

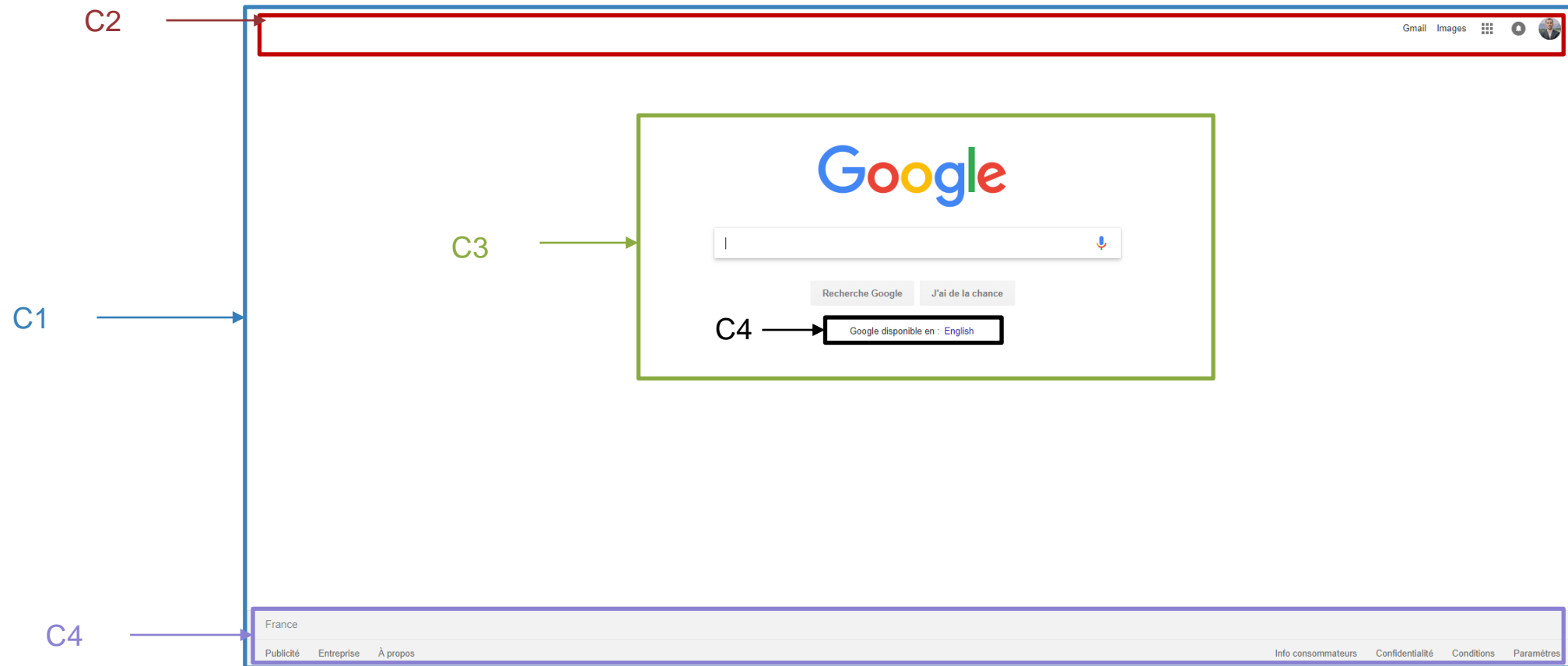
# Components

Animé par Mazen Gharbi

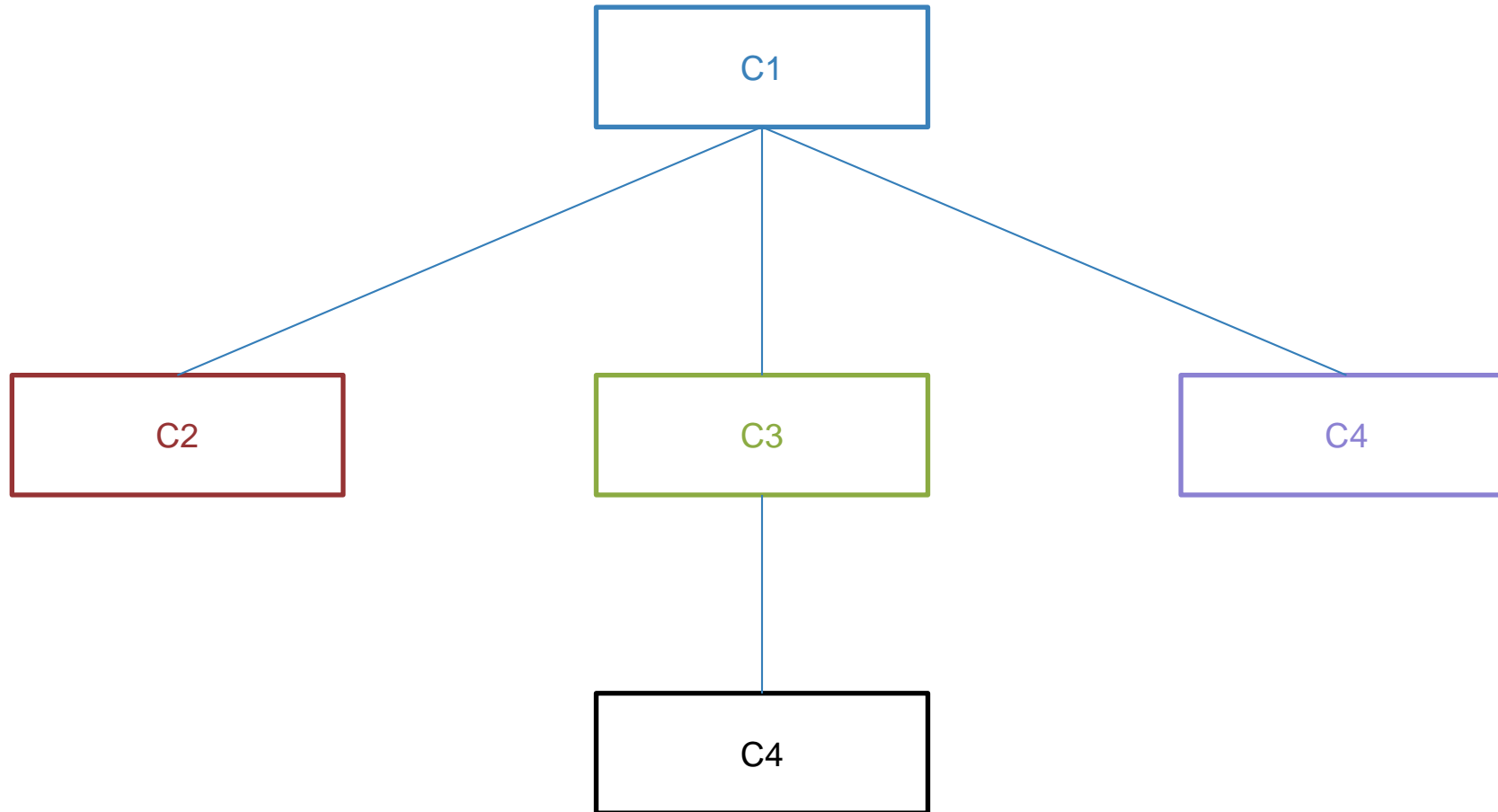
# Les piliers du framework

- ▷ **Un composant contrôle une vue ou une partie d'une vue**
- ▷ L'un de principaux concepts d'Angular est de voir une application comme une arborescence de composants.
- ▷ Les composants permettent une meilleure décomposition de l'application, facilitent le refactoring et le testing.
- ▷ Chaque composant est isolé des autres composants. Il n'hérite pas implicitement des attributs des composants parents.

