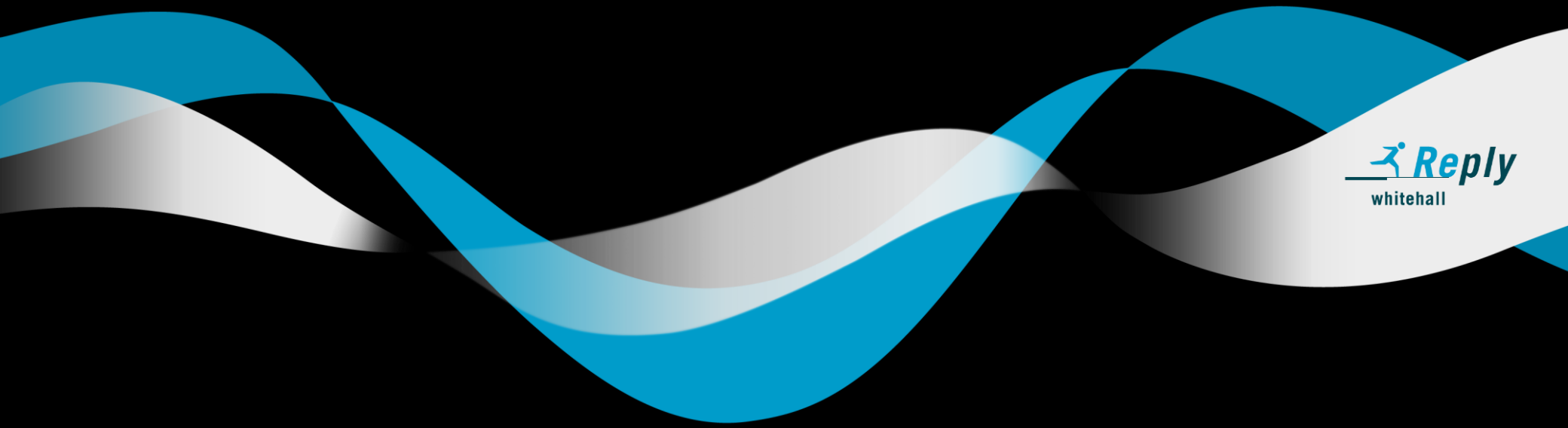
Introduzione a AngularJS

Simone Corrieri

29/03/2017



Obiettivo

- *Fornire una conoscenza di base della tecnologia AngularJS*
- *I componenti principali*
- *Taglio pratico verso il codice*

Agenda

- *Un po’ di storia..*
- *AngularJS: introduzione*
- *Primo approccio all’architettura*
- *Le direttive architetturali*
- *Il Two Way Data Binding*
- *Direttive comuni*
- *Filtri*
- *Creare un service*
- *Routing*

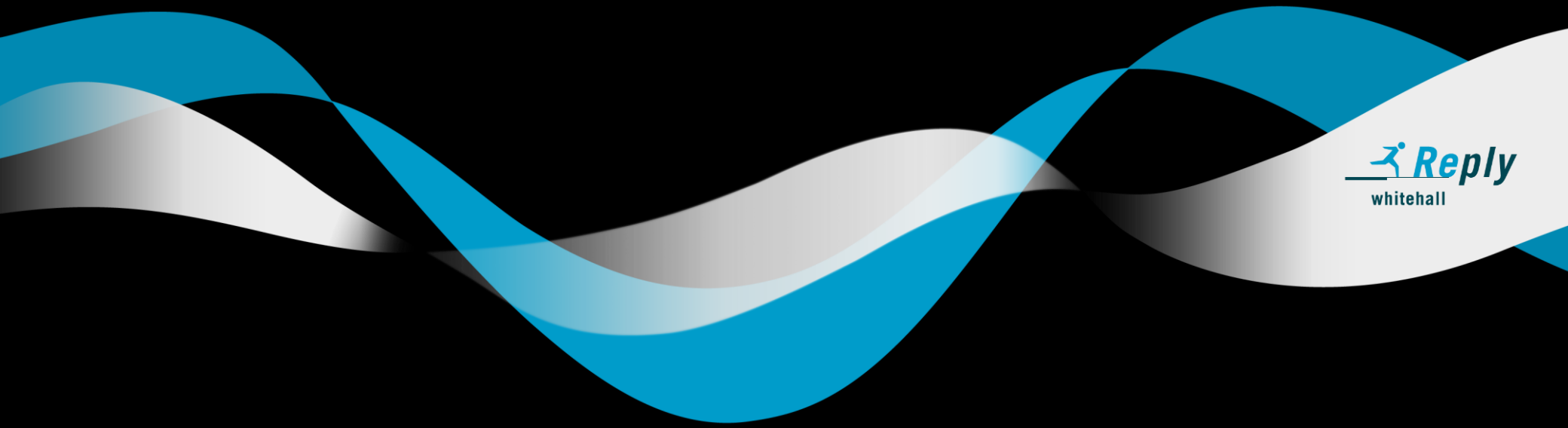
Perché non Angular 2

- La prima release ufficiale è del 14 Settembre 2016
- La conoscenza del framework è ancora poca nella rete
- L’architettura è molto differente dalla versione 1, il filo conduttore è sottilissimo

Università di Roma
Tor Vergata

Un po’ di storia..

