



# AngularJS

*Jour 2 - Quick overview & Scopes, templates*

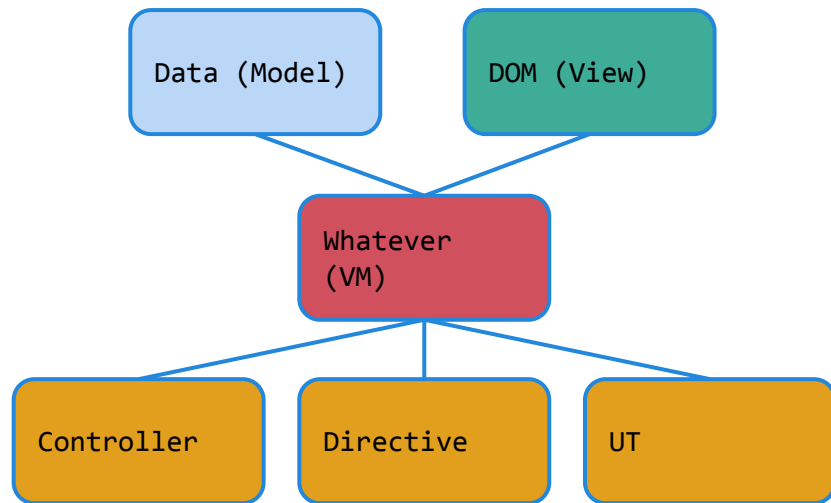
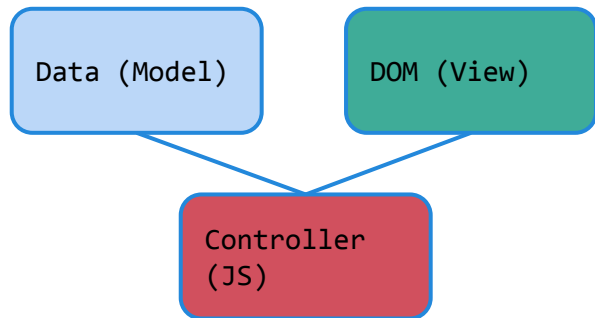
# Au programme

- Quick overview des concepts AngularJS
- Le scope
- Le cycle de vie AngularJS
- Les templates
- Les filtres

# 1 - Structure et core concepts

# Architecture MVW

- MVC vs (MVW or MVVM) (<https://plus.google.com/+AngularJS/posts/aZNVhj355G2>)



# Architecture MVW

Data (Model)

Structure du modèle métier

```
this.model = {  
  "firstname": "Benjamin",  
  "lastname": "Longearet"  
};
```

DOM (View)

Représentation (HTML)

```
<div ng-app>  
  {{model.firstname}}  
</div>
```

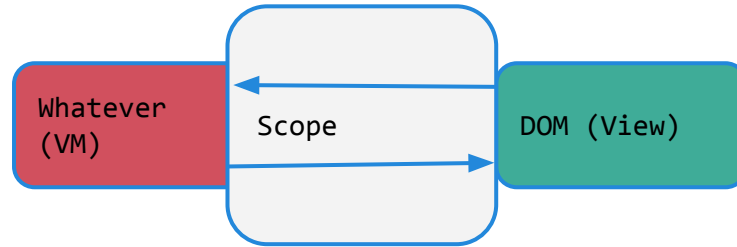
Contrôleur

Whatever - Code JS qui contrôle le flow de l'application - View-Model

```
var myWhatever = function (myService) {  
  this.user = myService.getUsers();  
};
```

# Le scope

- Responsable du dirty-checking
- Closure entre la vue et le contrôleur



# Le contrôleur

## - Logique de la vue

```
var myApp = angular.module('spicyApp1', []);
myApp.controller('SpicyController', ['$scope', function($scope) {
    this.spice = 'very';

    this.chiliSpicy = function() {
        this.spice = 'chili';
    };

    this.jalapenoSpicy = function() {
        this.spice = 'jalapeño';
    };
}]);
```

# Les vues

- HTML amélioré
- Utilisées à différents niveaux :
  - Composant built-in (ngName  $\Rightarrow$  ng-name)
  - Composant personnalisée
  - Routing

```
<div ng-app>  
  {{model.firstname}}  
  <input type="text" ng-model="model.firstname" />  
</div>
```

