

LOG2420

Analyse et conception des interfaces utilisateur

Automne 2020

Semaine 12

Composants Web & Angular

Jinghui Cheng, Ph.D. (Prof. Responsable)

Walter de Abreu Cybis, Dr. (Chargé de cours)

École Polytechnique de Montréal

Plan du cours : semaine 12

Composants Web ←

- Origines, définition, portabilité

- Cycle de vie, passage de données

- Exemples

Angular

- Définition, assemblage

- Cycle de vie, démarrage

- Exemples

Le TP4

Composants Web

Origens et définition

Historique de la réutilisation Web

- *Copy-paste*
- Génération du HTML du côté serveur avec php, python
- Composants Web (*W3C/WHATWG 2011-14*): APIs permettant de créer des nouvelles balises ex(<cc_moto>) mettant en place des applications encapsulées et réutilisables (ex. *cc_moto.js*)
- Frameworks de composants Web: Angular, React, Vue...

Composants Web

Portabilité

Disponible sur tous les navigateurs (à l'aide de polyfills¹) sans besoin de librairies.

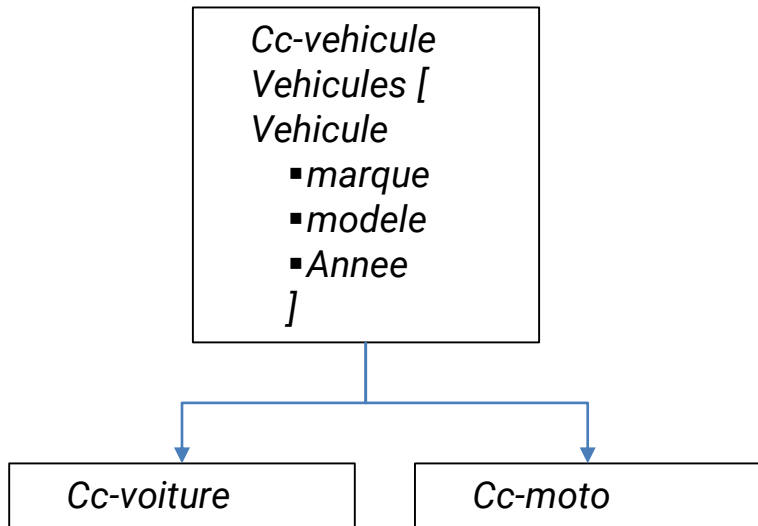
Standards :

- *window.customElements.define()*: pour créer et enregistrer de nouveaux éléments HTML `<custom_elements>` et les faire reconnaître par le navigateur;
- *element.attachShadow()*: pour encapsuler des définitions de style appliquées à de balises standards

¹un polyfill est une bibliothèque JavaScript destinée à émuler des fonctionnalités qui ne sont pas encore implémentées dans les navigateurs. Lorsqu'une fonctionnalité est absente, elle est émulée en JavaScript par le polyfill.

Composants Web

Example



```
<!DOCTYPE html>
<html lang="fr">
```

Index.html

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Composants Web</title>
  <script src="cc-liste-vehicules.js"></script>
  <script src="cc-liste-motos.js"></script>
  <script src="cc-liste-voitures.js"></script>
  <link rel="stylesheet" href="style.css">
```

```
</head>
```

```
<body>
```

```
  <div class="grid-container">
```

```
    <div class="grid-item">
```

```
      <h1>Voitures électriques</h1>
```

```
      <h2>En vedette</h2>
```

```
      <cc-liste-voitures affiche_marque="true" affiche_modele="true"
affiche_annee="true"></cc-liste-voitures>
```

```
    </div>
```

```
    <div class="grid-item">
```

```
      <h1>Motos électriques</h1>
```

```
      <h2>En vedette</h2>
```

```
      <cc-liste-motos affiche_marque="true" affiche_modele="true"
affiche_annee="true"></cc-liste-motos>
```

```
    </div>
```

```
</div>
```

```
</body>
```

Composants Web

Exemple

Élément customisé: cc-vehicules.js

```
Class CcListeVehicules extends HTMLElement {
  constructor() {
    super(); // hériter les attributs et méthodes de HTMLElement

    // obtient le shadow root pour recevoir le code encapsule
    this._root = this.attachShadow({ mode: 'open' });

