LIF'Tech – Marseille

Angular

TypeScript

TypeScript

Bertrand Estellon / Didier Villevalois

Aix*Marseille université
Initiative d'excellence

Pourquoi Angular ?

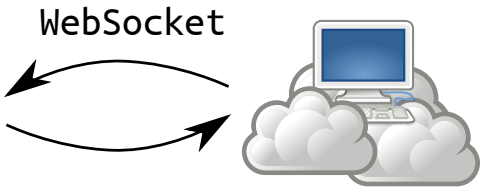
HTML

```
<ul id="list">
    <li>Bonjour</li>
    <li>Salut</li>
</ul>

<form id="form">
    <input type="text" />
    <button>Ok</button>
</form>
```

JS

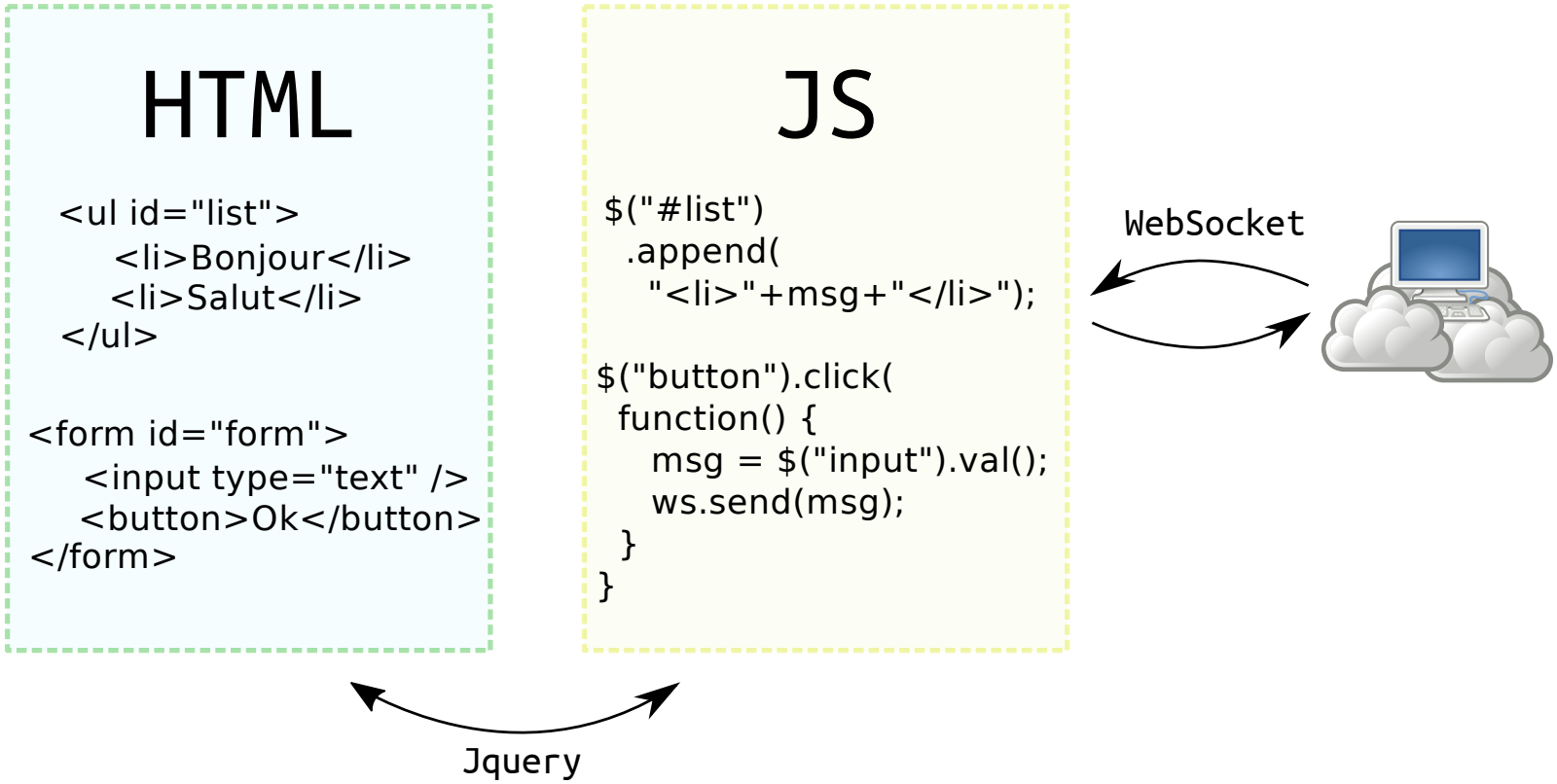
```
$("#list")
  .append(
    "<li>"+msg+"</li>");

$("button").click(
  function() {
    msg = $("input").val();
    ws.send(msg);
  }
}
```