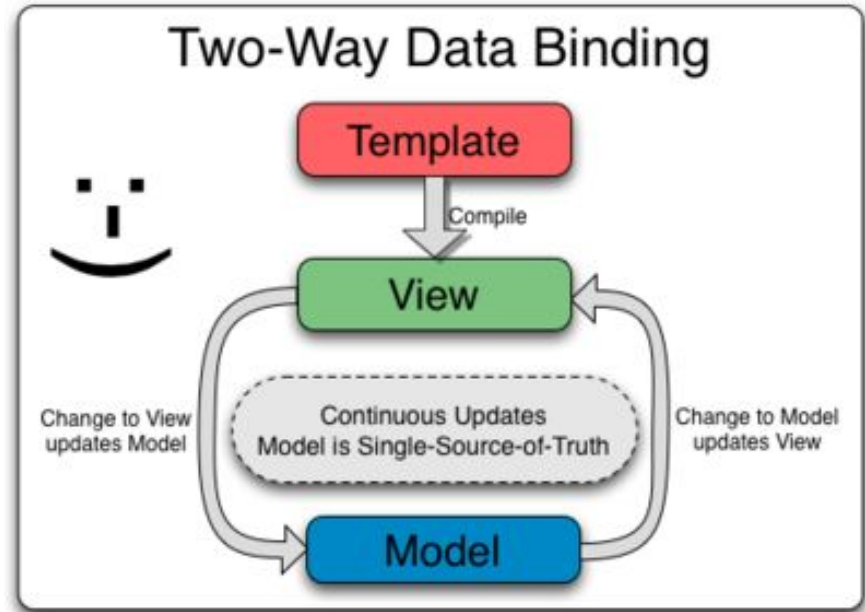
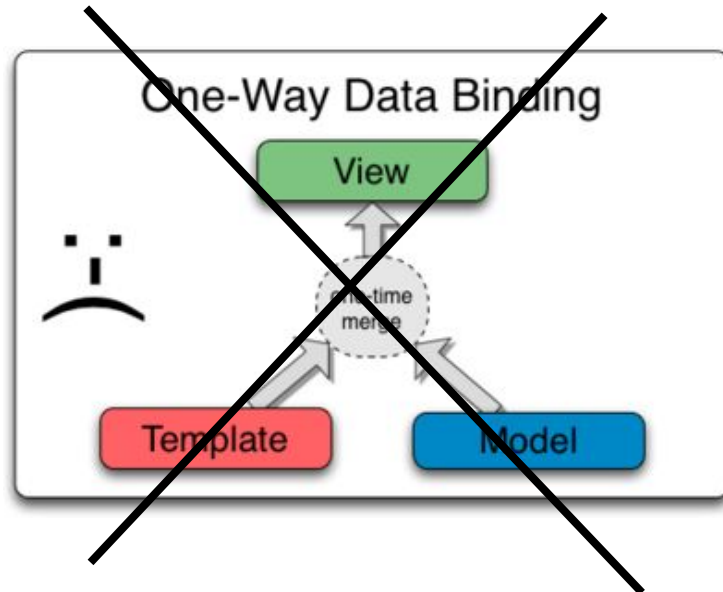


# Le routing

- Routing AngularJS
  - Trop simple, vieux (AngularJS 1.0)
- **UI-Router** ★
  - Multi-vues
  - Héritages
  - Built-in directive
- Nouveau router AngularJS (beta - AngularJS 2.0)

# Data Binding

- Automatiser les changements



# Composants *(anciennement directive)*

- Créer des composants **autonome** et **réutilisable**
- Marqueur HTML (element, class, attributes, etc.)
- Beaucoup de composant native  
ng-if, ng-show, ng-model, ng-repeat, etc.

# Filters

- Des “pure functions”
- Mute une valeur
- Beaucoup de filtres natifs  
limitTo, json, currency

```
<div ng-app ng-init="amount = 1234.56">  
  {{amount | currency:"USD$"}} // USD$1,234.56  
  {{amount | currency:"USD$":0}} // USD$1,235  
</div>
```

# Services

- factory, service, provider, constant, value
- Singleton
- Logique métier
- Discussion serveur
- Beaucoup de services natifs  
\$q, \$http, \$service, \$timeout, etc.

