



Service Web .Net

Cours n°5 : Application Web

Pierre-Loïc CHEVILLOT – Sébastien BEREIZIAT

Pierre-loic.chevillot@capgemini.com – sebastien.bereiziat@capgemini.com

Plan

- Développer pour tous les navigateurs
- MVC
- Component based Architecture App

Développer pour tous les navigateurs

- Interface Homme Machine (User Interface)
- Affichage parlant pour un utilisateur, on parle de « Fonctionnel »
- UX : User eXperience -> resenti de l'utilisateur final sur l'application
 - Un utilisateur doit comprendre « instinctivement » a quoi sert son écran.
 - Cohérence des écrans, enchainement, design

Développer pour tous les navigateurs

- Web IHM = Client léger \neq Win IHM = smart client
- Smart client = rafraichissement tps réel
- Web IHM = rafraichissement à la demande

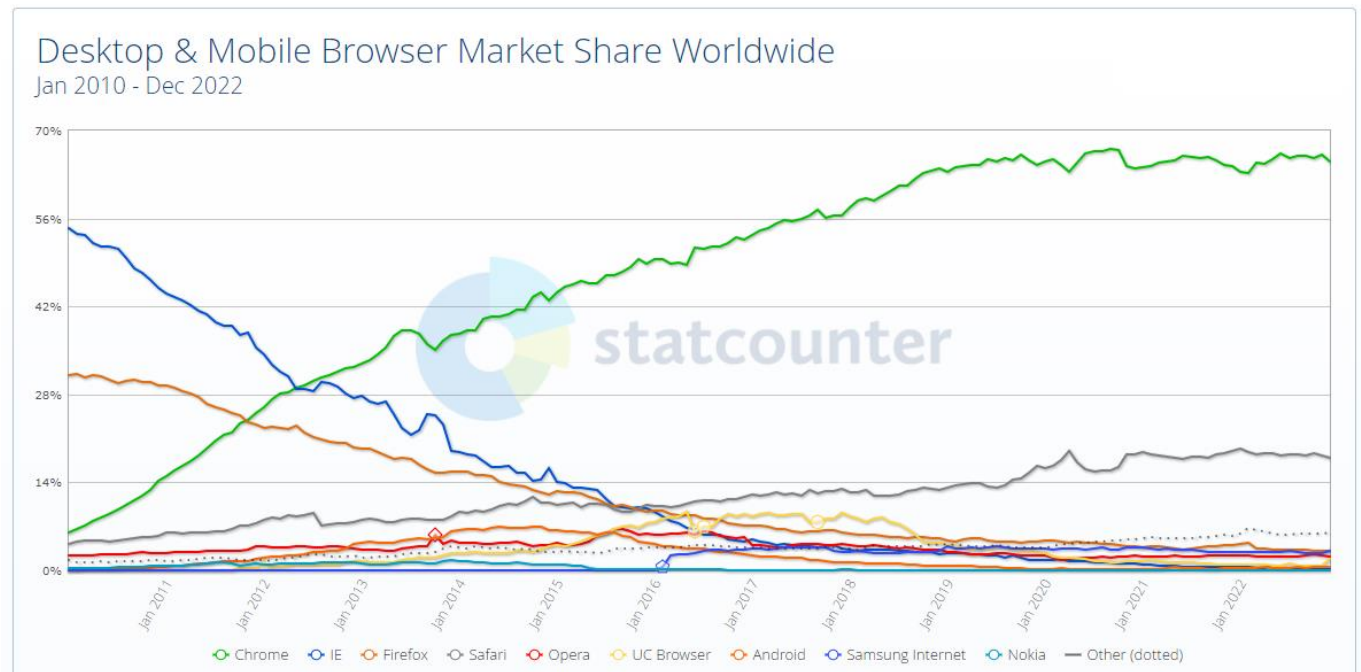
- **Avantage Web**
 - Pas d'installation, rien ne s'exécute sur le client
- **Inconvénient**
 - Accès limité aux ressources de la machine

Développer pour tous les navigateurs

- Changement de votre UI Web rapide
- Pas de couplage fort entre API & UI
- Pas de limitations dans les techno pour UI
- Front = Web
- Back = API + DB

Développer pour tous les navigateurs

- Navigateur en part de marché (2010/2020)

