

Institut Supérieur d'Informatique

# ANGULARJS

## UN FRAMEWORK

### DE DÉVELOPPEMENT JS LIBRE

Fahem KEBAIR – [kebairf@gmail.com](mailto:kebairf@gmail.com)

# INTRODUCTION

- C'est un framework javascript libre et open-source de développement d'applications web monopages (en anglais single-page-application ou SPA), développé et maintenu par Google depuis 2012 sur une idée de Misko Hevery.
- Une application web monopage est une application web accessible via une page web unique, permettant d'éviter le chargement d'une nouvelle page à chaque demande effectuée et d'améliorer ainsi l'expérience utilisateur.
- AngularJS (AJS) a pour but de simplifier la syntaxe JavaScript et de lui apporter d'autres fonctionnalités. Il se place donc comme un sérieux concurrent pour les librairies JS les plus connues comme JQuery, MooTools, ProtoType,...
- AJS se démarque par sa capacité d'amener une partie du traitement métier côté client, ce qui rend le traitement web plus efficace et plus fluide.
- Il est distribué en tant que fichier JS et peut être appelé dans un fichier HTML ainsi:

```
<script src="//ajax.googleapis.com/ajax/libs/angularjs/1.4/angular.min.js"></script>
```

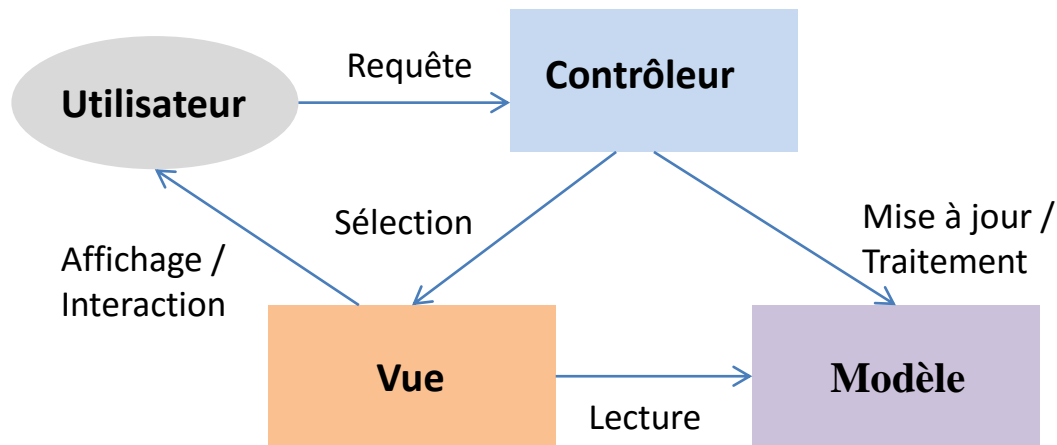
# PRINCIPES ET CARACTÉRISTIQUES

- Il étend le HTML pour le rendre dynamique et permet de développer ses propres balises et attributs HTML.
- Il fonctionne entièrement côté client, donc idéal pour construire des interfaces graphiques et gérer l'interaction avec l'utilisateur, mais pas pour gérer la logique métier qui est souvent implantée côté serveur.
- Il suit le patron de conception MVC (Modèle-Vue-Contrôleur) et en mettant l'accent sur le couplage faible entre la présentation, les données et les composants métiers.
- Il utilise l'injection de dépendance pour gérer les dépendances entre les objets.
- Il fournit une fonctionnalité routage pour gérer les appels de divers composants.
- Il fournit un « *databinding* » pour gérer la lecture et la saisie des données.
- Il permet d'effectuer des tests unitaires sur l'ensemble de l'application.