    // donnees
    this.affiche_marque = new Boolean("false");
    this.affiche_modele = new Boolean("false");
    this.affiche_annee = new Boolean("false");
    this.vehicules = [];
  }
}
```

Constructor()

- Création du shadowRoot
- Déclaration des attributs

connectedCallback()

- définition du shadowRoot
- avec <template></template>
- et appendChild des données

```
connectedCallback() { // lorsque connecté
  // définit le code encapsule
  this._root.innerHTML = `
    <style>
      .frame {
        background-color: #33b5e5;
        color: #ffffff;
        margin: 5px;
        padding: 5px;
      }
      h1, h2 {
        color: blue;
      }
    </style>
    <template id="template-vehicule">
      <div class="frame">
        <h2 id="marque"></h2>
        <p id="modele"></p>
        <p id="annee"></p>
      </div>
    </template>
    <div id="result"></div>
  `;

  // crée les variables avec le fragment du code encapsule
  this.templateContent = this._root.querySelector("#template-vehicule").content;
  this.result = this._root.querySelector("#result");

  this.vehicules.map(vehicules => {
    // clone le templateContent
    const clone = document.importNode(this.templateContent, true);
    // met à jour le clone avec les données de chaque véhicule si demandé
    if (this.affiche_marque === "true") {
      clone.querySelector("#marque").innerHTML = vehicules.marque;
    }
    if (this.affiche_modele === "true") {
      clone.querySelector("#modele").innerHTML = vehicules.modele;
    }
    if (this.affiche_annee === "true") {
      clone.querySelector("#annee").innerHTML = vehicules.annee;
    }
    // ajoute le clone au shadow DOM
    this.result.appendChild(clone);
  });
}
```

```
static get observedAttributes() {
  return ["vehicules", "marque", "modele", "annee"];
}

attributeChangedCallback(name, oldValue, newValue) {
  if (name === 'vehicules') {
    this.vehicules = newValue;
  }
  if (name === 'affiche_marque') {
    this.affiche_marque = newValue;
  }
  if (name === 'affiche_modele') {
    this.affiche_modele = newValue;
  }
  if (name === 'annee') {
    this.affiche_annee = newValue;
  }
}

// fin de la classe

// registre de la classe en dehors de la classe
window.customElements.define('cc-liste-vehicules', CcListeVehicules);
```

get observedAttributes()

attributeChangedCallback(...)

- et mise à jour des attributs

Registre de la classe

Héritage entre des éléments customisés

cc-liste-voitures.js

```
//registre de la classe en dehors de la classe
window.customElements.define('cc-liste-motos', CcListeMotos);
```