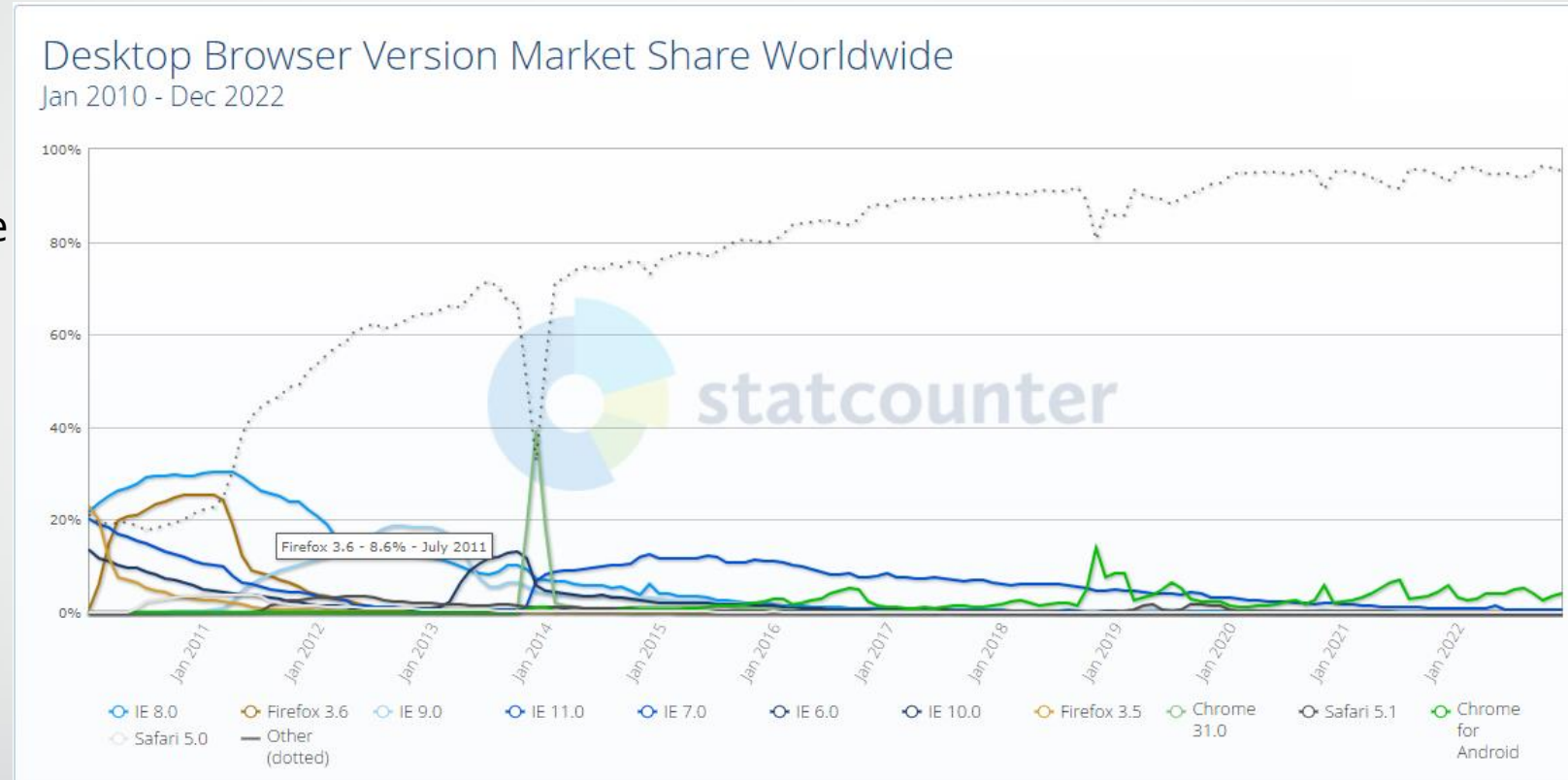


Chrome	Edge	Safari	Firefox	Opera	360 Safe
66.14%	10.98%	9.01%	7.21%	3.3%	1.03%

Desktop Browser Market Share Worldwide - December 2022

Développer pour tous les navigateurs

Version de navigateur entre
2010/2020



Développer pour tous les navigateurs

- En résumé : faire un site qui fonctionne sur tous les navigateurs (+smartphone) : c'est impossible ! Sauf si vous faites un... bloc de texte.
- Il existe des "bonnes pratiques" pour éviter au maximum les problèmes de compatibilité.