# CONCEPTS ET DÉFINITIONS

## MODÈLE MVC

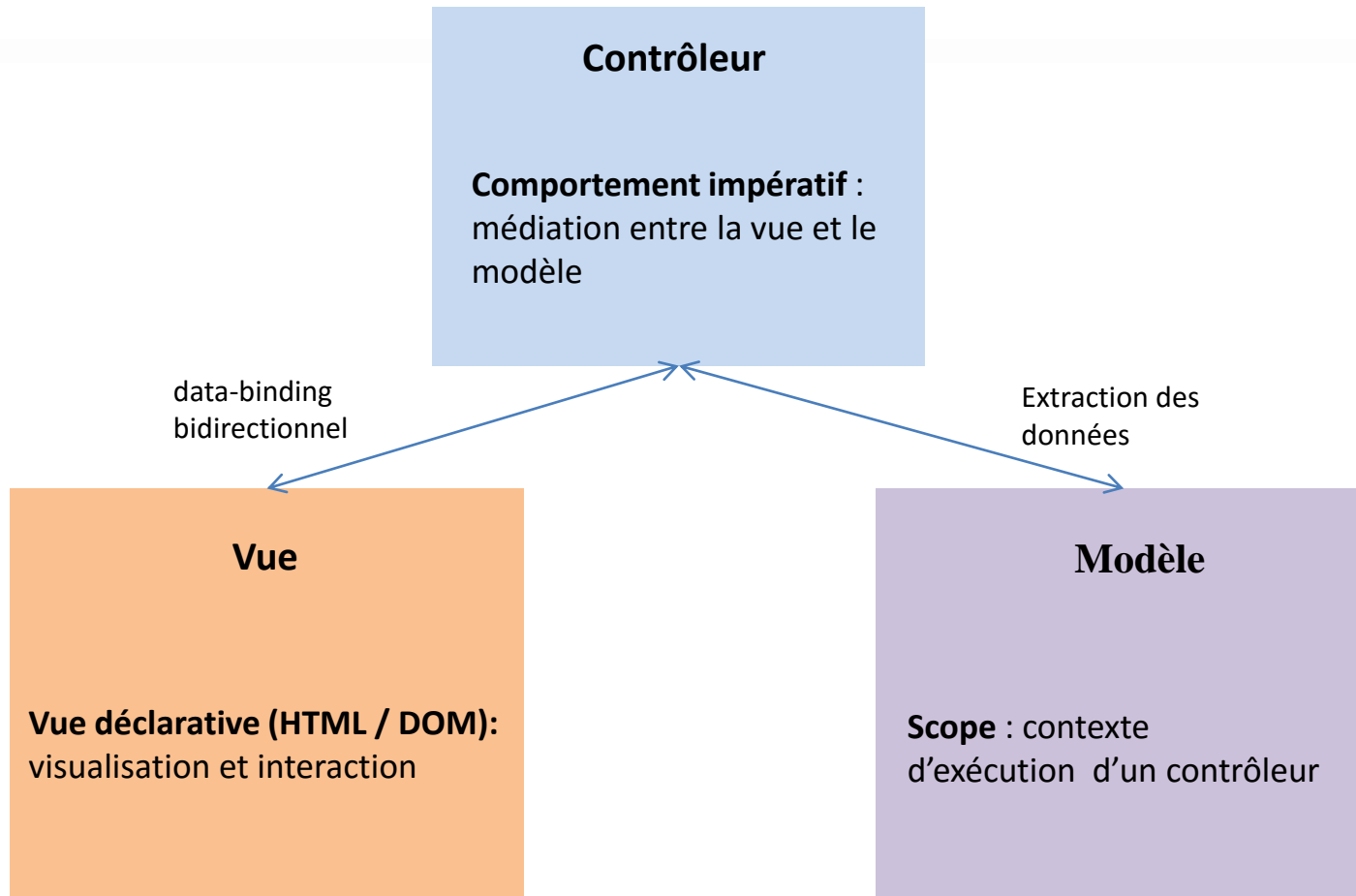
- Le modèle MVC (Modèle-Vue-Contrôleur) est un patron de conception qui aide à développer facilement et de manière efficace des programmes informatiques.
- Avantages du MVC :
  - Facilite et clarifie la conception ainsi que l'implantation d'un programme informatique.
  - Sépare clairement la présentation des données de leur traitement.
  - Permet de séparer les tâches.



- **Modèle (M)** : stocke et gère les données.
- **Vue (V)** : présente les données et permet l'interaction avec l'utilisateur.
- **Contrôleur (C)** : assure le lien entre la vue et le modèle. Il récupère les demandes venues de la vue et les dispatchent au modèle.