# Séparation par composants



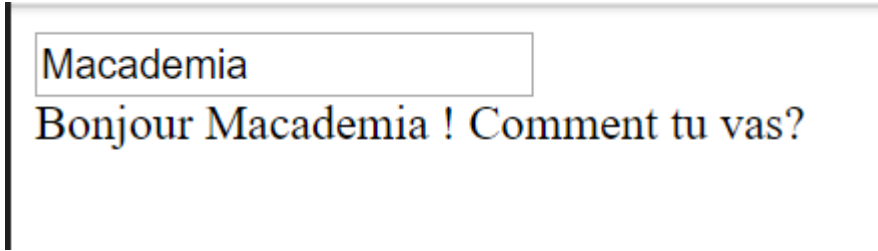
# Séparation par composants



# Notre première application

- ▷ Rien de mieux qu'un exemple

<https://stackblitz.com/edit/components-first-application>



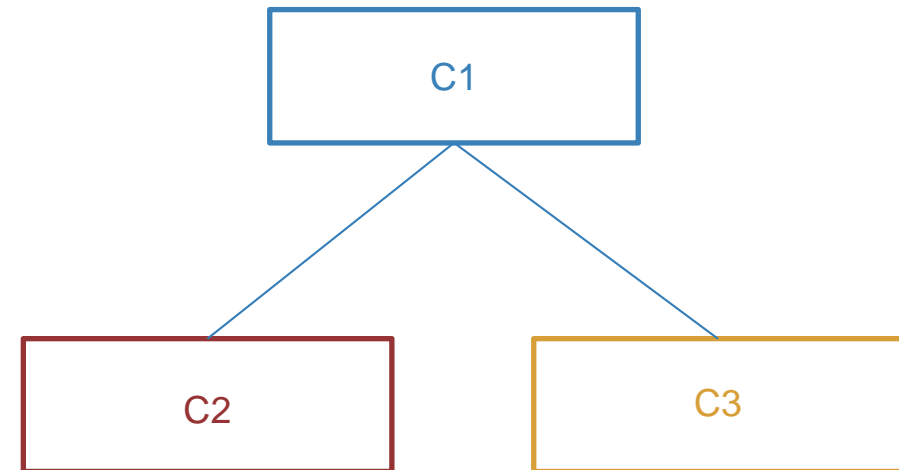
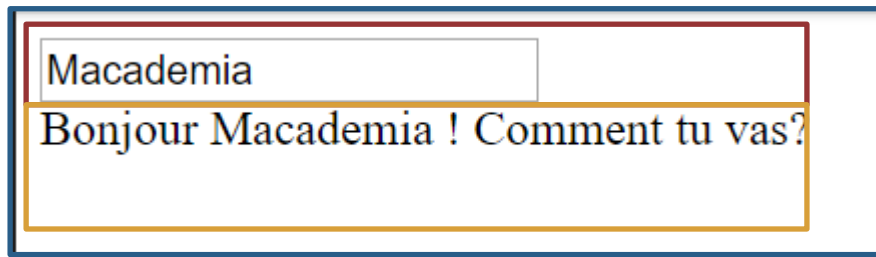
Macademia  
Bonjour Macademia ! Comment tu vas?

# Quelques bonne pratiques

- ▷ Suffixez vos composants avec 'Component' ;
- ▷ Dashcase pour les noms de fichiers ;
- ▷ UpperCamelCase pour les noms de classes ;
- ▷ Préfixez le "selector" avec un identifiant propre à votre produit pour éviter les collisions ;
- ▷ Préférez l'utilisation de templateUrl et styleUrls si le contenu est trop volumineux ;

# Créons une structure arborescente

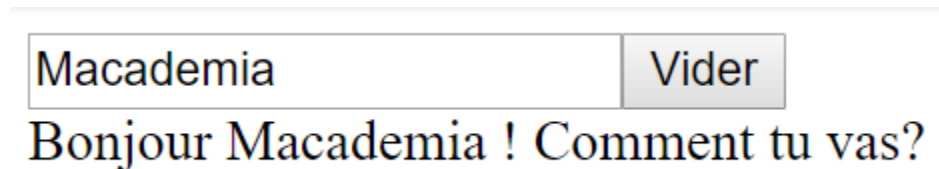
On va tenter de découper notre application précédente en plusieurs composants



<https://stackblitz.com/edit/components-first-application-2>

# Une demande de dernière minute

- Le client adore! Mais il aimerait ajouter un bouton reset pour vider le contenu de l'input ;



Macademia

Vider

Bonjour Macademia ! Comment tu vas?

 La directive « (click) » permet de réagir à l'évènement click sur l'élément auquel elle est appliquée

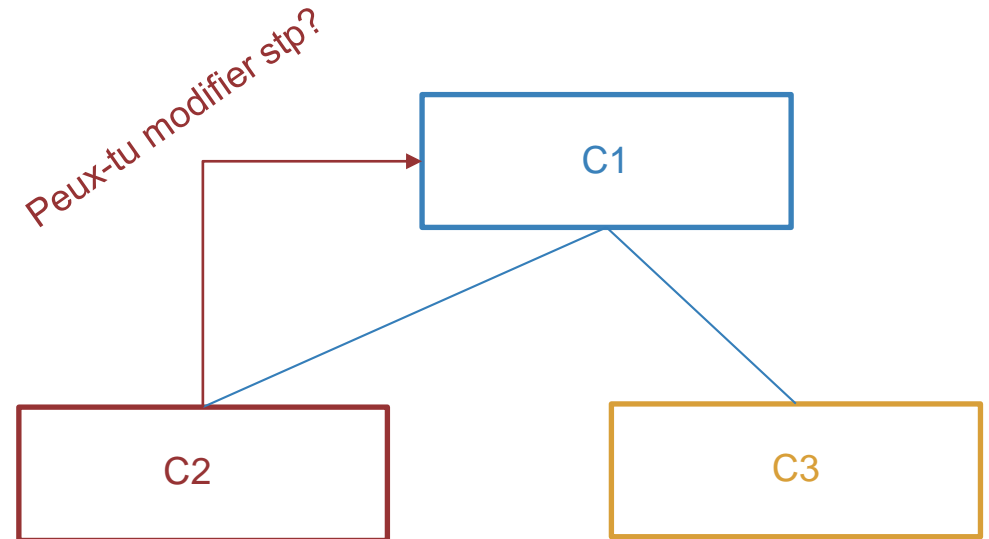
<https://stackblitz.com/edit/components-first-application-2-problem>