WebSocket



Jquery

Pourquoi Angular ?



Rôle du JS pas clairement défini

Pas de séparation HTML/JS

Pourquoi Angular ?

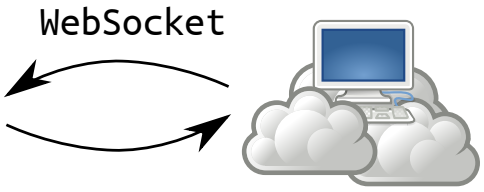
HTML

```
<ul id="list">
    <li>Bonjour</li>
    <li>Salut</li>
</ul>

<form id="form">
    <input type="text" />
    <button>Ok</button>
</form>
```

JS

```
$("#list")
  .append(
    "<li>"+msg+"</li>");

$("button").click(
  function() {
    msg = $("input").val();
    ws.send(msg);
  }
}
```

WebSocket



Jquery

Rôle du JS pas clairement défini

Pas de séparation HTML/JS

⇒

Difficile à maintenir

Difficile à tester

Non réutilisable

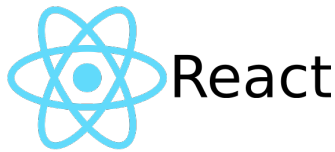
Angular

Angular est un framework "Single Page Application" (SPA)

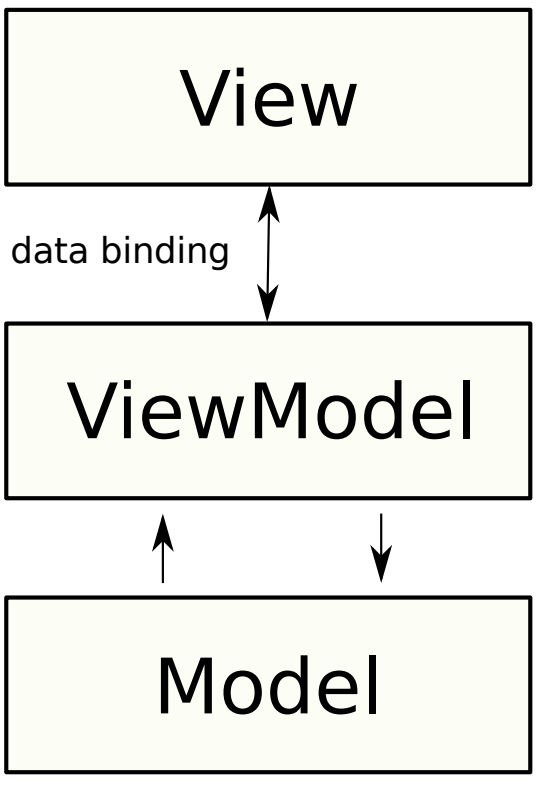
Historique :