# CONCEPTS ET DÉFINITIONS

## MVC DANS ANGULARJS



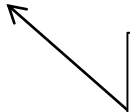
# CONCEPTS ET DÉFINITIONS

## INJECTION DE DÉPENDANCE

- C'est un patron de conception (*Dependency Injection* en anglais) qui est largement utilisé par les frameworks.
- Il a pour but d'éviter une dépendance directe entre deux classes et de définir dynamiquement la dépendance.
- *Explication* : une classe A dépend d'une autre classe B. Si A possède un attribut de type A, instancier A revient à instancier B. Cette tâche sera déléguée à une classe C qui se charge de son instantiation. Dans le contexte des frameworks, ces derniers représentent la classe C généralement.

### Fonctionnement dans AJS

```
<div ng-app="myApp" ng-controller="myCtrl"> Hello {{name}} </div>
<script>
  var myApp = angular.module("myApp", []);
  controller("myCtrl", function ($scope) {
    $scope.name = 'World';
  });
</script>
```



Le contrôleur a une dépendance \$scope, qui est passée en paramètre dans son constructeur. L'instanciation de cet objet a été réalisée par AJS.

# POUR COMMENCER

## EXEMPLE « HELLO WORLD »

- *Exemple* : une page index.html qui affiche dynamiquement « Hello world ! ».

```
<!doctype html>
<html ng-app="helloApp">
<head></head>
<body>
<h1>Introduction à AngularJS</h1>
<div ng-controller="helloCtrl">
  <h2>Hello {{name}} !</h2>
</div>
<script src="angular.js"></script>
<script src="app.js"></script>
</body>
</html>
```

- **ng-app** et **ng-controller** s'appellent des directives, ng-app délimite la zone d'action de l'application et ng-controller celle du contrôleur.
- **app.js** est le fichier d'application dans lequel est défini le contrôleur.

# POUR COMMENCER

## EXEMPLE « HELLO WORLD »

### Fichier « app.js »

```
var helloApp = angular.module('helloApp', []);  
helloApp.controller('helloCtrl', function ($scope) {  
    $scope.name = 'world';  
});
```

- **\$scope** est un objet défini par AJS, c'est le contexte d'exécution d'un contrôleur. Tous les objets définis par AJS commencent par le symbole '\$'.
- L'instanciation du contrôleur prend un paramètre scope, il s'agit d'une injection de dépendance.
- **\$scope.name** : affectation d'une valeur à la variable *name* invoquée dans la vue grâce à **{{ }}**.



# LES EXPRESSIONS

## Expressions

- Une expression AJS est écrite ainsi : {{ expression }}
- Elle peut contenir des variables ou des opérateurs

Ex : {{ 5 + 5 }} ou {{ firstname + " " + lastname }}

- Une expression peut être écrite aussi à l'aide de la directive **ng-bind** qui a le même effet que les accolades. Toutefois, il est recommandé d'utiliser **ng-bind** qui évite l'affichage momentané des accolades au moment de la compilation.

*Exemple :*

```
<div ng-app>
  <p>Hello <span ng-bind="name"></span></p>
</div>
```

## Nombres

```
<div ng-app ng-init="quantity=1;cost=5">
  <p>Total in dollar: {{ quantity * cost }}</p>
</div>
```

# LES EXPRESSIONS

## Strings

```
<div ng-app ng-init="firstName='John';lastName='Doe'">
  <p>The name is {{ firstName + " " + lastName }}</p>
</div>
```

## Objets

```
<div ng-app
  ng-init="person={firstName:'John',lastName='Doe'}">
  <p>The name is {{ person.lastName }}</p>
</div>
```

## Tableaux

```
<div ng-app
  ng-init="points=[1,10,15,32]">
  <p>The name is {{ points[2] }}</p>
</div>
```

# LES DIRECTIVES

- Les directives AJS sont une extension des attributs de HTML ayant un préfixe **ng**.
- **ng-app** : initialise une application AJS et définit l'élément HTML qui détient l'application.
- **ng-init** : initialise une donnée.
- **ng-model** : lie la valeur d'un élément de contrôle HTML (input, select, textarea) à une donnée de l'application.

*Exemple :*

```
<div ng-app ng-init="firstName='John'">  
  <p>Name: <input type="text" ng-model="firstName"></p>  
  <p>You wrote: {{ firstName }}</p>  
</div>
```

Ici, le détenteur de l'application AJS est le div.

