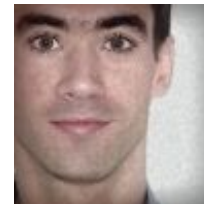




Intégrer un composant externe dans une directive



Thierry CHATEL



Antoine RICHARD



API Google Maps 1/4

[Tutoriel : Hello World](#)

```
var mapOptions = {  
  center: new google.maps.LatLng(-34.397, 150.644),  
  zoom: 8,  
  mapTypeId: google.maps.MapTypeId.ROADMAP  
};  
var map = new google.maps.Map(  
  document.getElementById("map-canvas"),  
  mapOptions  
);
```

API Google Maps 2/4

```
map.setZoom(zoom);  
map.setCenter(new google.maps.LatLng(lat, lng));  
  
map.getZoom()  
map.getCenter().lat()  
map.getCenter().lng()
```

API Google Maps 3/4

```
google.maps.event.addListener(map, 'zoom_changed',  
    function () {  
  
        // ... map.getZoom()  
    });  
  
google.maps.event.addListener(map, 'center_changed',  
    function () {  
  
        // ...map.getCenter()...  
    });
```

API Google Maps 4/4

```
new google.maps.Marker({  
  position: new google.maps.LatLng(lat, lng),  
  map: map,  
  title: label  
});
```

Création d'une directive

```
module.directive('nomEnCamelCase',  
    function (service1, service2) {  
  
    var directiveDefinition = {  
        restrict: 'A',    // 'A' : attribute  
        scope: false,  
        link: function (scope, element, attrs) {  
  
            // ... code ...  
  
        }  
    };  
  
    return directiveDefinition;  
  
});
```

Etape 1 : *afficher une carte*

a. Créer la directive `gmaps`

b. Ajouter la classe CSS `gmaps` pour la hauteur du `<div>`

```
element.addClass("gmaps");
```

c. Faire afficher la carte du *Hello World* de Google Maps

- en reprenant l'intérieur de la fonction `initialize()`
- en passant l'élément HTML `element[0]`

Scope isolé

```
scope: {  
  // Binding monodirectionnel depuis un attribut texte  
  propText: '@attrText',  
  
  // Binding bidirectionnel sur la valeur  
  //   d'une expression  
  propValue: '=attrValue',  
  
  // Fonction permettant de déclencher une action  
  //   (indiquée dans une expression)  
  propAction: '&attrAction',  
}
```

Etape 2 : *afficher la bonne carte*

a. Définir un scope isolé

- avec une propriété `zoom`
 - et une propriété `center`
-

b. Remplacer les valeurs en dur de la carte

\$watch

```
// Surveiller un type primitif ou la référence d'un objet  
scope.$watch('expression', function (newValue, oldValue) {  
    // ...  
});  
  
// Surveiller le contenu d'un objet  
scope.$watch('expression', function (newValue, oldValue) {  
    // ...  
}, true);
```

Etape 3 : *binding scope -> carte*

- a. Appeller `scope.$watch()` pour surveiller le zoom

```
map.setZoom(zoom);
```

- utiliser `parseInt()`
-

- b. Mettre un second watch pour les coordonnées du centre

- 3ème paramètre à `true` pour surveiller en profondeur

```
map.setCenter(  
    new google.maps.LatLng(lat, lng));
```

- utiliser `parseFloat()`

\$apply

```
// Exécution de code dans le contexte d'AngularJS
// suivie par un rafraîchissement de la vue
scope.$apply(function () {
    // ... code ...
});

// Exécution désynchronisée
setTimeout(function () {
    // ... code ...
});

// 'Safe apply'
function safeApply(scope, fn) {
    (scope.$$phase || scope.$root.$$phase)
        ? scope.$eval(fn)
        : scope.$apply(fn);
}
```

Etape 4 : *binding carte -> scope*

- a. Listener Google Maps sur l'événement 'zoom_changed'

```
google.maps.event.addListener(map,  
    'zoom_changed',  
    callbackFunction);  
  
map.getZoom()
```

-
- b. Listener Google Maps sur 'center_changed'

```
map.getCenter().lat()
```

Template

```
template: '... template AngularJS avec directives ...',
```

```
// ou :
```

```
templateUrl: 'filename'
```

```
// Insère le template dans l'élément courant
```

```
replace: false,
```

```
// Remplace l'élément courant par le template
```

```
// (avec un seul élément racine)
```

```
replace: true,
```

Etape 5 : directive en élément

- a. Modifier la directive pour qu'elle puisse servir d'élément HTML

```
<gmaps center="map.center"  
        zoom="map.zoom"></gmaps>
```

- il faut passer un <div> à Google Maps
- utiliser un template

Etape 6 : refactoring

a. Refactoring : template séparé

- externaliser le template dans un fichier `gmaps.html` séparé
- indiquer son URL (relative) avec `templateUrl`
- insérer un `<div>` à la racine du template, contenant celui de la carte avec la classe `gmaps`
- il faut alors passer à `GoogleMaps` `element.find('div')[0]`

Etape 7 : marqueur

a. Mettre sous la carte un formulaire `<form>` avec

- un champ `<input type="text" ng-model="label"/>`
 - un bouton `<input type="submit" value="Marqueur"/>`
-

b. Créer dans le scope une fonction qui ajoute un marqueur

- aux coordonnées courantes du centre de la carte
 - avec le titre saisi dans le formulaire (*label*), lequel est effacé
-

c. Appeler cette fonction à la soumission du formulaire

- `<form ng-submit="addMarker()">`

Etape 8 : snapshots

a. Dans la fonction d'ajout d'un marqueur

- ajouter, dans un tableau du scope, un objet snapshot `{lat, lng, zoom, label}`
 - `parseInt` pour `zoom`, `parseFloat` pour `lat` et `lng`
-

b. Afficher sous la carte et le formulaire une série de boutons

- bouton `<button>` répété d'après le tableau des snapshots
avec `ng-repeat="snapshot in snapshots"`
 - libellé saisi pour le marqueur : `{{snapshot.label}}`
-

c. Créer une fonction `goto(snapshot)` dans le scope

- qui positionne la carte sur les valeurs `zoom`, `lat` et `lng` enregistrées
- déclenchée au clic sur le bouton : `ng-click="goto(snapshot)"`

Etape 9 : validation et délai

a. Désactiver le bouton *Marqueur* s'il n'y a pas de libellé

- Mettre un attribut `name="form"` au formulaire
 - Rendre requis le champ de saisie du libellé, avec un attribut `required`
 - Mettre au bouton *Marqueur* : `ng-disabled="form.$invalid"`
-

b. Temporiser l'enregistrement du snapshot (*image animée 2s*)

- afficher le gif animé pendant 2s, avant que le nouveau bouton apparaisse
- utiliser le service `$timeout`
- utiliser `ng-show="condition"` pour conditionner la visibilité d'un élément HTML

Etape 10 : snapshots partagés

On veut que les snapshots soient communs à toutes les cartes.

a. Publier un service en appelant sur le module :

- `.value('serviceName', serviceValue)`
- avec comme valeur un tableau vide

.....

b. Stocker le tableau des snapshots dans le service

- penser à injecter le service
- publier le service dans le scope