Introduzione a JavaScript

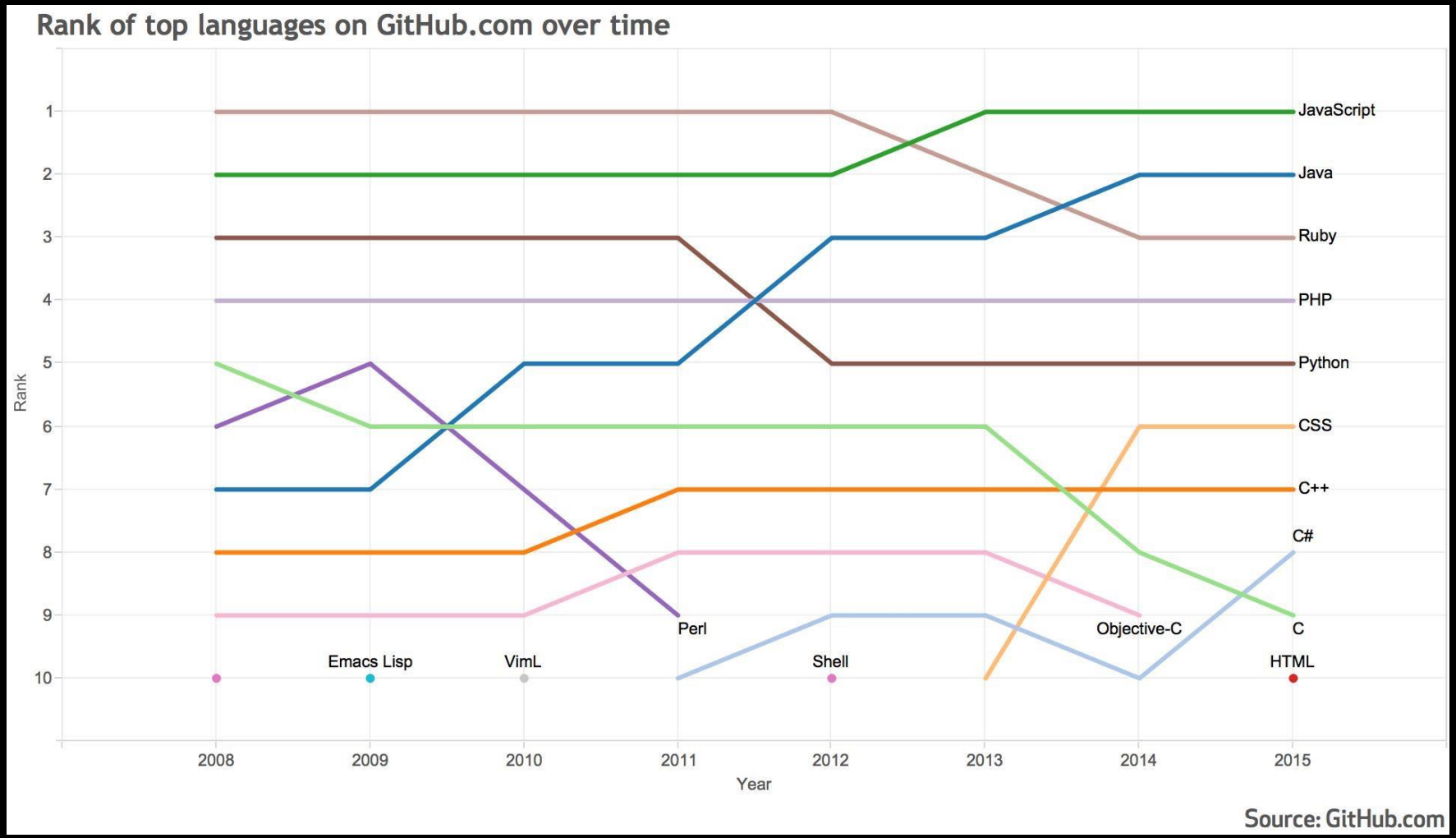


Reply
whitehall

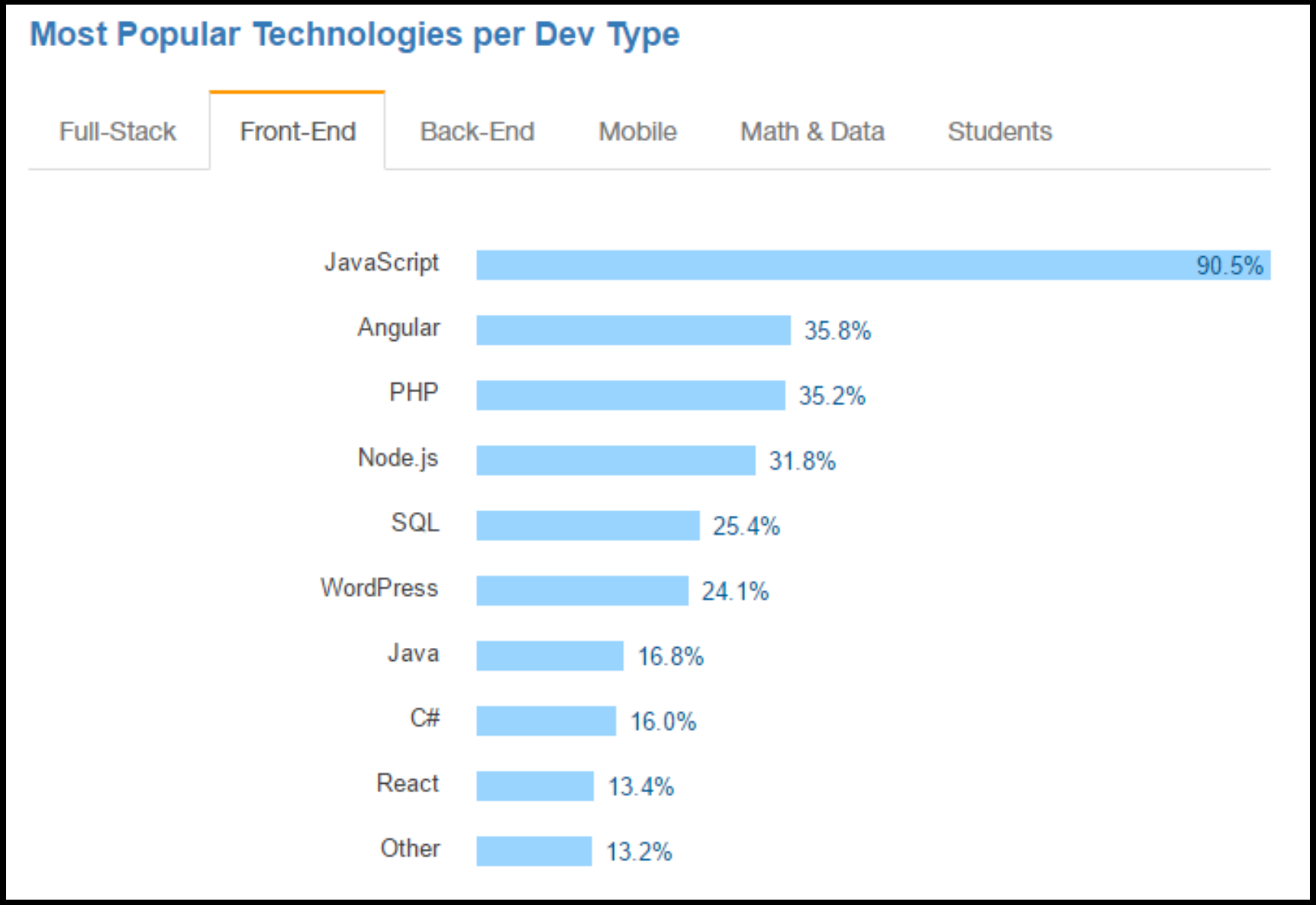
Chi è JavaScript

- E’ un linguaggio di scripting orientato agli oggetti e agli eventi
- Fu sviluppato da Brendan Eich della Netscape Communications nel 1995
- Nome originale: Mochan, poi LiveScript, in seguito JavaScript
- Utilizza una sintassi C-like
- E’ interpretato
- E’ debolmente tipicizzato
- Dalla versione 6 permette il paradigma funzionale
- Standardizzazione: ECMAScript (ECMA-262)
  - ES1: giugno 1997
  - ES2: giugno 1998
  - ES3: dicembre 1999
  - ES4: abbandonato
  - ES5: dicembre 2009
  - ES5.1: giugno 2011
  - ES6: giugno 2015