# LES DIRECTIVES

## Data Binding

- Synchroniser les données entre le modèle et les composants de la vue, ex : `{{ firstname }}`.
- La synchronisation se fait ainsi : **`ng-model="firstname"`**

```
<div ng-app ng-init="quantity=1;price=5">
  Quantity: <input type="number" ng-model="quantity">
  Costs:    <input type="number" ng-model="price">
  Total in dollar: {{ quantity * price }}
</div>
```

*Two way databinding.*  
L'élément HTML lit  
et change une donnée  
du modèle.

## Répéter des éléments HTML

- **`ng-repeat`** : Répéter des éléments HTML,  
qui est utile quand on lit des données d'une BD.

```
<div ng-app ng-init="names=['Jani','Hege','Kai']">
  <ul>
    <li ng-repeat="x in names">
      {{ x }}
    </li>
  </ul>
</div>
```

*One way databinding,* l'élément HTML lit  
une donnée du modèle seulement.

# LES CONTRÔLEURS

## DÉFINITION

- Un contrôleur AJS est un objet JS ayant pour but de faire le lien entre le modèle du domaine et la vue.
- Les contrôleurs sont déclarés à l'aide de **ng-controller** et sont nommés selon cette convention « *nomCtrl* »
- Le contexte d'exécution d'un contrôleur est défini par un objet AJS scope. **\$scope** représente le modèle de l'application. Il détient ainsi les variables et les fonctions.

### Ce que le contrôleur doit faire

- Initialiser l'objet \$scope et ses objets.
- Définir le comportement de l'objet \$scope.

### Ce que le contrôleur ne doit pas faire

- Manipuler le DOM. Le contrôleur doit se focaliser sur la logique métier et non sur la présentation des données.
- Partager du code entre des contrôleurs. Ceci est le rôle des services.
- Gérer le cycle de vie d'autres composants. Par exemple la création de services.

# LES CONTRÔLEURS

## TRAITEMENT DES DONNÉES PAR UN CONTRÔLEUR

*Exemple 1 :*

- *personController* est l'objet contrôleur.
- L'objet *person* a deux propriétés *firstName* et *lastName*.
- Dans AJS, tout objet du modèle est accessible à travers l'objet *\$scope*.

```
<div ng-app="myApp" ng-controller="personCtrl">
  First Name: <input type="text" ng-model="firstName">
  Last Name: <input type="text" ng-model="lastName">
  Full Name: {{firstName + " " + lastName}}
</div>
<script>
  angular.module('myApp', []).
    controller('personCtrl', function($scope) {
      $scope.firstName = "John";
      $scope.lastName = "Doe";
    });
</script>
```

Une autre écriture d'instanciation du contrôleur :

```
angular.module('myApp', []).
  controller('personCtrl', ['$scope', function($scope) {
    ...
  }]);
```

# LES CONTRÔLEURS

## TRAITEMENT DES DONNÉES PAR UN CONTRÔLEUR

*Exemple 2 :*

- Le même exemple avec un script JS écrit dans un fichier externe.
- De manière générale les scripts JS sont écrits dans des fichiers externes.

```
<div ng-app="myApp" ng-controller="personCtrl">
  First Name: <input type="text" ng-model="person.firstName">
  Last Name: <input type="text" ng-model="person.lastName">
  Full Name: {{person.firstName + " " + person.lastName}}
</div>
<script src="personCtrl.js"></script>
```

# LES CONTRÔLEURS

## APPLICATION AVEC PLUSIEURS CONTRÔLEURS

*Exemple 3 :*

```
<body ng-app="myApp" >
  <div ng-controller="personCtrl">
    First Name: <input type="text" ng-model="firstName">
    Last Name: <input type="text" ng-model="lastName">
    Full Name: {{firstName + " " + lastName}}
  </div>
  <div ng-controller="professionCtrl">
    Profession: {{profession}}
  </div>
<script>
  var app = angular.module('myApp', []);
  app.controller('personCtrl', function($scope) ){
    $scope.firstName = "John";
    $scope.lastName = "Doe";
  }
  app.controller('professionCtrl', function($scope) ){
    $scope.profession = "Student";
  }
</script>
```