# L'injection dépendance

- *Dependency Injection*
- Importer les services par leur nom
- Facilite le testing et les dépendances

# La communication avec le serveur

- Besoin de données (côté client)
- Pas trop  $\Rightarrow$  sécurité
- Service de requêtage
  - \$http
  - \$resource
  - Restangular

# Testing

- Deux types de tests :
  - Tests unitaires (karma)
  - Tests e2e (protractor)
- Browser ViewLess
  - PhantomJS





# Testing

- Deux types de tests :
  - Tests unitaires (karma)
  - Tests e2e (protractor)
- Browser ViewLess
  - PhantomJS



**Démo: démarrer avec  
le testing!**



**KEEP  
CALM  
AND  
HAVE A  
BREAK**

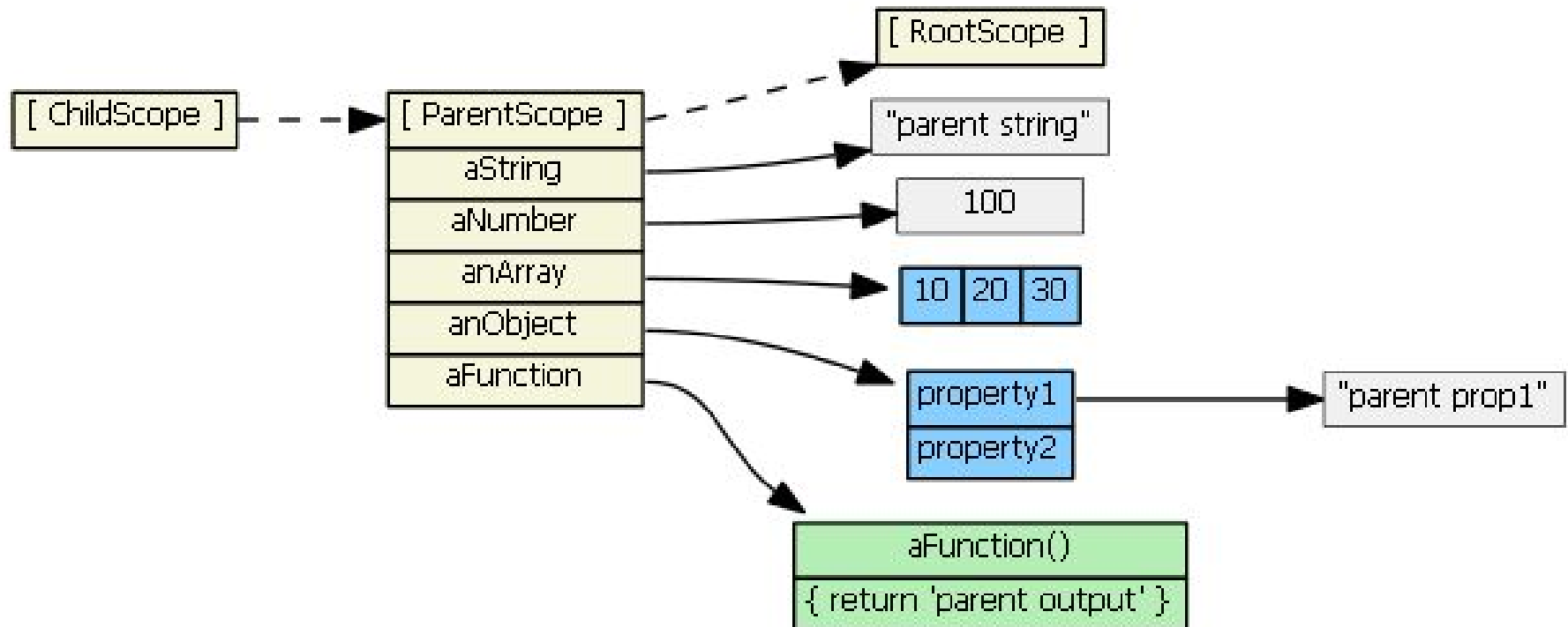


## 2 - Le scope

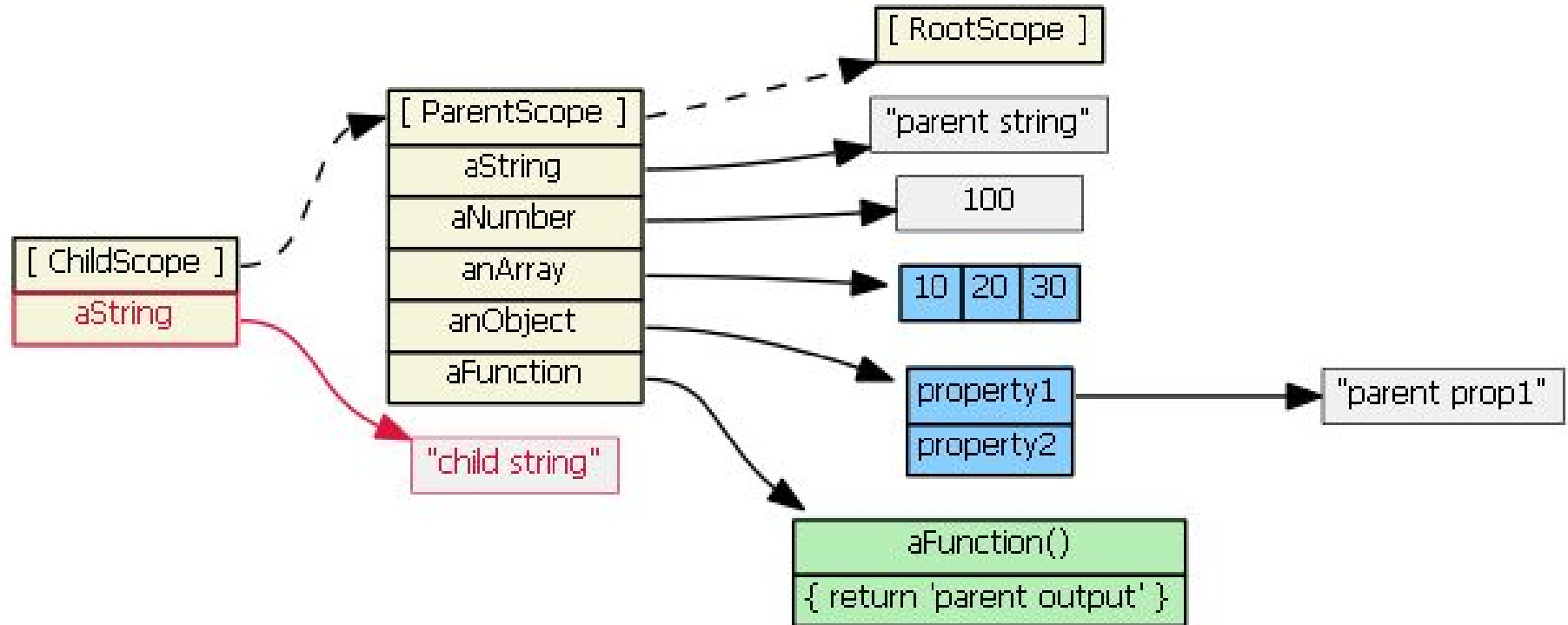
# Le scope

- JavaScript : L'héritage par prototypage