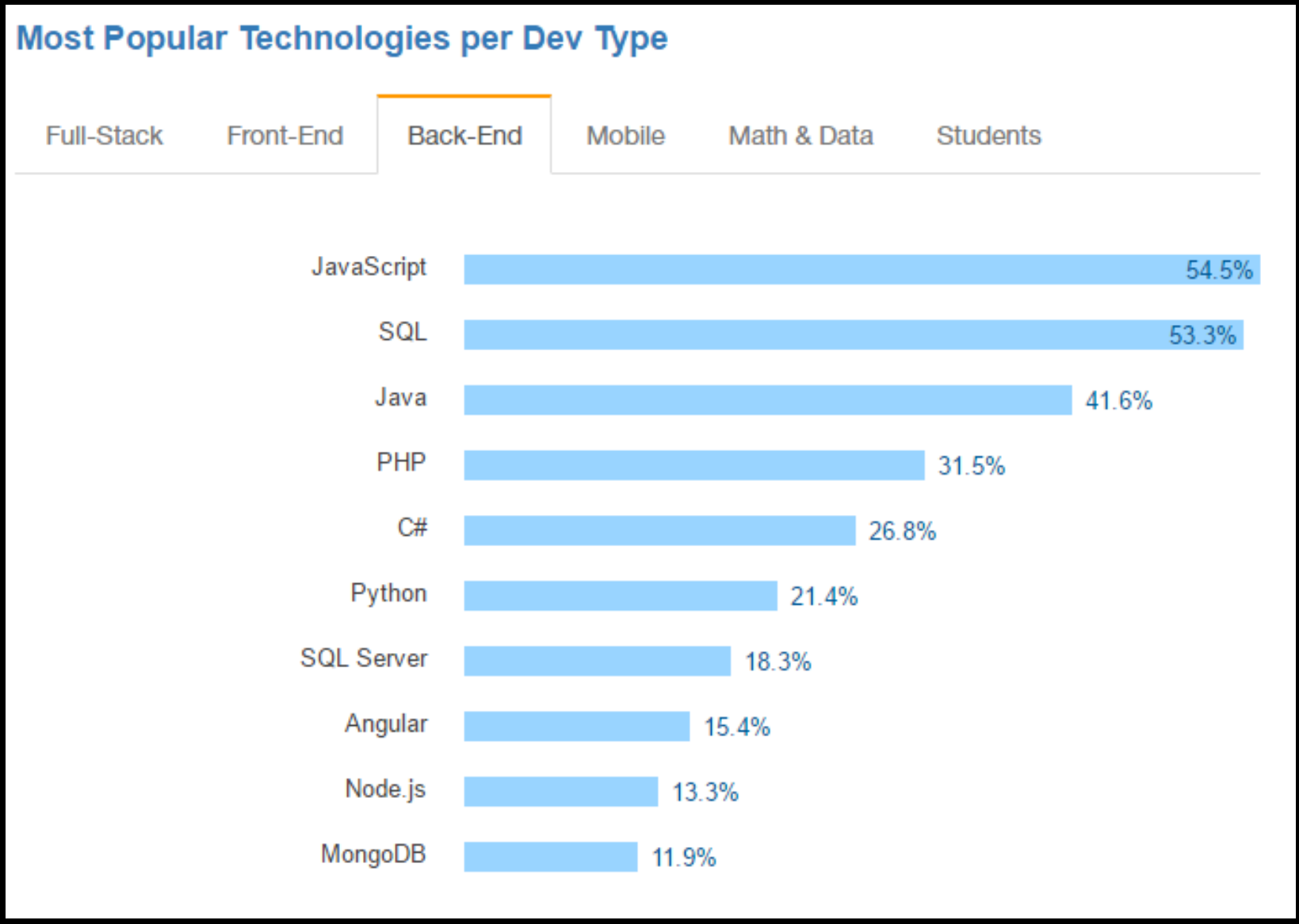
Chi è JavaScript



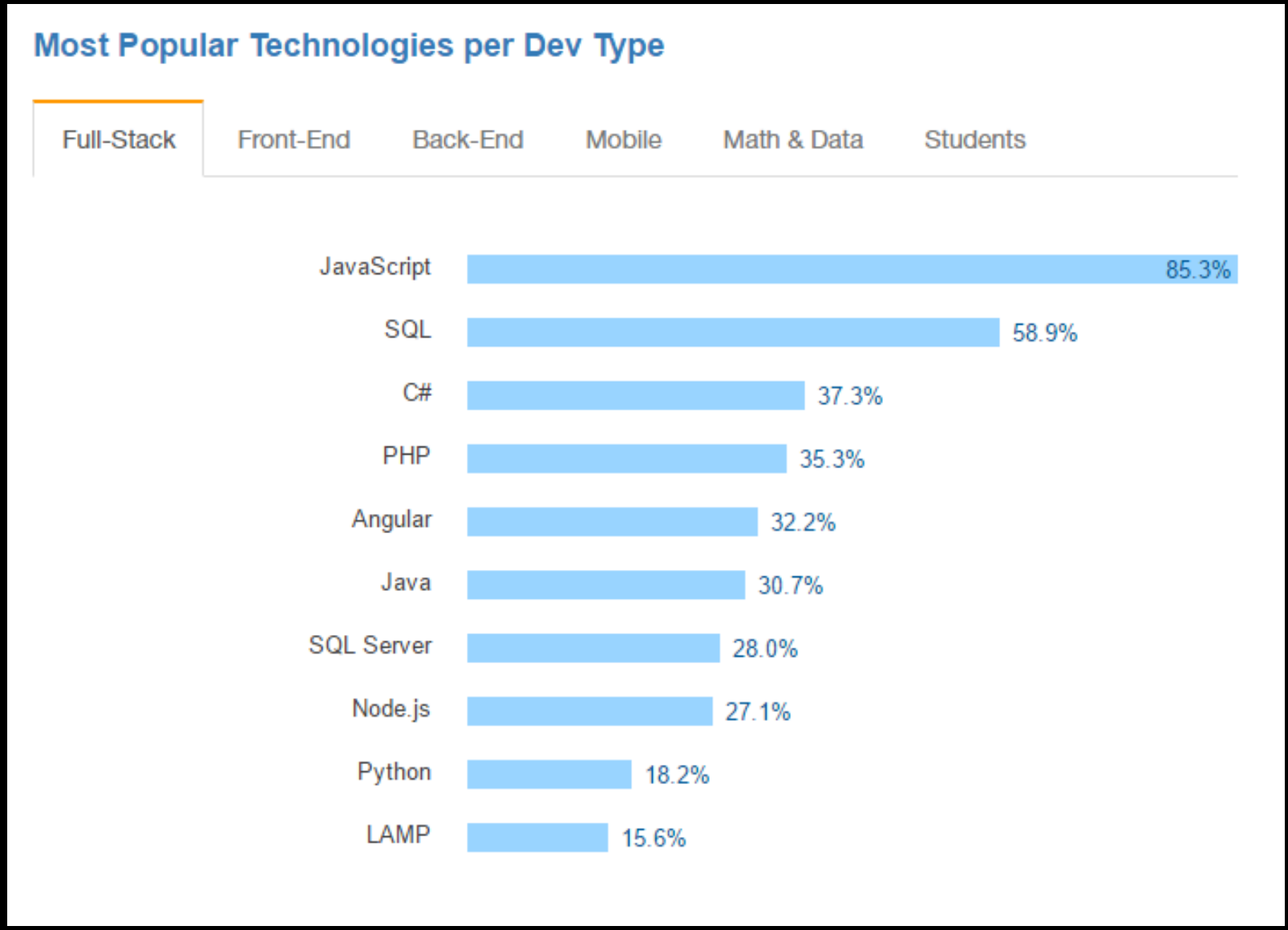
Chi è JavaScript



Chi è JavaScript



Chi è JavaScript




JAVASCRIPT
JAVASCRIPT EVERYWHERE

Problemi

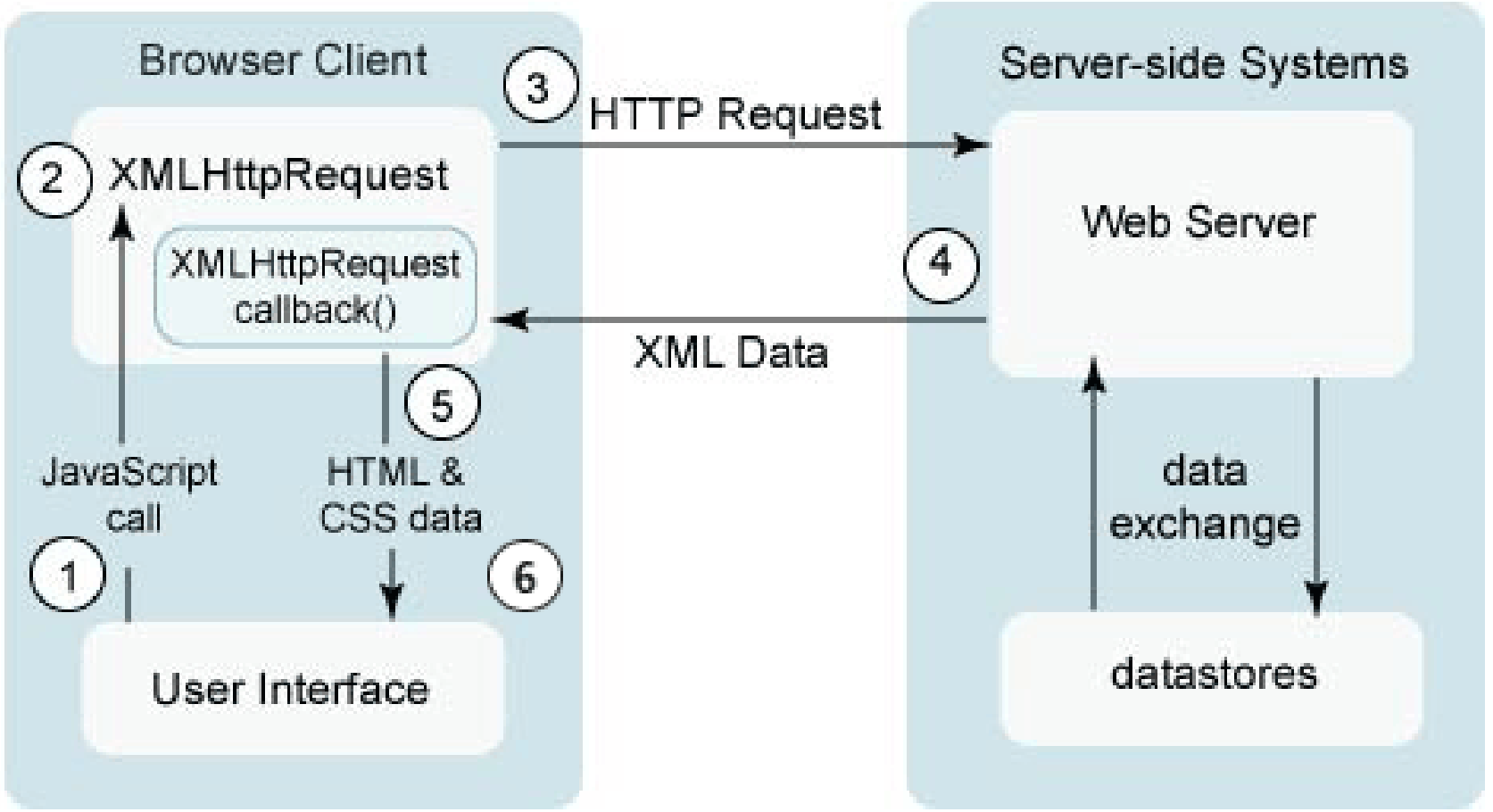
- Tra il 1995 e il 2001 c'è stata la prima guerra tra browser
  - IE vs Netscape
  - Non esisteva una standard di riferimento
  - Esistevano diversi problemi legati sopratutto alla gestione degli eventi
- Il javaScript era lento:
  - I motori del tempo compilavano il codice JavaScript in byte-code per poi interpretarlo
  - Per nulla efficiente