```
//registre de la classe en dehors de la classe
window.customElements.define('cc-liste-voitures', CcListeVoitures);
```

Composants Web

Example

Voitures électriques

En vedette

Hyundai IONIQ 2020
Nissan LEAF 2019
Tesla Model 3 2020
Toyota Mirai 2020
Chevrolet Bolt EV 2020
Kia Niro EV 2020

Motos électriques

En vedette

CSC City Slicker 2019
Zero FX 2020
Cake Kalk& 2020
Evoke Urban 2020
Emflux One 2020
Harey Davidson LiveWire 2019

Composants Web

Cycle de vie

- `constructor();`
 - `connectedCallback();`
 - `disconnectedCallback();`
 - `attributeChangedCallback(name, oldValue, newValue);`
 - `adoptedCallback()` ;- lorsque le composant est réutilisé
- `// Inscrire le composant`
- `window.customElements.define('my-element', MyElement);`

Passage de données

- Getters & setters propriétés JS //objets
ex. `get vehiculesList(){ }` & `set vehiculesList(list) { }`
- Attributs Html:
`get observedAttributes(); //strings`
`attributeChangedCallback();`

Composants Web

Librairies & frameworks

Librairies (éléments unitaires - from webcomponents.org/)

- Lightning/Aura
- Paper-icon-button
- Pf-calendar
- Iron-form
- Emoji-rain
- app-media

Frameworks (gestion d'états, base de données,...)

- React
- View
- Angular

Rendent systématique et facilitent la mise en place des composants Web

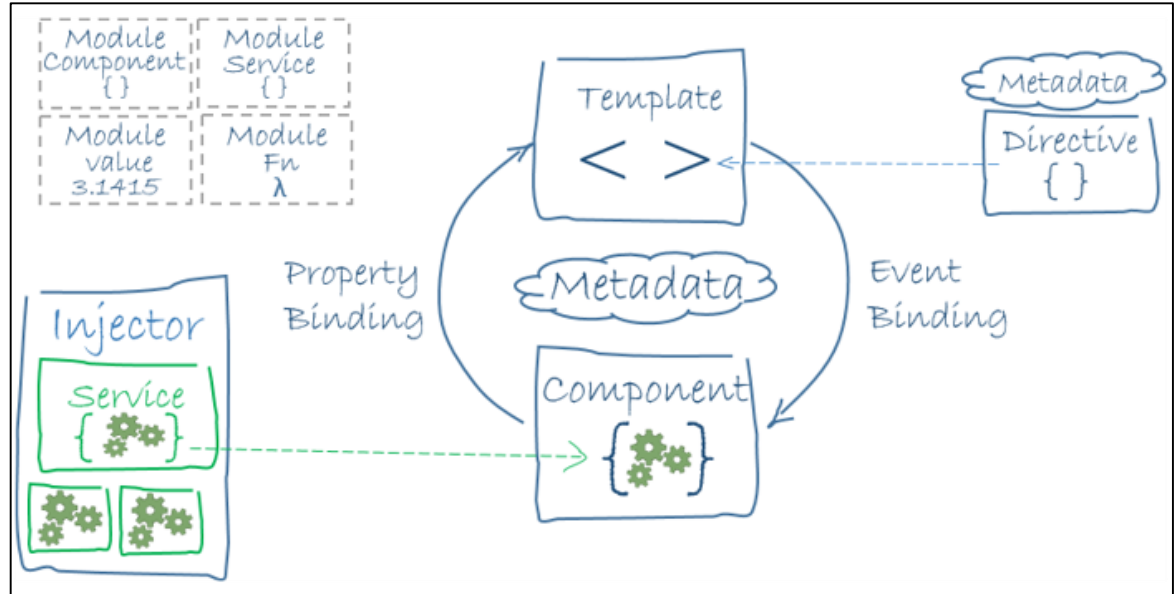
<https://www.webcomponents.org>

Angular

Définition

Framework proposé par Google en 2016, en typescript s'exécutant sur la technologie des composants Web.

- Application
- Modules
- Components
- Services
- Directives
- DataBinding



Angular

Assemblage

- Application - un ensemble de modules.
- Modules - des blocs cohésifs de code dédié à un domaine d'application, un flux de travail ou un ensemble de fonctionnalités étroitement liées.
- Composant - HTML + SCSS + TS
Joue le rôle de vue et de contrôleur
L'expérience utilisateur.

Angular

Assemblage

- Services: Fonctionnalité nécessaire à la coopération des composants d'une application.
Ex. Récupérer et stocker de données, valider les entrées de l'utilisateur.

Assemblage

- Data binding : assure la communication entre un composant et le DOM:
 - Interpolation: `{{ value }}` //simple binding
`Name: {{ user.name }}`
 - Property binding: `[property]="value"` //inputs
`<input type="email" [value]="user.email">`
 - Event binding: `(event)= "function"` //outputs
`<button (click)="cookBacon()"></button>`
 - Two-way data binding: `[(ngModel)]="value"` //two-way
`<input type="email" [(ngModel)]="user.email">`

Angular

Assemblage

- Directives HTML: des marqueurs sur un élément DOM qui indiquent au compilateur HTML d'AngularJS de transformer l'élément DOM et ses enfants.
 - Composants : `<app></app>`
 - Directives structurelles: `*ngFor`, `*ngIf`, `*ngIf else`
 - Directives d'attributs

Angular

Assemblage

- Directives HTML structurelles: ajoutent et suppriment des éléments DOM:

***ngFor**

```
<li *ngFor="let item of items">{{item}}</li>
```

***ngIf**

```
<p *ngIf=" newItem === " ">Please enter a value!</p>
```

Angular

Assemblage

- Directives HTML structurelles

***ngIf else**

```
<div *ngIf="isLoggedIn; else loggedOut">
```

```
  Welcome back.
```

```
</div>
```

```
<ng-template #loggedOut>
```

```
  Please, login.
```

```
</ng-template>
```

Angular

Assemblage

- Directives structurelles

[ngSwitch]