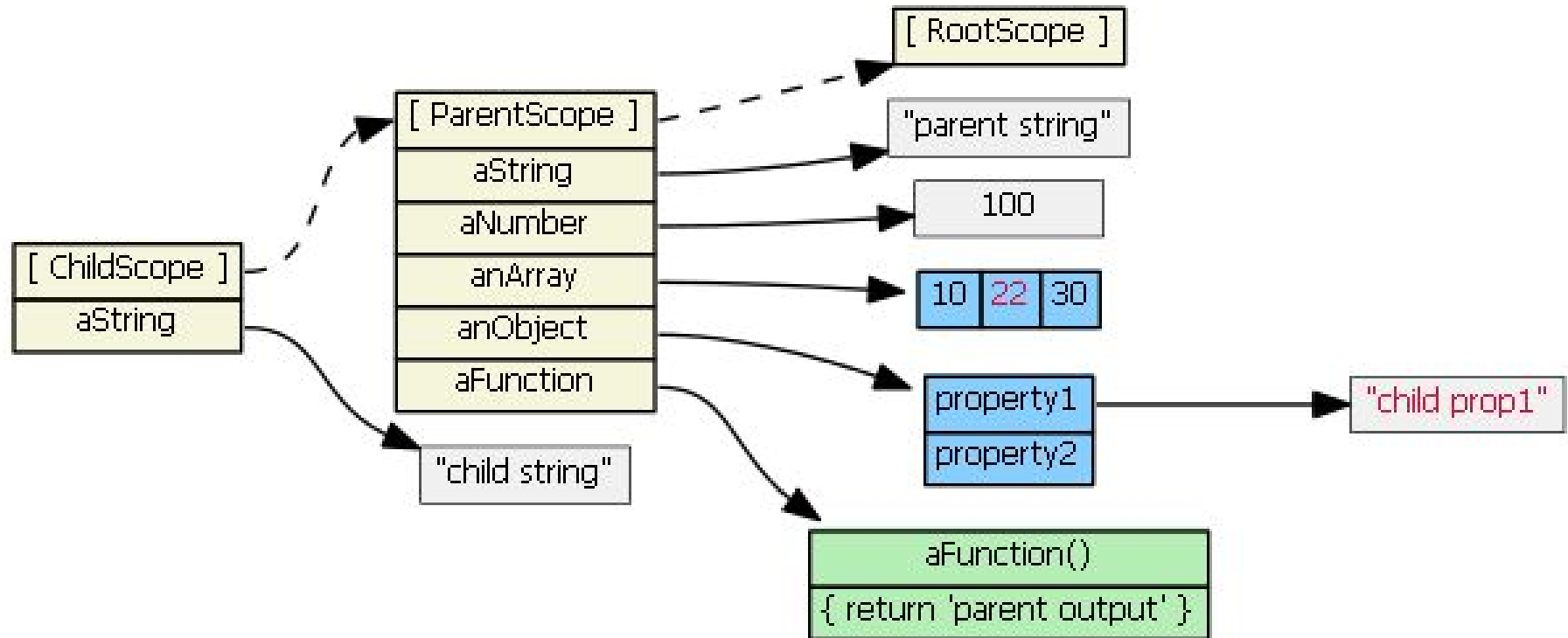
# Le scope - L'héritage par prototypage



# Le scope - L'héritage par prototypage

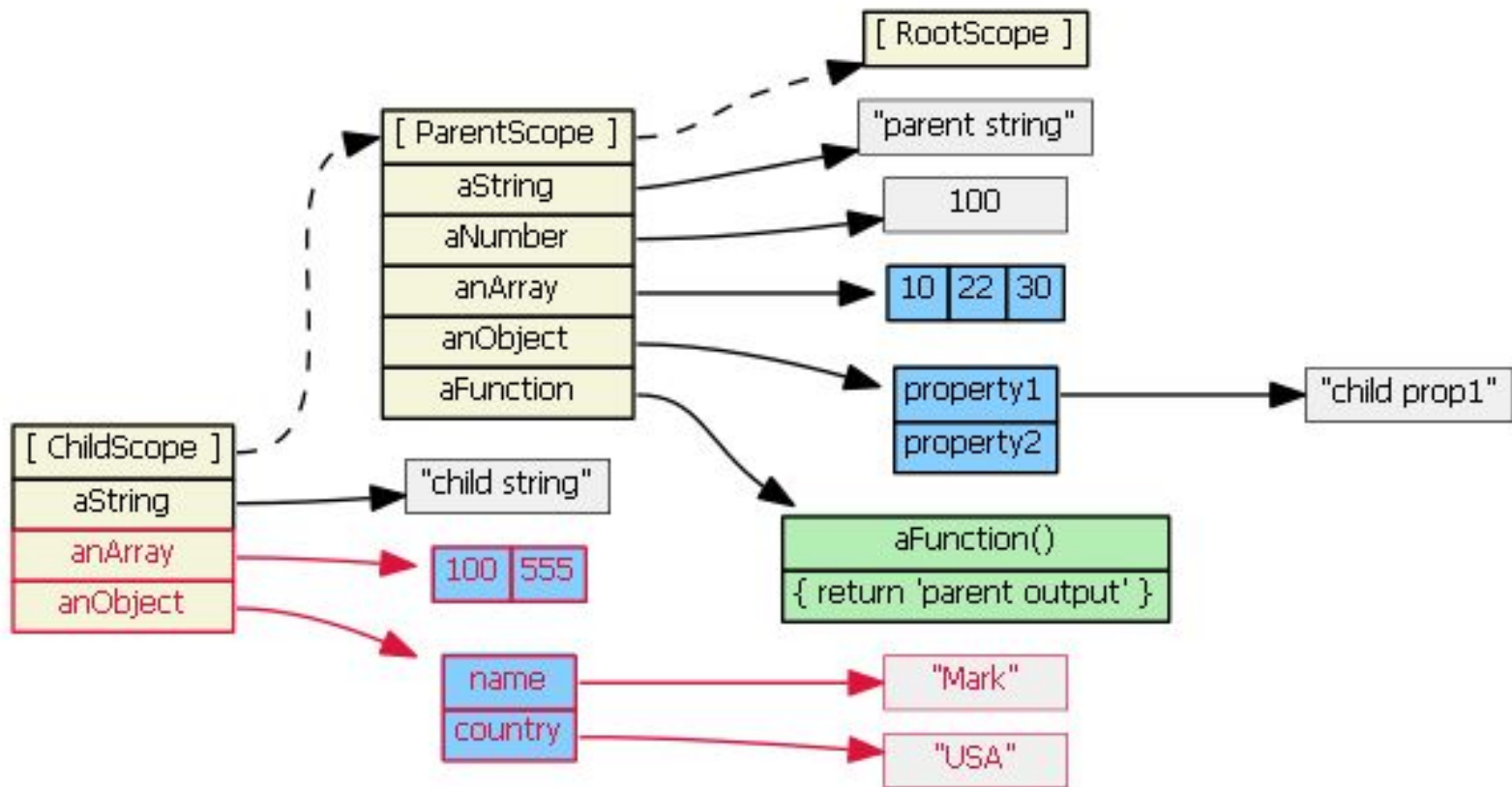


# Le scope - L'héritage par prototypage

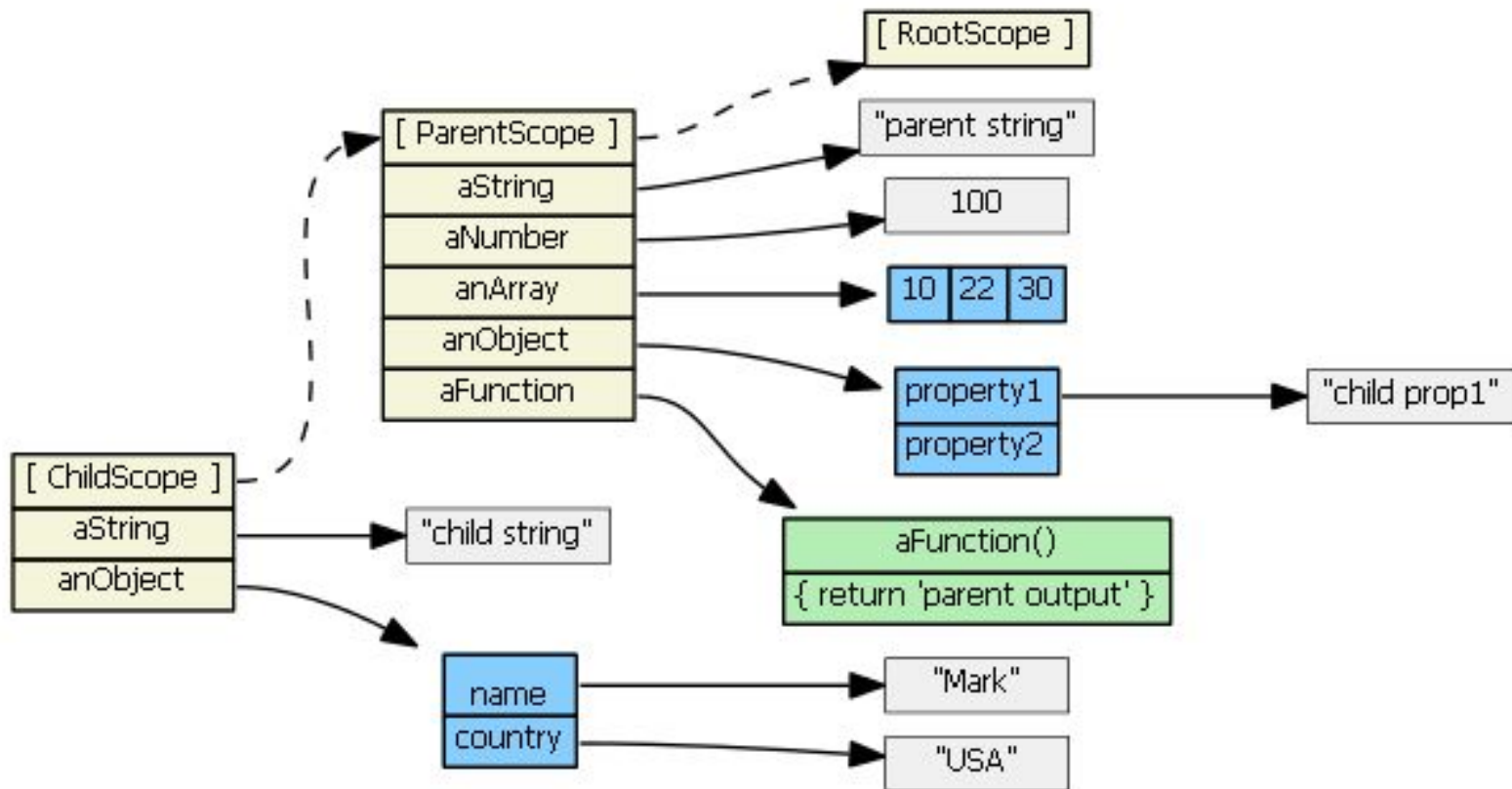




# Le scope - L'héritage par prototypage



# Le scope - L'héritage par prototypage



# Le scope

- JavaScript : L'héritage par prototypage
- L'héritage dans AngularJS

# Le scope - L'héritage dans AngularJS

- Plusieurs types de scope
  - Normal (héritage)
  - Isolé (aucun héritage)
- Toujours un rootScope (ng-app node)
  - \$rootScope

# Le scope

- JavaScript : L'héritage par prototypage
- L'héritage dans AngularJS
- ControllerAs syntax ( $\geq 1.1.5$  &  $< 1.5$ )

# Le scope - ControllerAs syntax

- AngularJS  $\geq 1.1.5$  &  $< 1.5$
- Permits les nested controllers
- Concrètement
  - `$scope.weather = this;`

# Le scope

- JavaScript : L'héritage par prototypage
- L'héritage dans AngularJS
- ControllerAs syntax ( $\geq 1.1.5$  &  $< 1.5$ )
- Attention aux types primitifs

# Le scope - Attention aux types primitifs

- Certaines directives
  - ng-repeat
  - ng-switch
  - ng-include
- Conseil : Toujours travailler sur un objet !