- Questo porta a "scartare" soluzioni progettuali che comportano lo sviluppo di applicazione client-side
- Il ruolo del JavaScript è sostazialmente ridimensionato a funzionalità marginali
- L'architettura di riferimento per il web è il server-side
- Poi arriva jQuery, ma….

Le webapp oggi

- Browser molto più potenti
- L’applicazione è caricata dal browser in fase di avvio:
  - Interamente
  - In maniera asincrona
- La dinamicità dell’app è data da codice JavaScript eseguito nel front-end
- L’applicazione è Single Page
- L’architettura è tipicamente client-side
- Può interaggire con un server (back-end)
  - Nella maggior parte dei casi
  - Nella comunicazione vengono scambiati solo dati (Es. In JSON) e non html

Le webapp oggi



Architettura

- Un’applicazione AngularJS è **CLIENT-SIDE**
- Caratteristiche:
  - L’applicazione viene eseguita sul browser
  - Il server restituisce solo file statici (Html, js, css)
  - Il browser interpreta il js a run-time!
  - Per l’hosting è sufficiente un Web Server
  - Facile la gestione del single-page
  - Per la comunicazione con il server è necessaria un’architettura a servizi:
    - REST
  - Attenzione all’elaborazione dei dati, può danneggiare le performance
  - Il carico di lavoro è sul browser
    - Sono sempre più performanti

Università di Roma
Tor Vergata

AngularJS

Introduzione

Reply
whitehall

Introduzione ad AngularJS

Università di Roma Tor Vergata

«Angular è quello che HTML avrebbe dovuto essere se fosse stato progettato per sviluppare applicazioni»

Introduzione ad AngularJS

- Nato come GetAngular da Misko Hevery e Adam Abrons nel 2009
- Abrons abbandona, Google si interessa al framework e diventa AngularJS
- E’ un progetto Open Source
  - Repository GitHub
- Famoso per implementare il pattern MVW
- Versione 1.0 del giugno 2012 -> 1.6.3 del 2017-03-08
- Versione 2.0

Introduzione ad AngularJS

Università di Roma Tor Vergata

- E’ un riferimento per lo sviluppo Web client-side
- Ha semplificato notevolmente lo sviluppo in JavaScript fornendo un framework in grado di organizzare il codice e separare il compito dei vari componenti
- Fornisce tutti gli strumenti per realizzare una Single Web Application
- Perfetto nelle architetture client-server e nell’interazione attraverso REST
- Strettamente legato all’HTML
- Funzionalità principali:
  - il binding bidirezionale (two-way binding)
  - la dependency injection
  - il supporto ai moduli
  - la separazione delle competenze
  - la testabilità del codice
  - la riusabilità dei componenti

Introduzione ad AngularJS

Università di Roma Tor Vergata

- Supporto al pattern MVC
  - In realtà MVP, anzi MVVM, …
  - Diciamo MVW: Model View “Whatever works for you”

- Il Model è responsabile del mantenimento dei dati, viene gestito attraverso l’oggetto Scope
- La View è responsabile della visualizzazione dei dati, sostanzialmente Html con l’aggiunta di direttive
- Il Controller si occupa dell’interazione tra Model e View
- La logica di businness è realizzata attraverso I service

Università di Roma
Tor Vergata

Università di Roma
Tor Vergata

AngularJS

Componenti architetturali

Reply
whitehall

Componenti

- AngularJS permette di organizzare l'applicazione in moduli:
  - Un App è composta da almeno un modulo
  - Attraverso I moduli è possibile sfruttare al meglio il riutilizzo del codice
  - Realizzare componenti sotto forma di libreria

- Oltre ai moduli esistono I seguenti componenti:
  - Controller
  - Direttive
  - Service
  - Filter
  - View

Module

- ng-app
- E’ un contenitore di componenti
- Ogni componente fa riferimento ad un modulo
- Consente di disaccoppiare le funzionalità
- Da non confondere con uno strumento per caricare librerie come RequireJS
  - AngularJS non ha il compito di caricare file!

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

Controller

- ng-controller
- Controlla l'interazione tra Model e View
- E' il luogo dove viene istanziato lo $scope
  - E' il modello dei dati
  - Sono organizzati in modo gerarchico
    - Ereditarietà: comanda la struttura html
    - La root è rappresentata dal $rootScope
  - Consente di verificare cambiamenti dei dati
- Consente di aggiungere metodi allo $scope
- Sfrutta la Dependency Injection (DI)

```
angular.module("myModule").controller("myController", function($scope) {
      // My code...
});
```

Controller

- Cosa non deve fare un Controller:
  - Manipolare il DOM: ci pensano le direttive
  - Formattare l'input: ci pensano le form
  - Formattare l'output: ci pensano I filtri
  - Condividere oggetti tra controller:
    - Meglio un Service
    - O un pattern Observer
  - Implementare funzionalità generali o comuni
    - Bisogna utilizzare un service
  - Non bisogna sovraccaricare I Controller!!!