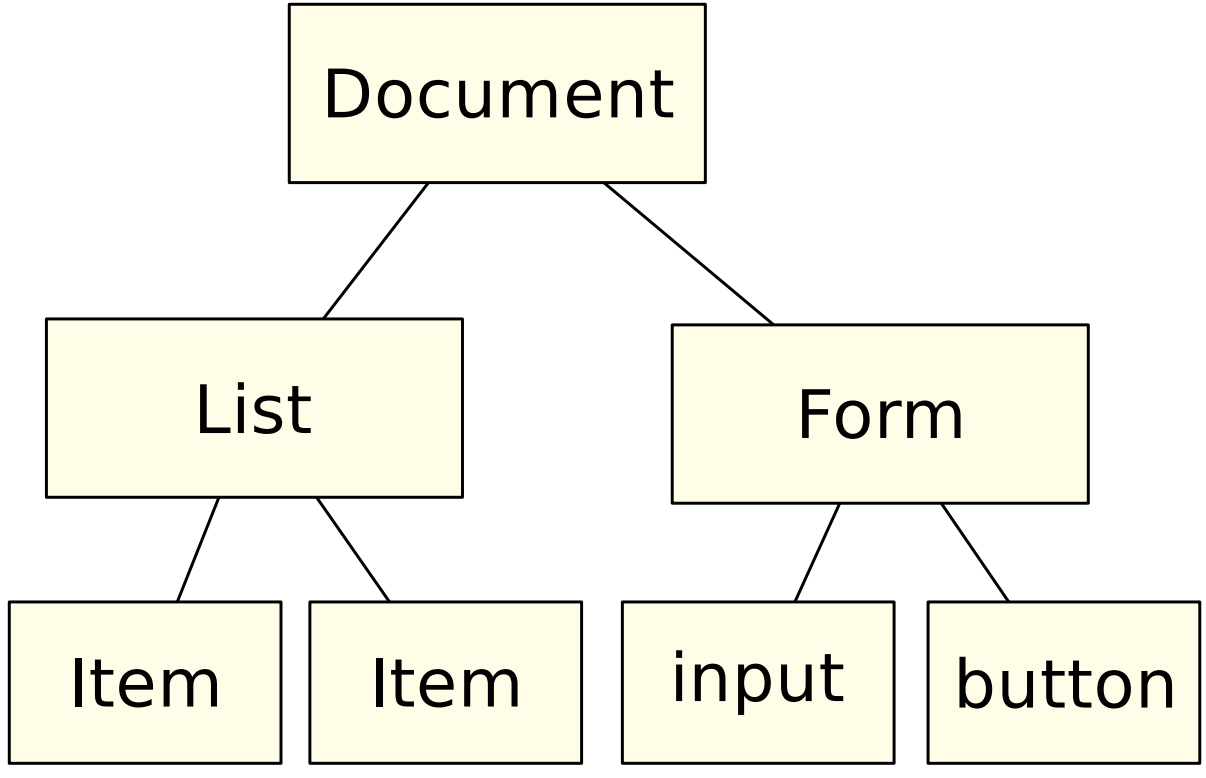
- AngularJS : Octobre 2010
- Angular 2.0
  - annonce : Septembre 2014
  - beta : Decembre 2015
  - final : Septembre 2016
- Angular 4.0
  - annonce : Decembre 2016
  - final : Mars 2017

Autres frameworks :