Développer pour tous les navigateurs

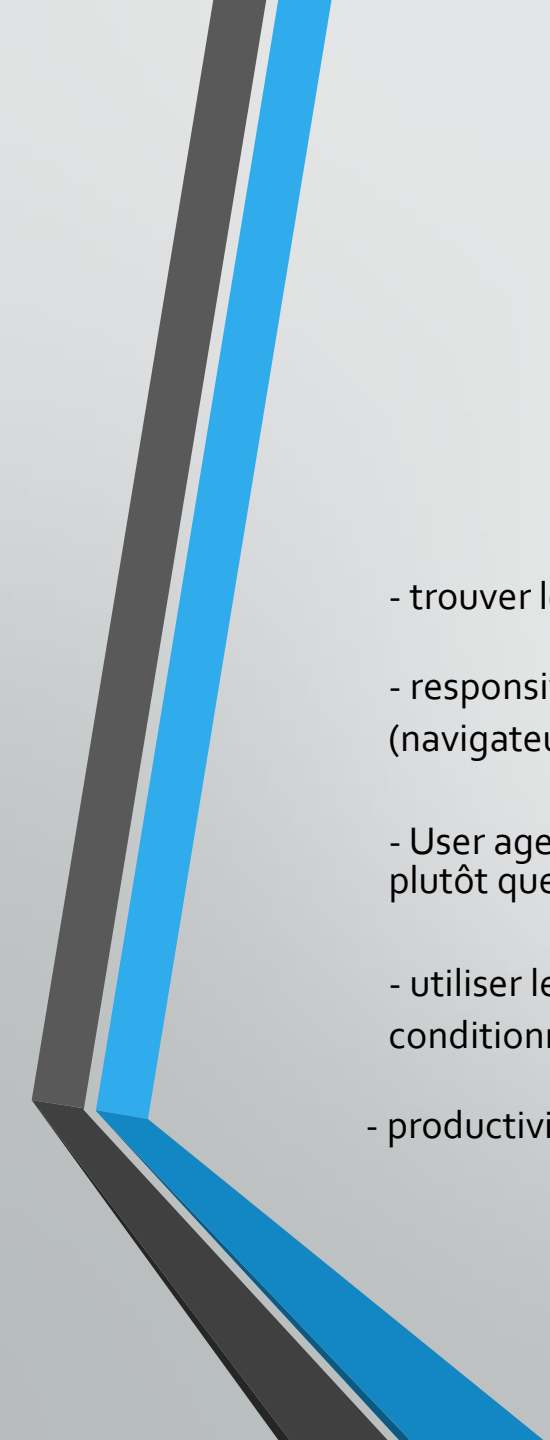
HTML5 : Ajoute tout simplement les objets Canvas, audio et vidéo à l'HTML

Javascript : agrégat de technologie (dom et ecmaScript et pleins API) non fiable car exécuté dans un environnement non maîtrisé.

Développer pour tous les navigateurs

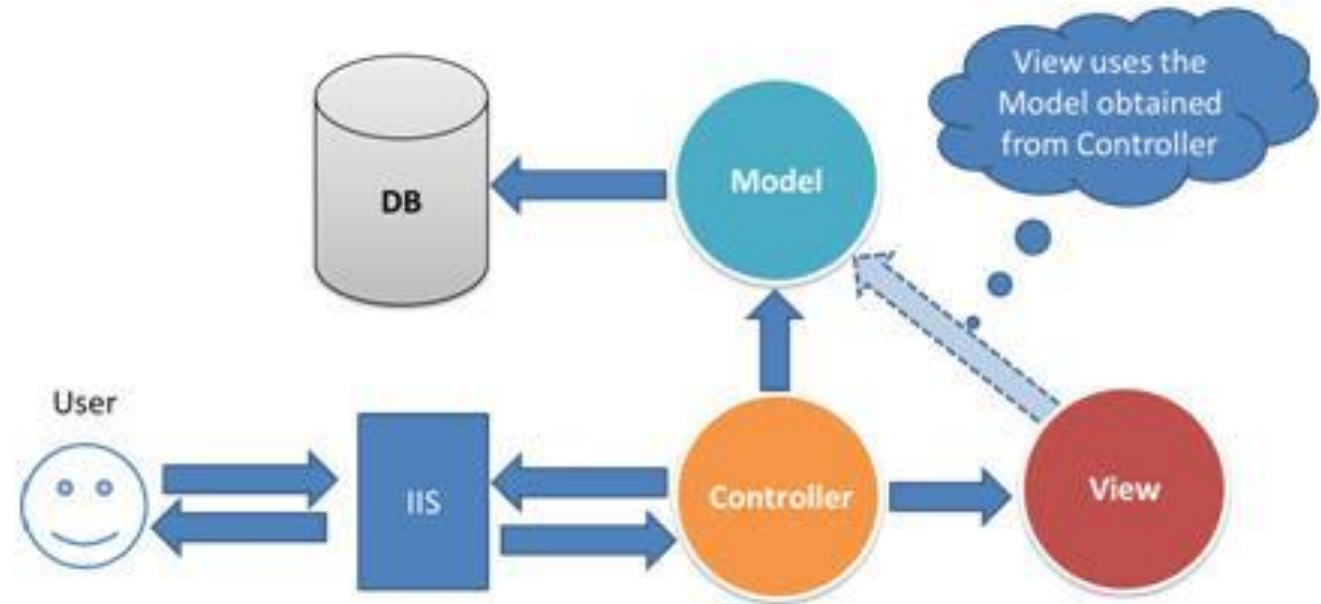
Les solutions :