```
<div [ngSwitch]="hero?.emotion">  
  <app-happy-hero *ngSwitchCase="'happy'" [hero]="hero"></app-happy-hero>  
  <app-sad-hero *ngSwitchCase="'sad'" [hero]="hero"></app-sad-hero>  
  <app-confused-hero *ngSwitchCase="'confused'" [hero]="hero"></app-confused-  
hero>  
  <app-unknown-hero *ngSwitchDefault [hero]="hero"></app-unknown-hero>  
</div>
```

Angular

Assemblage

- Directives d'attributs: Modifient l'apparence/rôle des éléments du DOM

[ngClass]

`[ngClass]="{'btn-primary': newItem !== ' ', 'btn-default': newItem === ' '}"`

[ngStyle]

`[ngStyle]="{'background-color': celeb.status === 'Dead' ? 'red' : 'green' }"`

[input] Passage de paramètres aux composants

`<app-cart [items]="rootItems"></app-cart>`

(output) Recevoir des sorties des composants

`<app-cart (itemAdded)="onItemWasAdded($event)"></app-cart>`

Angular - directives

Examples

Folder - directives

```
<button  
  (click)="onAddItem()"  
  class="btn"  
  [ngClass]="{'btn-primary': newItem !== ' ', 'btn-default': newItem === ' '}">Add Item</button>
```

```
<p *ngIf=" newItem === ' '">Please enter a value!</p>  
<hr>
```

```
<ul class="list-group">  
  <li  
    class="list-group-item"  
    *ngFor="let item of items; let i = index">{{ item }}</li>  
</ul>
```

cart.component.html

```
<div class="grid">  
  <p>Data </p><input type="text"  
  [(ngModel)]="newItem">  
</div>  
<div class="grid">  
  <p>Databind </p><p> {{newItem}}</p>  
</div>
```

databind.component.html

Angular

Assemblage

TypeScript

Un langage orienté à objets aux types génériques:

Ex.: `mayVar: any;`

Les composants Angular sont d'abord écrits en TypeScript et compilés en JavaScript du côté client

Angular

Assemblage

Decorateurs TS

Indiquent comment le composant doit être traité, instancié et utilisée en temps d'exécution.

```
//Decorateurs de classes
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent { }
```

Angular

Assemblage

Decorateurs TS

```
//decorateurs d'attributs
@Input() name:any;
@Output() valueChanged = new EventEmitter<string>(); //an event

onUserInput(event){
valueChanged.emit(event.target.value);
}
```


Angular

Assemblage

SCSS - Sassy CSS

```
/* SCSS */
```

```
$blue: #3bbfce;
```

```
$margin: 16px;
```

```
.content-navigation {  
  border-color: $blue;  
  color: darken($blue, 9%);  
}
```

```
.border {  
  padding: $margin / 2; margin: $margin / 2; border-color: $blue;  
}
```

