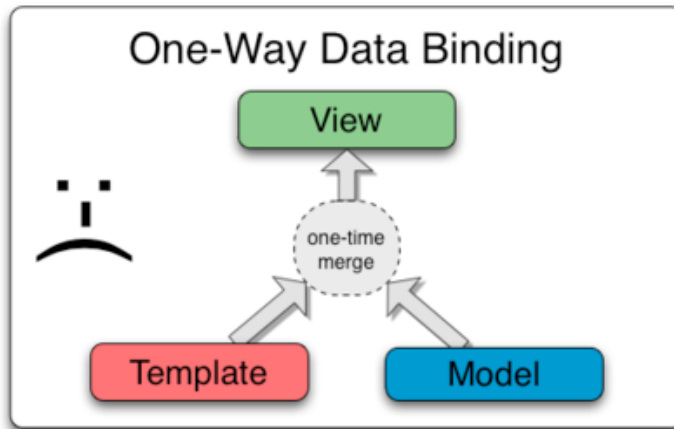
Any changes that the user makes to the view are not reflected in the model
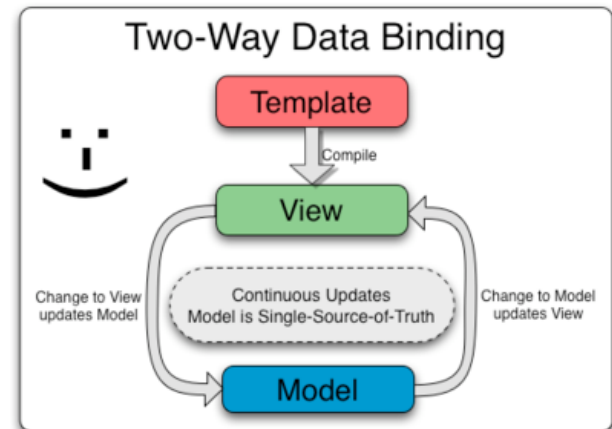
The view is just a projection of the model. So there is automatic refresh of the data between view and model

Angular Directives

- AngularJS directives allows to specify custom reusable HTML-like elements and attributes.
- Some of the Angular Directives are
    - ng-app
    - ng-controller
    - ng-bind
    - ng-model
    - ng-class
    - ng-repeat

Angular Directives (Cont'd)

- **ng-app**
  - Declares the root element of an AngularJS application.
- **ng-controller**
  - Specifies a JavaScript controller class that evaluates HTML expressions.
- **ng-bind**
  - Sets the text of a DOM element to the value of an expression.
- **ng-model**
  - Similar to ng-bind, but establishes a two-way data binding between the view and the scope.
- **ng-repeat**
  - Instantiate an element once per item from a collection.

Code Snippet – Angular Instantiation

```
<script>
var app = angular.module("myApp", []);
app.controller("myController", function($scope,$http) {
   //  $scope holds your model (metadata) for your application
   $scope.topic = "CSCI 571";
});
</script>

<body ng-app="myApp" ng-controller="myController">
</body>
```

- An AngularJS module defines an application.
- The module is a container for controllers
- Controllers always belong to a module.

Angular Module - Example

```html
<div ng-app="formExample" ng-controller="ExampleController">
  <form class="form container simple-form">
    <div class="form-group">
    <label>Name :</label> <input class="form-control" type="text" ng-model="user.name" />
    </div>
    <div class="form-group">
    <label>E-mail:</label> <input class="form-control" type="email" ng-model="user.email" /><br />
    </div>
    <div class="form-group">
    <label>School:</label> <input class="form-control" type="text" ng-model="user.school" /><br />
    </div>
    <div class="form-group">
    <label>Level:</label>
      <select class="form-control" ng-model="user.level">
        <option value=""></option>
        <option value="G">Graduate</option>
        <option value="UG">Under Graduate</option>
        <option value="PhD">Doctoral</option>
      </select><br />
    </div>
    <input class="btn btn-default" type="button" ng-click="reset()" value="Reset" />
    <input class="btn btn-primary" type="submit" ng-click="update(user)" value="Save" />
  </form>
  <br/>
  <div class="container">
    <pre>user = {{user | json}}</pre>
    <pre>master = {{master | json}}</pre>
  </div>
</div>
```

```javascript
angular.module('formExample', [])
  .controller('ExampleController', ['$scope', function($scope) {
    $scope.master = {};

    $scope.update = function(user) {
      $scope.master = angular.copy(user);
    };

    $scope.reset = function() {
      $scope.user = angular.copy($scope.master);
    };

    $scope.reset();
}])
```

A simple example which shows how angular JS process model.