# LES PORTÉES (SCOPE)

## DÉFINITION

- La portée *\$scope* est un objet AJS qui représente le modèle de l'application, englobant ainsi les fonctions ainsi que les données de l'application.
- Il crée le lien entre le contrôleur et le modèle.
- Les portées sont hiérarchiques par nature et suivent la structure DOM de l'application AJS.
- AJS définit deux portées **\$scope** et **\$rootScope**.
- *\$scope* : lie un élément DOM au modèle et aux fonctions définies dans le contrôleur.
- *\$rootScope* : est la racine de tous les scopes

# LES PORTÉES (SCOPE)

## EXEMPLE

```
<body ng-app="myApp">
  <div ng-controller="Ctrl1">
    Hello {{msg}}!
    Hello {{name}}! (rootScope)
  </div>    <br />
  <div ng-controller="Ctrl2">
    Hello {{msg}}!
    Hey {{myName}}!
    Hi {{name}}! (rootScope)
  </div>
<script src="angular.js"></script>
<script>
  var app = angular.module('myApp', []);
  app.controller('Ctrl1', function ($scope, $rootScope) {
    $scope.msg = 'World';
    $rootScope.name = 'AngularJS';
  });
  app.controller('Ctrl2', function ($scope, $rootScope) {
    $scope.msg = 'Dot Net Tricks';
    $scope.myName = $rootScope.name;
  });
</script>
</body>
```

Hello World!  
Hello AngularJS! (rootScope)

Hello Dot Net Tricks!  
Hey AngularJS!  
Hi AngularJS! (rootScope)

- Les données du modèles d'une portée \$rootScope sont accessibles par tous les contrôleurs de l'application AJS.
- Avec \$scope, le contrôleur ne peut voir que les données de son \$scope.

# LES MODULES

## DÉFINITION

- Un module définit une application AJS. C'est un conteneur des différents composants d'une application comme : les contrôleurs, les services, les filtres, les directives,...
- Les contrôleurs doivent appartenir toujours à un module.
- Le module permet à l'application d'être mieux lisible, de garder l'espace de nom global propre et de réutiliser le code.

*Exemple : un module avec un seul contrôleur*

```
<!DOCTYPE html>
<html>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
  {{ firstName + " " + lastName }}
</div>
<script src="angular.js"></script>
<script>
  angular.module('myApp', []).controller('myCtrl', function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
  })
</script>
</body>
</html>
```

- **angular.module('myApp', [])** permet de créer ou de réécrire le module myApp.
- **angular.module('myApp')** permet de retrouver un module.
- Le paramètre [] est utilisé pour définir les dépendances du module (ex: autres modules).
- **.controller** est une méthode permettant la création d'un contrôleur. Le module dispose d'autres méthodes, comme : **factory(...)**, **filter(...)**, **config(...)**, **animation(...)**,....

# LES MODULES

## DÉFINITION

*Exemple* : sortir le script du module et celui du contrôleur dans deux fichiers JS distincts

```
<!DOCTYPE html>
<html>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
  {{ firstName + " " + lastName }}
</div>
<script src="angular.js"></script>
<script>myApp.js</script>
<script>myCtrl.js</script>
</body>
</html>
```

var app = angular.module('myApp', [])

app.controller('myCtrl', function(\$scope) {  
 \$scope.firstName: "John";  
 \$scope.lastName: "Doe";  
})

# LES SERVICES

## DÉFINITION

- Un service est un objet JS qui fournit une fonctionnalité dans toute l'application.
- Les services AJS sont :
  - Instanciés que lorsqu'un composant en dépend.
  - Des singletons, ainsi chaque composant dépendant reçoit l'unique instance générée par le créateur de services.
- Le module AJS peut créer des services avec trois méthodes différentes : *factory*, *service* et *provider*. L'objet obtenu par les trois méthodes est le même, la manière avec laquelle il est créé et géré par contre est différente.
- La manière la plus simple de créer un service est à travers la méthode *Module.factory*, que nous prenons comme exemple.