  - Esistono anche I controller senza $scope

Data binding

- E’ il meccanismo di sincronizzazione tra I dati e la view
- Permette di aggiornare lo scope in tempo reale
- La maggior parte dei sistemi di template supporta il data binding in una sola direzione:
  - Tipicamente dal modello dei dati verso la view in fase di costruzione
  - Una modifica dei dati non aggiorna la view
  - Una modifica della view non aggiorna i dati
  - Occorre codice dedicato!

- Il data binding di AngularJS invece è bidirezionale (**two-way binding**):
  - Ogni modifica al modello dei dati si riflette automaticamente sulla view e ogni modifica alla view viene riportata sul modello dei dati

Direttive

- Sono dei marcatori che estendono l'Html
- Aggiungono o modificano il comportamento di un elemento Html
- Sono gli unici elementi che sono *autorizzati* a manipolare il DOM
- Possono essere associate a: elementi, classi css, attributi, commenti
- Sono caratterizzare da una sintassi che ha prefisso ng
- Ne esistono diverse predefinite:
  - ng-app
  - ng-controller
  - **ng-model**
  - ng-show
  - ……

```
angular.module("myModule",[])
.controller("myController", function($scope) {
      $scope.name = "Simone";
});
```

```
<div ng-app="myModule">
    <p ng-controller="myController">{{ name }}</p>
</div>
```

- E' possibile creare direttive custom

Service

- Sono dei componenti che offrono funzionalità indipendenti dalla View
- Consentono di implementare la logica dell'applicazione
- Consentono di condividere la funzionalità accessibili da altri componenti (Controller, direttive, filtri,…)
- AngularJS crea un istanza di un service solo quando richiesta da un componente che lo ha come dipendenza
- Un service è un oggetto Singleton
- Possono essere:
  - Forniti dal framework
  - Custom

Service

- Dal framework:
  - Sono circa 30
  - $interval
  - $timeout
  - $http

```
angular.module("myModule", [])
.controller("myController", function($scope,$timeout) {
      $scope.name = "Simone";

      $timeout(function () {
        $scope.name = "Giulia";
      },1000);
});
```

- Custom:
  - Esistono vari metodi di creazione:
    - Provider
    - Service
    - Factory
    - Constant
    - Value
  - Li vediamo successivamente..

Filter

- Sono componenti con il compito di formattare o comunque applicare una elaborazione al risultato di una espressione
- Sono usati principalmente nella View:
  - Attraverso l'operatore pipe

```
<p>{{"Hello Angular" | uppercase }}</p>
```

    - HELLO ANGULAR
- Possono essere combinati tra loro

```
<p>{{"Hello Angular" | uppercase | lowercase }}</p>
```

    - hello angular
- Possono prevedere parametri

```
<p>{{"Hello Angular" | limitTo:5 }}</p>
```

    - Hello

Filter

- Filtri messi a disposizione dal framework:

Filtro	Descrizione
lowercase	Trasforma una stringa in caratteri minuscoli
Uppercase	Trasforma una stringa in caratteri maiuscoli
number	Formatta un numero
currency	Formatta un numero come valuta
date	Formatta una data
orderBy	Ordina gli elementi di un array
limitTo	Estrae i primi n elementi di un insieme (stringa, array)
filter	Estrae gli elementi di un array che soddisfano un determinato criterio

Filter

- E’ possibile utilizzare I filtri anche nel Controller
- Tramite DI aggiungiamo il parametro che identifica il filtro nel nostro Controller
- Richiamiamo il filtro

```
angular.module("myModule", [])
    .controller("myController", function($scope,dateFilter) {
        var dataISO = dateFilter(Date.now(), "yyyy-MM-ddTHH:mm:ssZ");
});
```