Angular Repeat with data from static array

```html
<div class="row">
    <div class="col-md-6 col-sm-12">
        <h3>Loading Data from Array</h3>
        <h4>Web Tech Producer</h4>
        <table class="table table-striped">
            <tr>
                <th>#</th>
                <th>Name</th>
                <th>Office Hours</th>
                <th>Location</th>
            </tr>
            <tr ng-repeat="x in producers track by $index">
                <td>{{$index + 1}}</td>
                <td>{{x.Name}}</td>
                <td>{{x.Office}}</td>
                <td>{{x.Location}}</td>
            </tr>
        </table>
    </div>
</div>
```

```javascript
var app = angular.module("data", []);
app.controller("data", function ($scope, $http) {
    $scope.producers = [
        {
            Name: "Producer 1",
            Office: "10-11 AM",
            Location: "Leavey Library (LVL) 201"
        },
        {
            Name: "Producer 2",
            Office: "9-10 AM",
            Location: "Leavey Library (LVL) 202"
        },
        {
            Name: "Producer 3",
            Office: "4-5 PM",
            Location: "Leavey Library (LVL) 203"
        },
        {
            Name: "Producer 4",
            Office: "2-3 PM",
            Location: "Leavey Library (LVL) 204"
        },
        {
            Name: "Producer 5",
            Office: "5-6 PM",
            Location: "Leavey Library (LVL) 201"
        },
        {
            Name: "Producer 6",
            Office: "10-11 AM",
            Location: "Leavey Library (LVL) 209"
        },
        {
            Name: "Producer 7",
            Office: "10-11 AM",
            Location: "Leavey Library (LVL) 202"
        }
    ];
});
```

ng-repeat works like a for loop and replicates
the template to the number of rows in the model

Angular Repeat from external source

```html
<table class="table table-responsive table-striped" ng-app="myapp" ng-controller="myapp">
  <tr>
    <th>Name</th>
    <th>City</th>
    <th>Country</th>
  </tr>
  <tr ng-repeat="x in rows">
    <td>{{x.Name}}</td>
    <td>{{x.City}}</td>
    <td>{{x.Country}}</td>
  </tr>
</table>
```

```javascript
angular.module('myapp', [])
 .controller('myapp', function($scope, $http) {
   $scope.rows = {};

   $http.get("http://www.w3schools.com/angular/customers.php").then(function(response){
     $scope.rows = response.data.records;
   });
});
```

$http holds the xml http request handler in Angular.

Name	City	Country
Alfreds Futterkiste	Berlin	Germany
Ana Trujillo Emparedados y helados	México D.F.	Mexico
Antonio Moreno Taquería	México D.F.	Mexico
Around the Horn	London	UK
B's Beverages	London	UK
Berglunds snabbköp	Luleå	Sweden
Blauer See Delikatessen	Mannheim	Germany

Angular Sort and Search

```html
<table class="table table-responsive table-striped" ng-app="myapp" ng-controller="myapp">
  <tr>
    <th>Name</th>
    <th>City</th>
    <th>Country
      <input type="text" class="form-control pull-right" style="width:40%;" placeholder="Search" ng-model="search" />
    </th>
  </tr>
  <tr ng-repeat="x in rows| orderBy:Name | filter: search">
    <td>{{x.Name}}</td>
    <td>{{x.City}}</td>
    <td>{{x.Country}}</td>
  </tr>
</table>
```

```javascript
angular.module('myapp', [])
  .controller('myapp', function($scope, $http) {
    $scope.rows = {};

    $http.get("http://www.w3schools.com/angular/customers.php").then(function(response){
      $scope.rows = response.data.records;
    });
});
```

orderBy: sort the Column ascending
orderBy:<column>:<reverse>
<column> - the column you want to sort
<reverse> - true-descending, false-ascending

Filter: search the rows in the model
Filter:<searchstring> e.x. filter: search
Or
Filter:<column_based_search> e.x.
filter:{<column>:search}

Name	City	Country
Alfreds Futterkiste	Berlin	Germany
Ana Trujillo Emparedados y helados	México D.F.	Mexico
Antonio Moreno Taquería	México D.F.	Mexico
Around the Horn	London	UK
B's Beverages	London	UK
Berglunds snabbköp	Luleå	Sweden
Blauer See Delikatessen	Mannheim	Germany

Angular External UI Components

```
<table class="table table-responsive table-striped" ng-app="myapp" ng-controller="myapp">
 <tr>
   <th>Name</th>
   <th>City</th>
   <th>Country
     <input type="text" class="form-control pull-right" style="width:40%;" placeholder="Search" ng-
model="search" />
   </th>
 </tr>
 <tr dir-paginate="x in rows| orderBy:Name:false | filter: search| itemsperpage: 10" pagination-
id="example">
   <td>{{x.Name}}</td>
   <td>{{x.City}}</td>
   <td>{{x.Country}}</td>
 </tr>
</table>
<dir-pagination-controls max-size="10" boundary-links="true" direction-links="true" max-size="10"
pagination-id="example"></dir-pagination-controls>
```

External components need to be added to the angular application.
angular.module('myapp',
[<external_components>])

```
angular.module('myapp', ['angularUtils.directives.dirPagination'])
 .controller('myapp', function($scope, $http) {
   $scope.rows = {};

   $http.get("http://www.w3schools.com/angular/customers.php").then(function(response){
     $scope.rows = response.data.records;
   });
});
```

Copyright © Marco Papa 2017

Example usage & Related URLs

Examples
- **Hello World:** https://angularjs.org/#the-basics
- **Todo List:** https://angularjs.org/#add-some-control
- **Advanced Single Page App:** https://angularjs.org/#wire-up-a-backend

Related URLs
- **Angular.js website:** https://angularjs.org
- **Angular.js on Github:** https://github.com/angular/angular.js
- **Tutorial:** https://docs.angularjs.org/tutorial
- **Angular.js Course**
  http://campus.codeschool.com/courses/shaping-up-with-angular-js/level/1/section/1/creating-a-store-module
- **Angular 2 (In beta):** https://angular.io/