# LES SERVICES

## EXEMPLE

- *Exemple* : un service de notification injecté dans un contrôleur

```
<div ng-controller="MyCtrl">
  <p>Let's try this simple notify service, injected into the controller...</p>
  <input ng-init="message='test'" ng-model="message" >
  <button ng-click="callNotify(message);">NOTIFY</button>
</div>
<script>
  var app = angular.module('myServiceModule', []);
  app.controller('MyCtrl', ['$scope', 'notify', function ($scope, notify) {
    $scope.callNotify = function(msg) {
      notify(msg);
    };
  }]);
  app.factory('notify', ['$window', function(win) {
    return function(msg) {
      win.alert(msg);
    };
  }]);
</script>
```

- **notify** : service injecté dans le contrôleur
- **\$window**: référence à la fenêtre du navigateur, passé comme paramètre lors de l'instanciation du service *notify*.

Let's try this simple notify service, injected into the controller...

# LES SERVICES

## EXEMPLE

- *Exemple* : un service de notification injecté dans deux contrôleurs différents

```
<script>
  var app = angular.module('myServiceModule', []);
  app.controller('MyController1', ['$scope', 'notify', function ($scope, notify) {
    $scope.callNotify = function(msg) {
      notify(msg);
    };
  }]);
  app.controller('MyController2', ['$scope', 'notify', function ($scope, notify) {
    $scope.callNotify = function(msg) {
      notify(msg);
    };
  }]);
  app.factory('notify', ['$window', function(win) {
    return function(msg) {
      win.alert(msg);
    };
  }]);
</script>
```

# REQUÊTES AJAX DANS AJS

- Les requêtes AJAX sont des requêtes HTTP standards effectuées par le service **\$http**.

*Exemple* : afficher des données récupérées à partir d'une requête HTTP.

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

```
  <table>
```

```
    <tr ng-repeat="x in names">
```

```
      <td> {{ x.Name }} </td>
```

```
      <td> {{ x.Country }} </td>
```

```
    </tr>
```

```
  </table>
```

```
</div>
```

```
<script>
```

```
angular.module('myApp', []).controller('ajaxCtrl', function ($scope, $http) {
```

```
  $http.get("http://www.w3schools.com/website/Customers_JSON.php")
```

```
    .success(function(response) {
```

```
      $scope.names = response;
```

```
    });
```

```
}
```

```
</script>
```

- Les méthodes **success** et **error** prennent en paramètres respectivement les données venant du serveur et un string décrivant le problème.
- Si la réponse reçue est en format JSON, AJS la parse, la transforme en objets JS et l'envoie automatiquement à la méthode success.



# LES FORMULAIRES

- Un formulaire AJS est une collection d'éléments de contrôle HTML : *input*, *select*, *button* et *textarea*.
- Les formulaires dans AJS fournissent un service de validation côté client pour une meilleure expérience utilisateur.

# LES FORMULAIRES

*Exemple : ng-model* relie les éléments *input* aux données du modèle,

- l'objet du modèle *master* a la valeur {firstName:"Joh", lastName:"Doe"} (dans form.js),
- la fonction *reset* copie la valeur de *master* dans celle de *user*.

## index.html

```
<!doctype html>
<html lang="en">
<head>
  <script src=" angular.js"></script>
  <script src="app.js"></script>
</head>
<body>
<div ng-app="myApp" ng-controller="formCtrl">
  <form novalidate>
    First Name:<input type="text" ng-model="user.firstName">
    Last Name:<input type="text" ng-model="user.lastName">
    <button ng-click="reset()">RESET</button>
  </form>
  <p>form = {{user | json}} </p>
  <p>master = {{master | json}} </p>
</div>
```

First Name:

Last Name:

RESET

form = {"firstName":"John","lastName":"Doe"}

master = {"firstName":"John","lastName":"Doe"}

- **{{user | json}}** : un filtre prédéfini de type json est appliqué à l'objet JS user afin d'avoir un format de sortie JSON.
- **novalidate** empêche la validation native du navigateur.

## LES FORMULAIRES

```
<script>
  angular.module('myApp', [])
    .controller('formCtrl', function($scope) {
      $scope.master = {firstName:"John", lastName:"Doe"};
      $scope.reset = function() {
        $scope.user = angular.copy($scope.master);
      };
      $scope.reset();
    });
}
</script>

</body>
</html>
```

# LES FORMULAIRES

## VALIDATION

- AJS fournit une implémentation basique pour la majorité des composants HTML (text, number, url, email, date, radio, checkbox), ainsi que d'autres directives de validation (required, pattern, minlength, maxlength, min, max).