```
/* CSS */
```

```
.content-navigation {  
  border-color: #3bbfce;  
  color: #2b9eab;  
}
```

```
.border {  
  padding: 8px;  
  margin: 8px;  
  border-color: #3bbfce;  
}
```

Angular

Cycle de vie

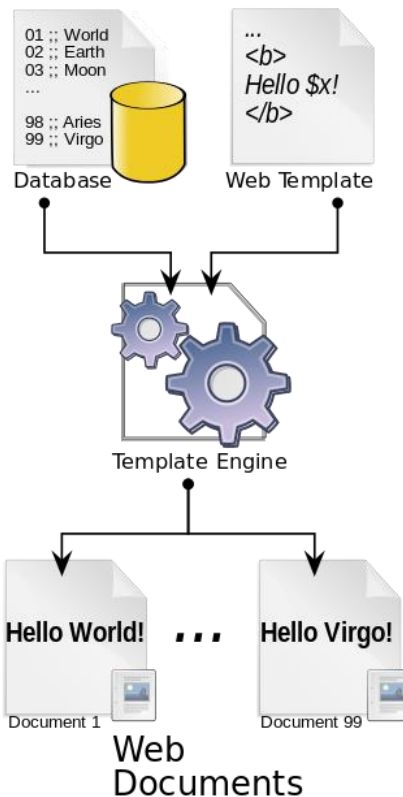
- `ngOnChanges()`: lorsqu'une liaison de données est établie
- `ngOnInit()`¹: après l'initialisation, soit le premier `ngOnChanges()`
- `ngDoCheck()`: après des modifications - `ngOnChanges()` et `ngOnInit()`
- `ngAfterContentInit()`¹: après la charge du TS - le premier `ngDoCheck()`.
- `ngAfterContentChecked()`: après tout `ngDoCheck()`.
- `ngAfterViewInit()`¹: après l'initialisation de la View
- `ngAfterViewChecked()`: après tout `ngAfterContentChecked()`
- `ngOnDestroy()`¹: juste avant la destruction d'un component.

¹- appelé une fois uniquement

Angular Templates

Ensemble de pages statiques conçues par un designer pouvant composer la présentation de sites/applications Web:

- Commerce en ligne
- Présence Web - Organisation, Individus
- Portail media
- Gestion de processus d'affaire
- ...

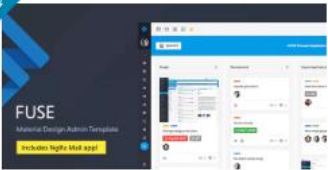


Angular Templates

Fuse
Fury
StartNG
Emporium
Elastic UI

...

Sofu



Fuse
Material Design Admin Template
Includes NgRx, MUI, and more


Tags: angular 7, admin, admin dashboard, admin template... See all tags

Fuse - Angular 8+ Material Design Admin Template
By scrn in Admin Templates

- Angular Material library
- Built-in ready to use applications
- Configurable layouts

\$28
★★★★★ (590)
13.4K Sales
Last updated: 20 Jul 19

Preview



Fury
Responsive Angular 8+ Material Design Template


Tags: angular 7, Angular2, AngularJS, admin, angular, ang... See all tags

Fury - Angular 8+ Material Design Admin Template
By vsurol in Admin Templates

- Modern Angular 8+ TypeScript Code
- Pure Angular + Material Design + Clean Code
- 100% Responsive - Mobile + Tablet + Desktop

\$24
★★★★★ (54)
1.3K Sales
Last updated: 30 Jul 19

Preview



startNG
Angular & Bootstrap 4
- Angular CLI
- Fully responsive & different layout
- 16 color styles
- Customizable
- Easy to use
- Well documented
- More components
- And much more...


Tags: angular 7, Full Calendar, admin, angular, angular 2, ... See all tags