# Pas de two-way binding en Angular !

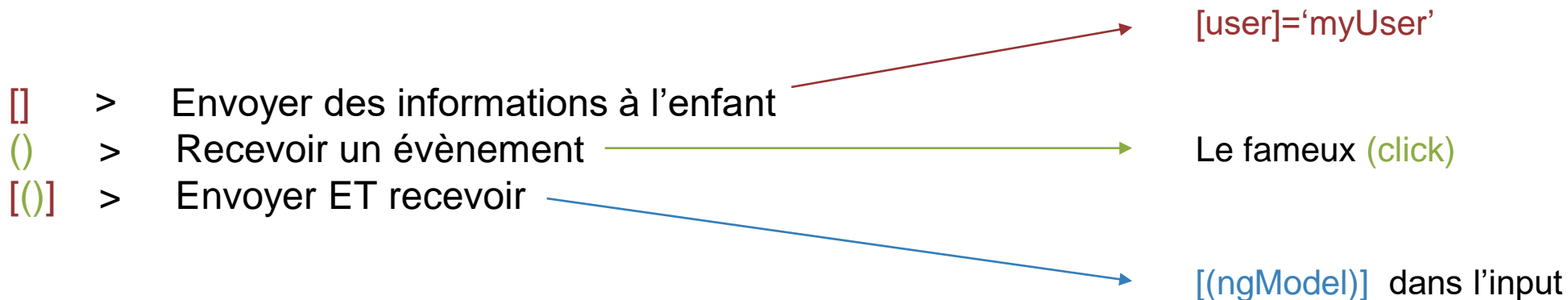
- L'édition de "user.name" fonctionnait car les trois composants partageaient une référence vers le même objet 'user' ;
- Les variables sont mises à jour du haut vers le bas, mais l'inverse est « impossible » ;

Ce serait bien d'avoir quelque chose du genre :



# EventEmitter

- ▷ Les EventEmitters vont nous permettre d'envoyer un évènement de l'enfant vers le père ;
- ▷ Le père décide de ce qu'il veut faire une fois l'évènement reçu
- ▷ Mais avant, petit retour sur la syntaxe Angular :



# Mise en place

```
export class AskNameComponent implements OnInit {  
  @Input() user;  
  @Output() cleanText = new EventEmitter();  
}
```





*Pour que cela fonctionne, il est nécessaire de créer l'eventEmitter directement lors de l'initialisation, sinon angular lève une erreur*

<https://stackblitz.com/edit/components-first-application-eventemitter>

# Simuler le two-way binding

- ▷ S'il n'y a pas de two-way binding, comment [(ngModel)] fonctionne-t-il ?
- ▷ [(ngModel)] n'est qu'en fait qu'un raccourci de nommage! Voyons comment le mettre en place pour notre application

-  *La fonction « emit » de nos EventEmitter peut prendre un paramètre correspondant à la valeur que l'on souhaite envoyer au père. Ce paramètre est optionnel*
-  *Le père peut récupérer la valeur envoyer par un enfant avec le mot-clé « \$event »*

<https://stackblitz.com/edit/components-first-application-eventemitter-2waybinding>

# Directives structurelles

- ▷ Les directives structurelles modifient le DOM en ajoutant/supprimant des éléments du DOM ;
- ▷ Le préfixe '\*' indique qu'il s'agit d'une directive structurelle.
- ▷ Voici les 2 plus importantes :

**\*ngIf**

<https://stackblitz.com/edit/macademia-components-ng-if>

**\*ngFor**

<https://stackblitz.com/edit/macademia-components-ng-if-tzixzp>

# Smart et Dumb components

- ▷ Vous allez créer un nombre conséquents de composants ;
- ▷ Il va être nécessaire de catégoriser nos composants, il va exister 2 types : les Smart et les Dumb
- ▷ Les composants « Smart » sont des composants "High-Level" qui contrôlent la logique « business » ;
- ▷ Les composants « Dumb » ne contiennent pas de logique « business ». Ils se chargent principalement du design et doivent échanger les données avec les composants parents via les « Inputs/Outputs » du composant.

# Questions