- E’ possibile creare filtri custom


VEDO ANGULARJS
MA NON CAPISCO COME FUNZIONA

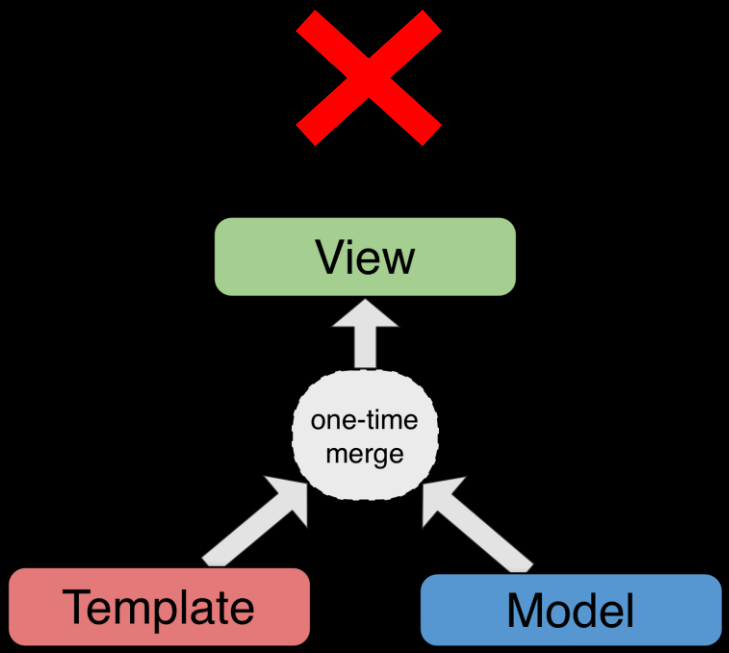
Università di Roma
Tor Vergata

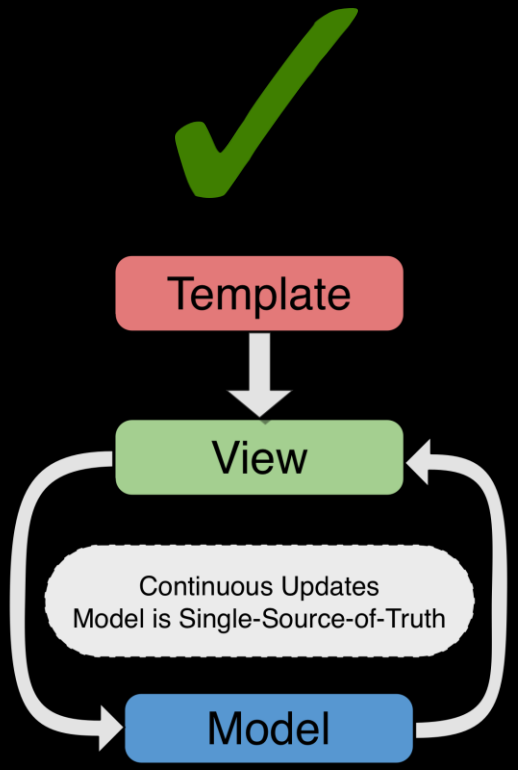
AngularJS

Data Binding

Reply
whitehall

Data binding





Data binding

- Abbiamo visto che è possibile introdurre direttive all'interno della View
- Molte di queste sono già native nel framework
- Come vengono processate?
  - Il browser scorre il DOM e memorizza le direttive: Compilazione
  - Successivamente prende le direttive appartenenti ad uno stesso scope e le inserisce all'interno di una funzione, nella quale vengono istanziati I rispettivi watch: Link
- In questo modo ogni modifica nel modello (scope) si riflette nella vista (direttive) e viceversa

Data binding

- Problemi:
  - Il data-binding è realizzato attraverso gli oggetti watch
  - Questi hanno il compito di verificare cambiamenti sullo scope
  - Come? Mediante polling…
  - Il data-binding è un meccanismo molto oneroso!
  - Se non usato con attenzione può portare ad un drastico deterioramento delle performance

- Ottimizzazioni:
  - Quando è necessario soltanto mostrare un attributo dell'oggetto scope, è possibile indicare ad AngularJS di utilizzare il one-way binding per una certa variabile


GUARDA SIMBA
LAGGIÙ C'È UN USO SBAGLIATO DELL'NG-MODEL
PROMETTIMI DI NON ANDARCI MAI

Università di Roma
Tor Vergata

AngularJS

Le direttive

Reply
whitehall

Direttive

- ng-app
- ng-controller
- ng-module
  - Consente di associare un attributo dello scope ad un elemento html
- ng-show/ng-hide
  - Consente di mostrare o meno un elemento html a secondo della valutazione di un espressione
  - Ng-hide nega la logica
- ng-if
  - Effetto grafico uguale a ng-show
  - Sostanziale differenza: crea e distrugge il DOM, non agisce sui CSS
- ng-switch
  - Seleziona la porzione di interfaccia da visualizzare in base al confronto tra una espressione di controllo ed un elenco di possibili valori

Direttive

- ng-include
  - consente di includere, nella vista corrente, del codice HTML esterno cioè codice che si trova in un altro file
  - *È importante sottolineare che il codice contenuto nel file esterno viene interpretato da Angular prima di effettuare il rendering sulla pagina.*
  - applica la *same origin policy*
- ng-repeat
  - Consente di generare un elenco di elementi HTML a partire da una collezione di dati
- ng-options
  - Consente di aggiungere a partire da un oggetto gli elementi option di una select

Direttive

- ng-click
  - Consente di associare un espressione all'evento click dell'elemento html
- ng-change
  - Consente di associare un espressione all'evento change di un elemento html
- ng-src
  - Prima valuta l'espressione e poi la inserisce all'interno dell'attributo src
  - Risolve fastidiosi problemi grafici
- ng-disabled
  - Consente di aggiungere l'elemento «disabled» secondo la valutazione di un espressione
- ng-init
  - Consente di valutare un espressione all'interno del DOM
  - Usato in casi eccezionali per inizializzare alcune variabili

Direttive

- ng-bind
  - Consente di sostituire il contenuto di un elemento HTML con il risultato della valutazione di un'espressione e di aggiornare tale contenuto quando il valore dell'espressione cambia
  - Ha le stesse finalità delle parentesi graffe {{ }}
- ng-cloack
  - Consente di evitare il fastidioso effetto grafico «flicker», dovuto alla compilazione nel caricamento del DOM
- ng-style
  - Consente di aggiungere proprietà CSS a secondo della valutazione di un espressione
- ng-class
  - Consente di aggiungere classi CSS a secondo della valutazione di un espressione
- *E molte altre…*


SONO GRATIS USIAMOLE!

Università di Roma
Tor Vergata

AngularJS

I service

Reply
whitehall

Service

- Sono dei componenti che offrono funzionalità indipendenti dalla View
- Consentono di implementare la logica dell'applicazione
- Un service è un oggetto Singleton
- Esistono vari metodi di creazione:
  - Provider
  - Service
  - Factory
  - Constant
  - Value
- In realtà ogni tipo di service è una specializzazione di un Provider

Factory

- Presenta una sintassi molto semplice
- Usate quando occorre solamente passare la funzione $get
  - Ovvero non occorrono configurazioni
- Presenta le seguenti caratteristiche:
  - Possibilià di utilizzare altri service
  - Inizzializzazione del service
- Le Factory possono ritornare un valore oppure un oggetto
  - Quindi anche un insieme di metodi per esempio Utility

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

Service

- Caso particolare di Factory
- Il Service ritorna un istanza dell'oggetto
  - Non un valore associato
- E' realizzato definendo una constructor function
- Una volta che richiamiamo il service abbiamo bisogno di invocare un metodo

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

Scambiare dati tra Controller

- Come faccio a condividere dati tra controller?
  - Definisco variabili globali → NO!
  - Utilizzo un pattern observer
  - Utilizzo un Service
    - Sfrutto le proprietà del singleton
    - Definisco l'oggetto che conterrà l'informazione
    - Implemento I metodi get() e set()
- Attenzione al data-binding

Scambiare dati tra Controller


```
angular.module("myApp").service('RoleService', [
  function () {

    this.role = null;

    this.set = function(obj){
      this.role = obj;
    };

    this.get = function(){
      return this.role;
    };

  }
]);
```

Service di Utility

- Voglio implementare un service che ritorna una serie di metodi di utility
- Factory o Service?

```
var app = angular.module('myApp', [])
  .factory('myUtilityFactory', function(){
    return {
      sayHello: function(text){
        return "Factory says \"Hello " + text + "\"";
      },
      sayGoodbye: function(text){
        return "Factory says \"Goodbye " + text + "\"";
      }
    }
});
```