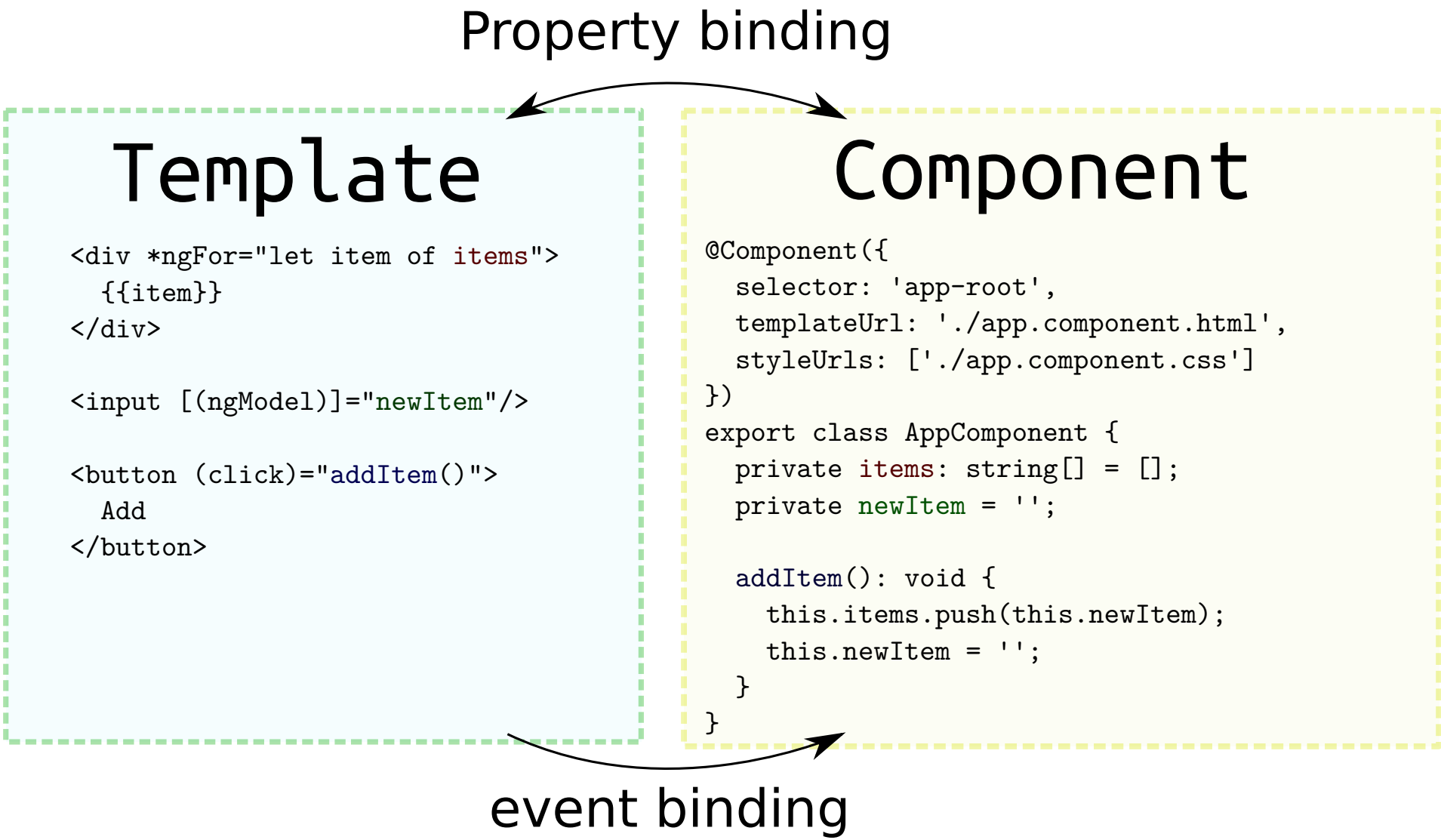


Angular = Model–View–ViewModel (MVVM) + Hierarchy





Data binding



Dependency injection

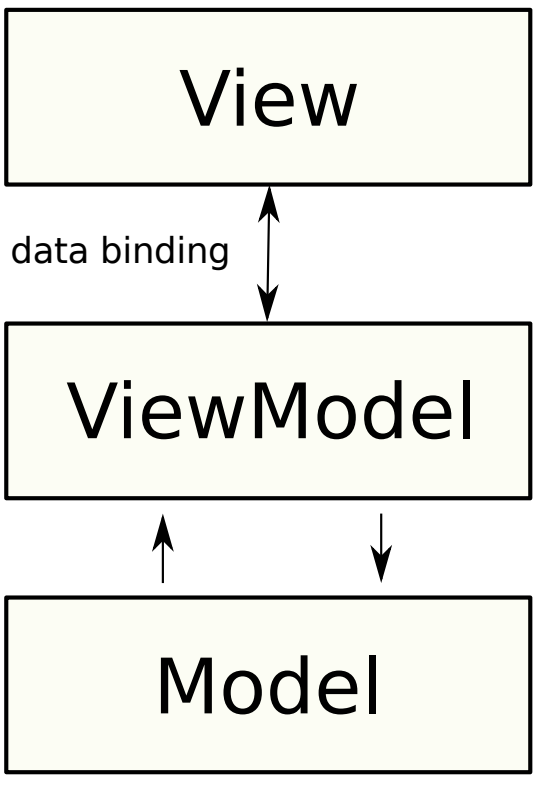
Service

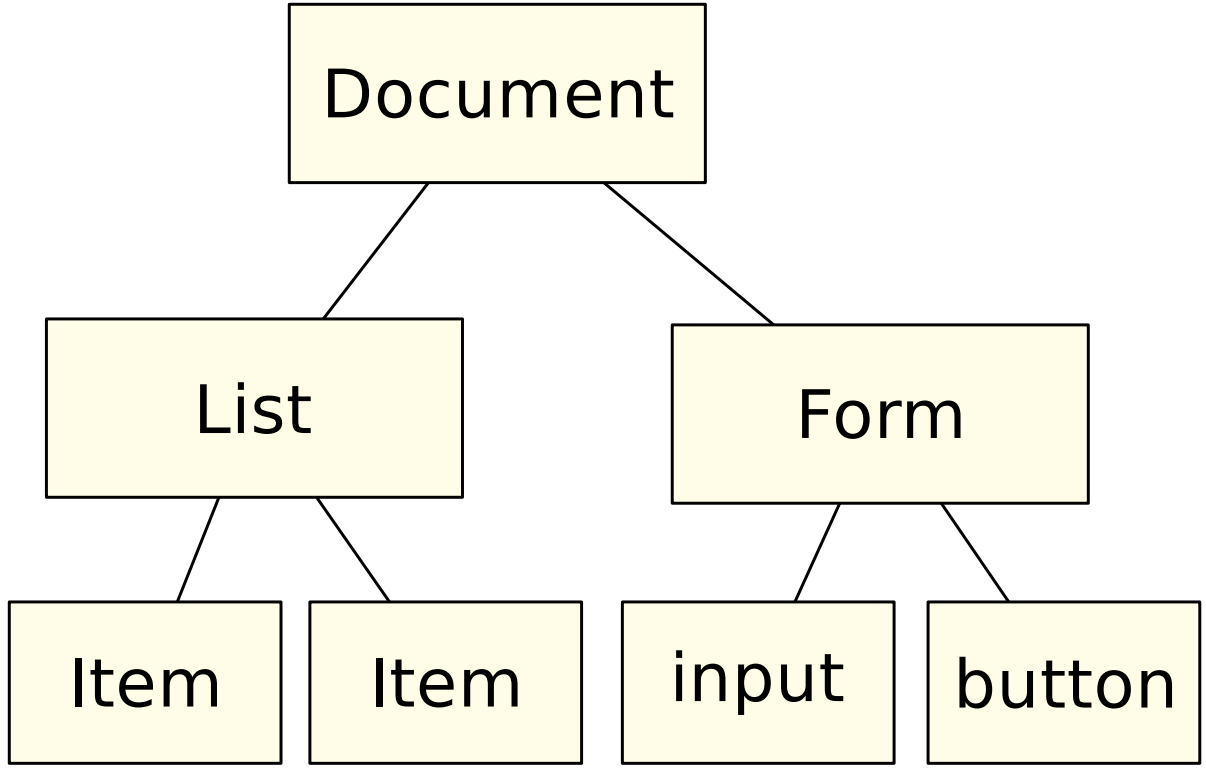
```
@Injectable()
export class ItemListService {
  private items: string[] = [];

  addItem(item: string): void {
    this.items.push(item);
  }

  getItems(): string[] {
    return this.items;
  }

}
```

Component

```
@Component({ .... }})
export class AppComponent {
  private newItem = '';

  constructor(private service: ItemListService) {}

  items(): string[] { return this.service.getItems(); }

  addItem(): void {
    this.service.addItem(this.newItem);
    this.newItem = '';
  }
}
```