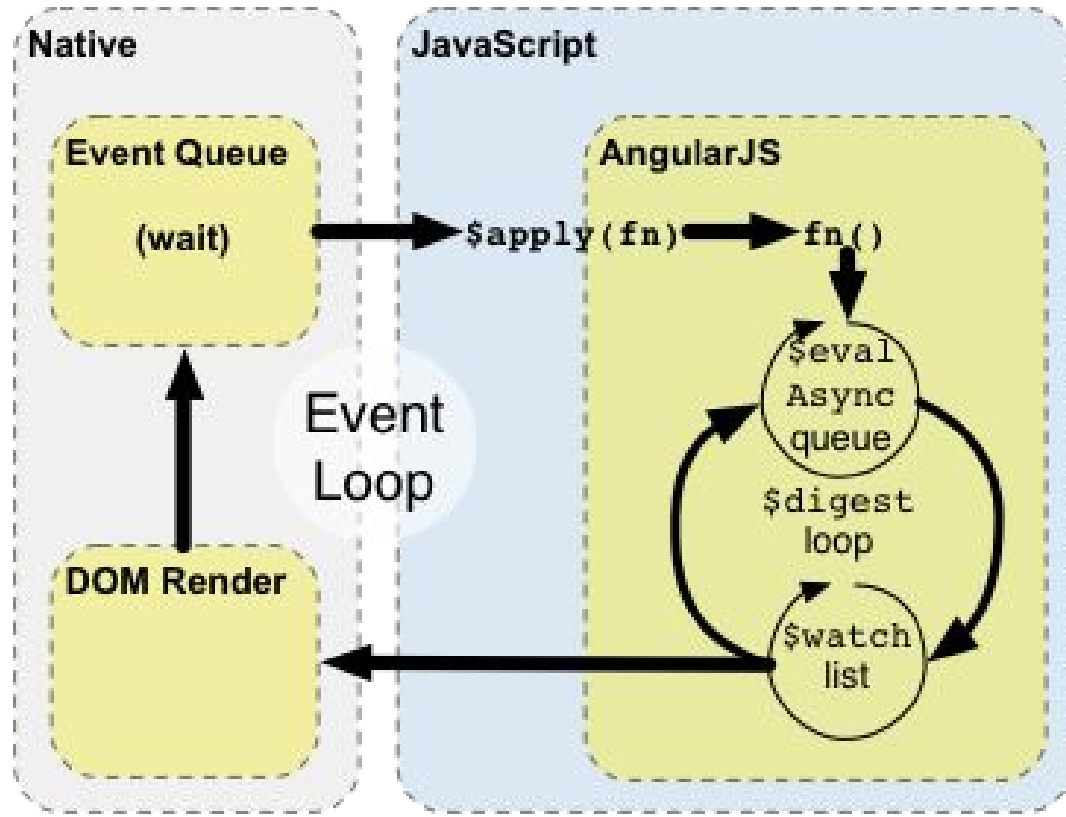


# Le scope

- JavaScript : L'héritage par prototypage
- L'héritage dans AngularJS
- ControllerAs syntax ( $\geq 1.1.5$  &  $< 1.5$ )
- Attention aux types primitifs
- DEMO
  - [http://localhost:8080/day\\_02/step\\_01](http://localhost:8080/day_02/step_01)

# 3 - Le cycle de vie

# Le cycle de vie AngularJS



# Le cycle de vie AngularJS

## \$apply et \$digest

- Two-way data binding
- On peut écouter les changements
  - `$scope.$watch`
- DEMO
  - [http://localhost:8080/day\\_02/step\\_02](http://localhost:8080/day_02/step_02)

# Le cycle de vie AngularJS

## Pourquoi & quand appeler \$apply manuellement

- Cycle de vie AngularJS = Monde AngularJS
- Toute opération extérieur
  - Plugin jQuery
  - Requête AJAX (sans les services AngularJS)
  - Code asynchrone
- DEMO
  - [http://localhost:8080/day\\_02/step\\_03](http://localhost:8080/day_02/step_03)

# Le cycle de vie AngularJS

## Performance

- Les humains sont
  - Lent
  - Limité
- Benchmark
  - <http://jsperf.com/angularjs-digest/6>
- L'avenir du dirty-checking
  - Object.observe dans ECMAScript 7

# 3 - Les templates

# Les templates

## Les directives

- Beaucoup de directives natives
  - <https://docs.angularjs.org/api/ng/directive>
- DEMO
  - [http://localhost:8080/day\\_02/step\\_04](http://localhost:8080/day_02/step_04)



# Les templates

## Les formulaires simples

- HTML5 Validation API
  - <http://www.html5rocks.com/en/tutorials/forms/constraintvalidation/>

# Les templates

## Les formulaires simples

HTML5 Attribute	ng Attribute	Registered Error
required="bool"	ng-required="..."	ngModel.\$error.required
minlength="number"	ng-minlength="number"	ngModel.\$error.minlength
maxlength="number"	ng-maxlength="number"	ngModel.\$error.maxlength
min="number"	ng-min="number"	ngModel.\$error.min
max="number"	ng-max="number"	ngModel.\$error.max
pattern="patternValue"	ng-pattern="patternValue"	ngModel.\$error.pattern

# Les templates

## Les formulaires simples

<code>&lt;input type="..."&gt;</code>	Registered Error
<code>type="email"</code>	<code>ngModel.\$error.email</code>
<code>type="url"</code>	<code>ngModel.\$error.url</code>
<code>type="number"</code>	<code>ngModel.\$error.number</code>
<code>type="date"</code>	<code>ngModel.\$error.date</code>
<code>type="time"</code>	<code>ngModel.\$error.time</code>
<code>type="datetime-local"</code>	<code>ngModel.\$error.datetimelocal</code>
<code>type="week"</code>	<code>ngModel.\$error.week</code>
<code>type="month"</code>	<code>ngModel.\$error.month</code>

# Les templates

## Les formulaires simples

- HTML5 Validation API
  - <http://www.html5rocks.com/en/tutorials/forms/constraintvalidation/>
- DEMO
  - [http://localhost:8080/day\\_02/step\\_05](http://localhost:8080/day_02/step_05)