StartNG - Angular 8+ Material Design Admin Template with Bootstrap 4
By theme season in Admin Templates

- Angular 8
- Bootstrap 4
- Angular CLI

\$24
★★★★★ (36)
727 Sales
Last updated: 10 Aug 19

Preview



EMPORIUM
Angular Material Design eCommerce Template
SUMMER COLLECTION


Tags: angular 7, SSR, angular, angular 2, angular 4, angular... See all tags

Emporium - Angular Material Design eCommerce Template
By theme season in Shopping

- Angular eCommerce
- Angular Universal
- Server Side Rendering (SSR), SEO Friendly

\$26
★★★★★ (27)
673 Sales
Last updated: 11 Aug 19

Preview



elastic ui
Flexible & Responsive
Angular 8+ Material Design
Angular Universal + Lazy Loading

elastic ui - Angular 8 Material Design & Redux Admin Template
By vsurol in Admin Templates

- Modern Angular 8+ TypeScript Code
- Pure Angular + Material + Service Worker
- Completely customizable layouts

\$24
★★★★★ (14)
627 Sales
Last updated: 10 Sep 19

Angular

Démarrage

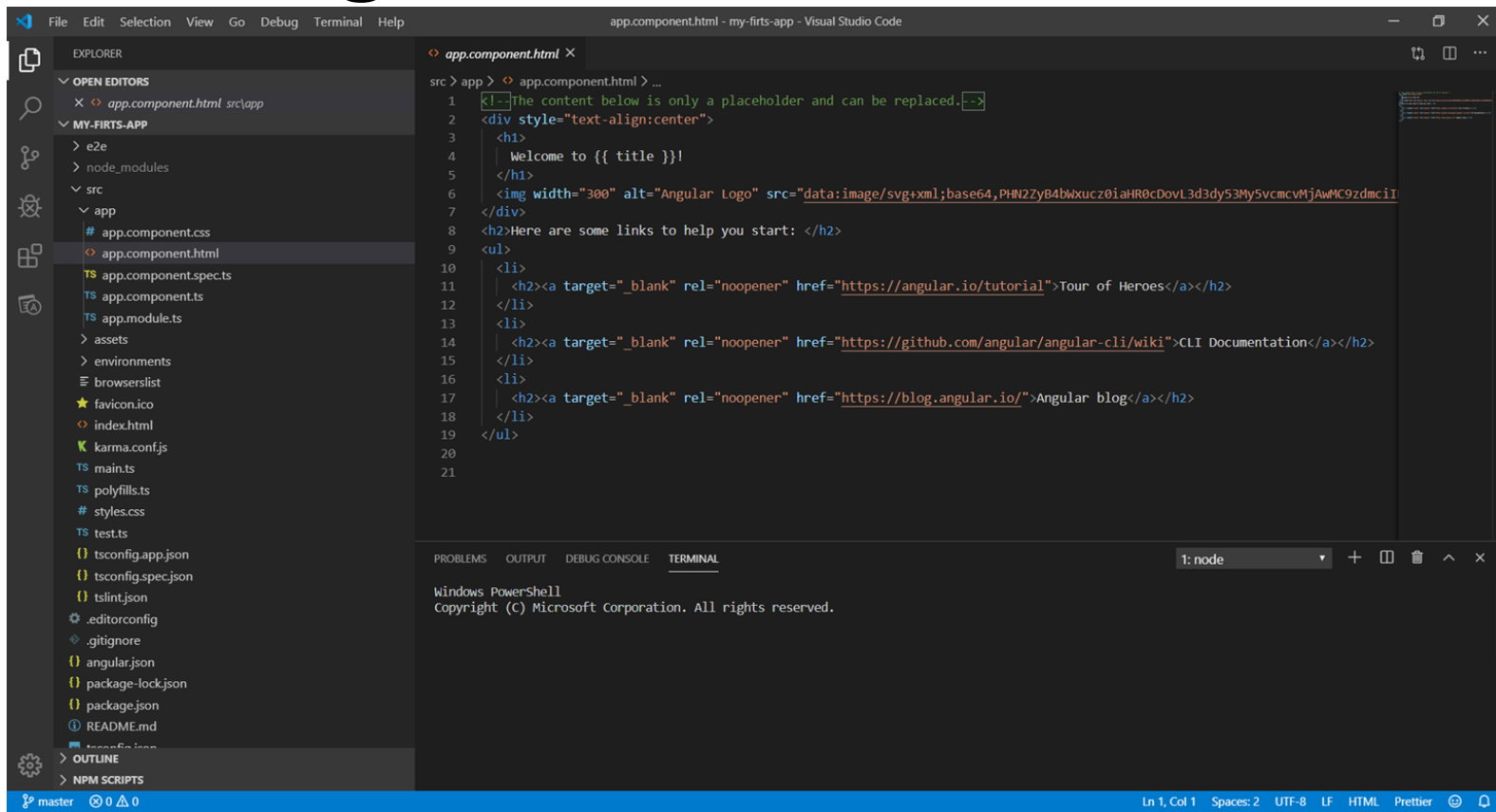
Workflow (avec Visual Studio Code)

- Installer Nodes.js : y compris NPM (Nodes Packaged Manager) et un serveur web
- Installer Angular CLI (Command Language Interface)
> npm install -g @angular/cli
- Créer d'un application Angular par CLI
> ng new my-first-app
- Installer les dépendances génériques
> npm install
..... Conception, programmation et tests
- Compiler, exécuter et tester
> npm start

Angular

Démarrage

my-first-app



Angular

Conclusion

Composants Web, frameworks & templates sont de plus en plus populaires dans la communauté Web, surtout l'Agile.

- Productivité avec souplesse!
- Réutilisez de templates, et centrez vos efforts surtout sur l'expérience utilisateur!

Tests avec utilisateur

Références

Composants Web

<https://html.spec.whatwg.org/multipage/custom-elements.html>

<https://www.webcomponents.org/>

https://developer.mozilla.org/fr/docs/Web/Web_Components

https://fr.wikipedia.org/wiki/Composants_web

<https://dev.to/thepassle/web-components-from-zero-to-hero-4n4m>

<https://lerjen.me/introductions-aux-composants-web-webcomponent/>

Angular

<https://angular.io/>

<https://angular.io/guide/lifecycle-hooks>

<https://www.typescriptlang.org/index.html>