Angular = Model–View–ViewModel (MVVM) + Hierarchy



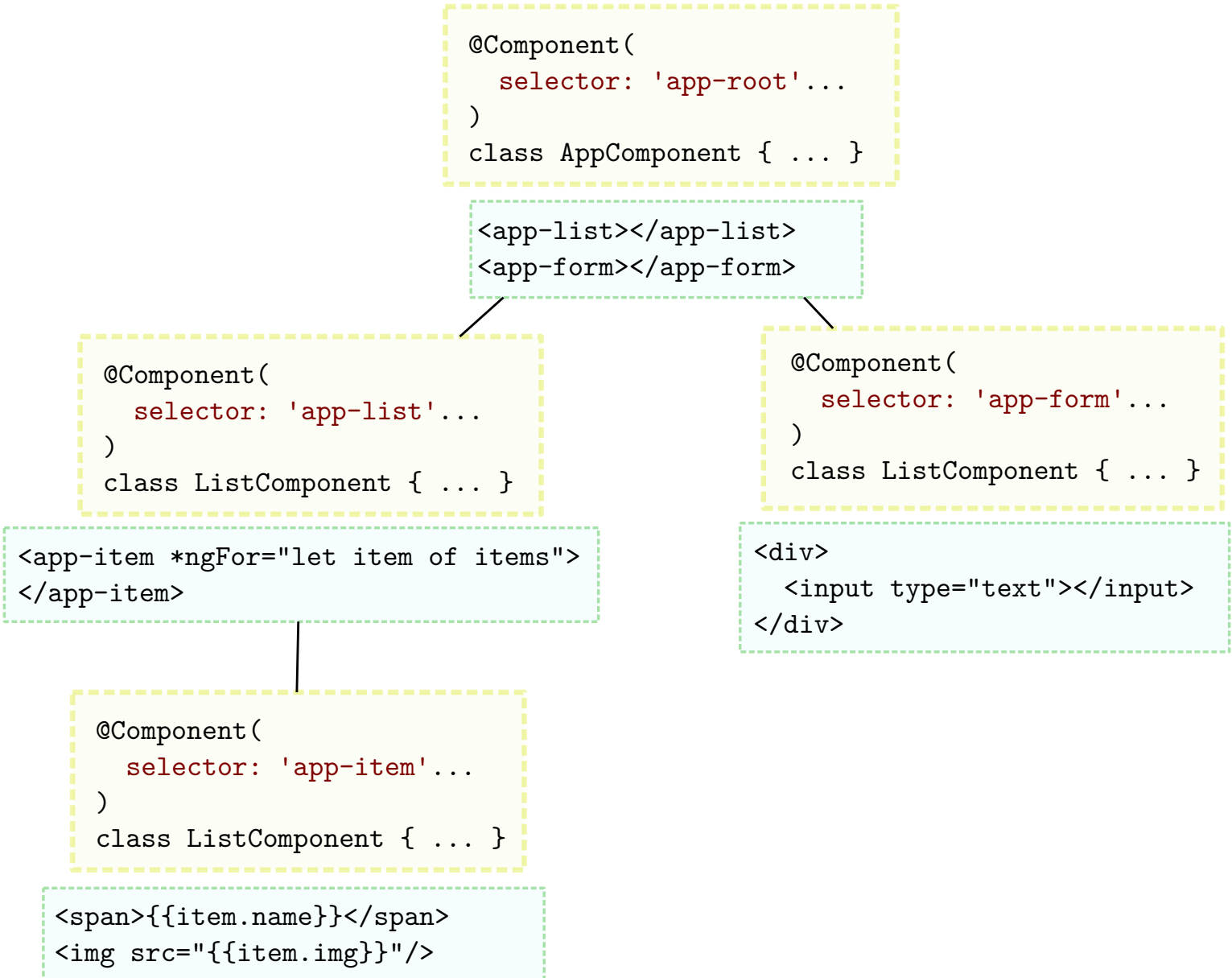


Directive = DOM++

```
@Directive({ selector: '[appHighlight]' })
export class HighlightDirective {
  @Input()
  private color: string;

  constructor(private el: ElementRef) {}

  @HostListener('mouseenter') onMouseEnter() {
    this.highlight(this.color);
  }

  @HostListener('mouseleave') onMouseLeave() {
    this.highlight(null);
  }

  private highlight(color: string) {
    this.el.nativeElement.style.backgroundColor = color;
  }
}
```

```
<span appHighlight [color]="'red'">
  Toto
</span>
```