# LES FORMULAIRES

## VALIDATION

### *Exemple : page html*

```
<div ng-controller="defaultCtrl">
  <form name="myForm" novalidate ng-submit="addUser(newUser)">
    <div>
      <div>
        Name : <input name="userName" type="text"
          required ng-model="newUser.name">
      </div>
      <div>
        Email: <input name="userEmail" type="email"
          required ng-model="newUser.email">
      </div>
      <div>
        <input name="agreed" type="checkbox"
          ng-model="newUser.agreed" required> I agree to...
      </div>
      <button type="submit" ng-disabled="myForm.$invalid"> Submit </button>
    </div>
    Message: {{message}}
    Valid: {{myForm.$valid}}
  </div></div></form></div>
```

- **ng-disabled="myForm.\$invalid"**: directive qui désactive le bouton tant que le formulaire est invalide (false).
- **{{myForm.\$valid}}** : affiche la valeur de la propriété \$valid du formulaire (true ou false)

# LES FORMULAIRES

## VALIDATION

*Exemple : (suite – script)*

```
<script> angular.module("exampleApp", [])
    .controller("defaultCtrl", function ($scope) {
        $scope.addUser = function (userDetails) {
            $scope.message = userDetails.name +
                " (" + userDetails.email + ") (" + userDetails.agreed + ")";
        }
        $scope.message = "Ready";
    });
</script>
```

Name:

Email:

☒ I agree to the terms and conditions

Message: toto (toto@gmail.com) (true)

Valid: true

**Formulaire valide**

Name:

Email:

☐ I agree to the terms and conditions

Message: Ready

Valid: false

**Formulaire invalide**

# APPLICATION MONOPAGE ET ROUTAGE

- Une application web monopage (*single one page*) est une application contenant une seule page HTML et utilisant des requêtes AJAX afin de rafraichir et mettre à jour cette page, sans toutefois recharger la page entière.
- AJS dispose d'un système de routage permettant de charger de manière asynchrone des vues/templates HTML selon les URLs.
- L'usage du routage nécessite un module optionnel se trouvant la bibliothèque angular-route.js.

```
<script src="angular-route.js"></script>
```

- Les composants :
- **ngRoute** : module faisant partie de bibliothèque permettant de définir le système de routage.
- **\$route** : un service appartenant au module ngRoute dont l'objectif est de mapper des URLs à des templates HTML.
- **\$routeProvider** : responsable de la configuration.
- **\$location** : un service qui parse l'URL affichée dans la barre d'adresse du navigateur.

Ex : `$location.path = http://localhost:5000/create`

# APPLICATION MONOPAGE ET ROUTAGE

- Une application web monopage (*single one page*) est une application contenant une seule page HTML et utilisant des requêtes AJAX afin de rafraichir et mettre à jour cette page, sans toutefois recharger la page entière.
- Avantages
- Application web semblable à une application native
- Modularisation et maintenabilité du code
- Minimiser l'usage du DOM
- Supporte les templates



# APPLICATION MONOPAGE ET ROUTAGE

- Exemple d'un script de routage :

```
angular.module("exampleApp", ["ngRoute"])
.constant("baseUrl", "http://localhost:5000/products/")
.config(function ($routeProvider, $locationProvider) {
    $routeProvider.when("/list", {
        templateUrl: "/tableView.html"
    });
    $routeProvider.when("/edit", {
        templateUrl: "/editorView.html"
    });
    $routeProvider.when("/create", {
        templateUrl: "/editorView.html"
    });
    $routeProvider.otherwise({
        templateUrl: "/tableView.html"
    });
})
.controller("defaultCtrl", function ($scope, $http, $resource, baseUrl) {
    // .....
});
```

On peut spécifier un contrôleur pour chaque route :

```
$routeProvider.when("/list", {
    templateUrl: "/tableView.html«
    controller: "TableCtrl"
});
```

# APPLICATION MONOPAGE ET ROUTAGE

- Exemple (suite) : affichage d'une vue sélectionnée
- La directive *ng-view* affiche la vue sélectionnée par rapport à l'URL saisie.

```
<body ng-controller="defaultCtrl">  
  <div class="panel panel-primary">  
    <h3 class="panel-heading">Products</h3>  
    <div ng-view></div>  
  </div>  
</body>
```