$http

- Permette di effettuare chiamate Ajax
- Il servizio fornisce I metodi che permettono di invocare I corrispettivi metodi http
  - $http.get(url)
  - $http.post(url,data)
  - $http.put(url,data)
  - $http.delete(url)
  - $http.head(url)
- I metodi restituiscono delle promise
  - Consente di rendere asincrono il codice
  - 2 metodi implementati:
    - success()
    - error()

$http

- Spesso è necessario invocare chiamate Ajax passando parametri nell'header oppure specificare configurazioni particolari
- Il servizio permette anche di invocare un metodo generico con il quale personalizzare la chiamata

```
$http({
  method: 'GET',
  url: '/someUrl',
  timeout: 3000,
  headers: {
    'Content-Type': 'application/json'
  },
  params : data
}).then(function successCallback(response) {
  // code here
}, function errorCallback(response) {
  // code here
});
```

Le promise

- Consentono di rendere asincrona una porzione di codice
- Necessario per evitare di bloccare l'applicazione in attesa di particolari elaborazioni
  - Operazioni computazionalmente onerose
  - Wrapping di service in altri service che utilizzano le promise
    - Chiamate Ajax

```
var example_promise = function(data) {
  var deferred = $q.defer();
  if (typeof data != "undefined")
    deferred.resolve(data);
  else {
    deferred.reject({ err : "Error in typeof data"});
  }
  return deferred.promise;
};
```

Università di Roma
Tor Vergata

I SERVICE SONO TROPPO POTENTI

Università di Roma
Tor Vergata

AngularJS

Il routing

Reply
whitehall

SPA

- Single Page Application
- Viene eseguita all'interno di una singola pagina html
  - Tutte le risorse necessarie alla sua esecuzione vengono caricate dinamicamente ed aggiunte al DOM della pagina corrente
- La transazione tra una vista e l'altra da l'illusione di navigare in pagine web differenti
- *Effettivamente la URL cambia...*
- Il meccanismo che consente di mappare la url alla vista è detto Routing

SPA

- Problemi???
- Il concetto di SPA rompe il design di navigazione dei browser
  - Pensiamo ai tasti Avanti o Indietro
  - Come gestire la logica di navigazione?
- Per fortuna attraverso JavaScritp è possibile replicare il comportamento
  - Si utilizza il trucco dell'#
  - Indica al browser che non deve effettuare chiamate http
  - Il JS memorizza gli eventi nella history
  - Ricostruisce la view da visualizzare ricostruendo la funzione Avanti o Indietro
- Html5 ha introdotto l'oggetto window.history in grado di eliminare il trucco dell'#
  - *Ovviamente solo I browser più recenti lo supportano…*

Il Routing

- In AngularJS esiste un modulo che consente di gestire il routing
- Non fa parte del core, quindi occorre importarlo
- Il modulo è ngRoute
- E' necessario configurare il routing in modo da mappare le view alle URL
  - Non solo…
- Il componente più importante è il $routeProvider che si occupa della configurazione attraverso i metodi
  - When()
  - Otherwise()
- Un aspetto importante è rappresentato dalla possibilità di passare parametri tra una view e l'altra

IL ROUTING AIUTA A PROGETTARE

Configurazione

Università di Roma Tor Vergata

```
angular.module("myApp", ["ngRoute"])
    .config(function($routeProvider) {
      $routeProvider
          .when("/", {
            templateUrl: "/templates/home.html",
            controller: "HomeCtrl"
          })
          .when("/utenti", {
            templateUrl: "/templates/listaUtenti.html",
            controller: "listaUtentiCtrl"
          })
          .when("/utenti/:userId", {
            templateUrl: "/templates/dettaglioUtente.html",
            controller: "dettaglioUtenteCtrl"
          })
          .otherwise({redirectTo: "/"});
    });
```

Configurazione

- Una volta configurato il routing è necessario indicare nella pagina html dove iniettare le view
- Si utilizza la direttiva ng-view

```
<body ng-app="myApp">
    <div ng-view></div>
</body>
```

- Il codice viene quindi inserito all'interno del div dove è definita la direttiva
- Quando cambio "route" l'unico codice ad essere sostituito è sempre quello all'interno della direttiva
- La pagina non viene mai ricaricata!!

Cambiare «route»

- Come faccio ad andare in una certa “route”?
- 2 modi… *anzi 3*
  - Attraverso l’utilizzo di href: è necessario specificare la url della view
  - Attraverso il service $location: tipicamente utilizzato all’interno del codice js
    - $location.path(“url/di/destinazione”);
  - *Scrivendo nel browser la url che vogliamo raggiungere*
- La url che inserisco sia in href che nel servizio $location è il path relativo all’applicazione
  - Non “http://ilmiodominio.it/lamiaapplicazione/utenti”
  - Ma “/utenti”

Passare parametri

- Riprendiamo la configurazione del routing

```
.when("/utenti/:userId", {
  templateUrl: "/templates/dettaglioUtente.html",
  controller: "dettaglioUtenteCtrl"
})
```

- I due punti prima di userId indicano che il valore successivo è un parametro
- Per recuperare il parametro si utilizza il service $routeParams

```
angular.module("myApp")
    .controller("dettaglioUtenteCtrl", function($scope, $routeParams) {
      var userId = $routeParams.userId;

    });
```

Riferimenti

- Documentazione ufficiale: https://docs.angularjs.org/guide
- HTML.it: http://www.html.it/guide/guida-angularjs/
- Sullaprogrammazione.com:
  - http://www.sullaprogrammazione.com/passato-e-futuro-di-javascript.html
  - http://www.sullaprogrammazione.com/il-web-dinamico-secondo-angularjs.html
- W3schools: https://www.w3schools.com/angular/
- Wikipedia: https://en.wikipedia.org/wiki/AngularJS
- Github.com: https://github.com/blog/2047-language-trends-on-github
- Stackoverflow.com: http://stackoverflow.com/insights/survey/2016

Grazie e buon AngularJS

Simone Corrieri
s.corrieri@reply.it