Component = Directive + Template



Template – Interpolation

```
<div>My name is {{item.name}}</div>
<img src="{{item.img}}"/>
<p>The sum of 1 + 1 is {{1 + 1}}</p>
<p>The computed value is {{getComputedValue()}}</p>
```

```
@Component({
  selector: 'app-root',
  template: 'app.component.html'
})
export class AppComponent {
  private item: Item;
  /* ... */
  private getComputedValue(): number {
    /*...*/ return result;
  }
}
```

Template – Binding

Property

```
<img [src]="heroImageUrl">
<app-item [item]="currentItem"></app-item>
```

Event

```
<button (click)="onSave()">Save</button>
<app-item (delete)="delete($event)"></app-item>
```

Two-way

```
<input [(ngModel)]="name">
```

Class

```
<div [class.special]="isSpecial">Special</div>
```

Style

```
<button [style.color]="isSpecial
                        ? 'red'
                        : 'green'">
```

Template – Binding

```
<li [class.completed]="todo.completed">
    <div class="view">
        <input class="toggle" type="checkbox" [(ngModel)]="todo.completed">
        <label>{{ todo.content }}</label>
        <button class="destroy" (click)="onRemove()"></button>
    </div>
</li>
```

```
@Component({ selector: 'app-todo-item',
             templateUrl: './todo-item.component.html'})
export class TodoItemComponent {
  @Input()
  todo: Todo;

  @Output()
  remove = new EventEmitter<Todo>();

  public onRemove() { this.remove.emit(this.todo); }
}
```

```
<app-todo-item [todo]="currentTodo" (remove)="removeTodo($event)">
</app-todo-item>
```

Template – Structural directives

```
<div *ngIf="isActive">Active</div>
```

```
<div *ngFor="let item of items">{{item.name}}</div>
<div *ngFor="let item of items, let i = index">
   {{i + 1}} : {{item.name}}
</div>
<app-item *ngFor="let item of items"
          [item]="item"></app-item>
```

```
<div [ngSwitch]="emotion">
  <img *ngSwitchCase="'happy'"    src="happy.jpg"/>
  <img *ngSwitchCase="'sad'"      src="sad.jpg"/>
  <img *ngSwitchCase="'confused'" src="confused.jpg"/>
  <img *ngSwitchDefault           src="unknown.jpg"/>
</div>
```

Modules

```
@NgModule({
  declarations: [
    AppComponent,
    HighlightDirective
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  providers: [ItemListService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Angular CLI

Installation :

```
> npm install -g @angular/cli
```

Création du projet :

```
> npm new myApp
```

Exécution de l'application :

```
> cd myApp
> npm serve --open
```

Création des éléments de l'application :

```
> npm g component item-list
> npm g directive highlight
> npm g service server-connection
```

Pipes

```
{{ content | trim: 30}}
```

```
@Pipe({ name: 'trim' })
export class TrimPipe implements PipeTransform {
  transform(content: string, maxLength: number): string {
    const trimmedContent = content.trim();
    if (trimmedContent.length <= maxLength) {
       return trimmedContent;
    }
    return trimmedContent.substring(0, maxLength)
                         .concat('...');
  }
}
```

Router

```
const appRoutes: Routes = [
  { path: 'items', component: ItemListComponent },
  { path: 'items/:id',       component: ItemComponent },
  { path: '', redirectTo: '/items', pathMatch: 'full' },
  { path: '**', component: PageNotFoundComponent }
];

@NgModule({
  imports: [
    RouterModule.forRoot(appRoutes)
    ...
  ],
  ...
})
```

Router

Dans le template principal :

```
<router-outlet></router-outlet>
```

Changement de route dans les autres templates :

```
<a routerLink="/items" routerLinkActive="selected">
  All items
</a>
```

```
<a [routerLink]="['/items', id]">
  Go to detail
</a>
```