- Utiliser les standards du web (W3C). Des plugins comme "HTML Validator" existent pour vous aider.
- Oublier le "pixel perfect".
- Utiliser les framework JavaScript (jQuery) et préprocesseur CSS (g6ogs).
- Afficher un message à l'utilisateur si le javascript n'est pas activé ou si sa version de navigateur n'est pas prise en compte.
- Utiliser la méthode "dégradation harmonieuse" ou "amélioration progressive".

- 
- trouver le plus petit dénominateur commun (sélecteur css marche partout).
 - responsive web design : visite et utilisable quelque soit le contexte d'utilisation (navigateur + résolution). utiliser media query dans css (selectionne une feuille de style et javascript selon le contexte).
 - User agent : a ne pas utiliser ! Car parfois renvoie la mauvaise information ! Demander plutôt "qu'est ce que tu sais faire ?" plutôt que "qui es-tu ?"
 - utiliser le modernizr : détection de fonctionnalité, chargement de fichier conditionnel (polyfills JS : remplace des trucs non géré par certains navigateurs)
 - productivité : ne pas lier css ou html et javascript.

MVC

- Les objets de modèle sont les parties de l'application qui implémentent la logique du domaine de données de l'application. C'est une simple classe pouvant cependant implémenter la logique de validation des données.
- Les vues sont les composants qui affichent l'interface utilisateur (IU) de l'application. Cette interface utilisateur est créée à partir des données du modèle.
- Les contrôleurs sont les composants qui gèrent les interventions de l'utilisateur, exploitent le modèle et finalement sélectionnent une vue permettant de restituer l'interface utilisateur. Dans une application MVC, la vue sert uniquement à afficher les informations ; le contrôleur gère les entrées et interactions de l'utilisateur, et y répond.



Javascript

- Est exécuté par le navigateur.
- Permet de manipuler et d'interagir avec la DOM HTML de ce dernier.

Ce que peut faire le javascript :

https://www.w3schools.com/js/js_examples.asp

What can JavaScript do?

JavaScript can change HTML content

JavaScript can change HTML attributes

JavaScript can change CSS style

JavaScript can hide HTML elements

JavaScript can show hidden HTML elements

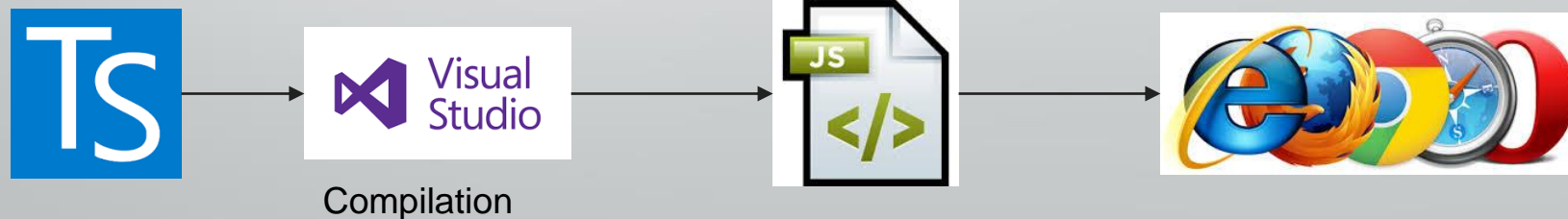
Javascript : les limites

- Est exécuté par le navigateur.
- Pas d'accès au disque dur.
- Langage non typé et non compilé
- Pas d'"intelligence"
- Difficulté de maintenance et de refactorisation du code



TypeScript

- TypeScript est un langage de programmation libre et open-source développé par Microsoft qui a pour but d'améliorer et de sécuriser la production de code JavaScript. C'est un sur-ensemble de JavaScript (c'est-à-dire que tout code JavaScript correct peut être utilisé avec TypeScript). Le code TypeScript est transcompilé en JavaScript, pouvant ainsi être interprété par n'importe quel navigateur web ou moteur JavaScript. Il a été cocréé par Anders Hejlsberg, principal inventeur de C#.



TypeScript : les avantages

- Langage typé et compilé
- Intellisense
- Simplicité de maintenance et de refactorisation du code

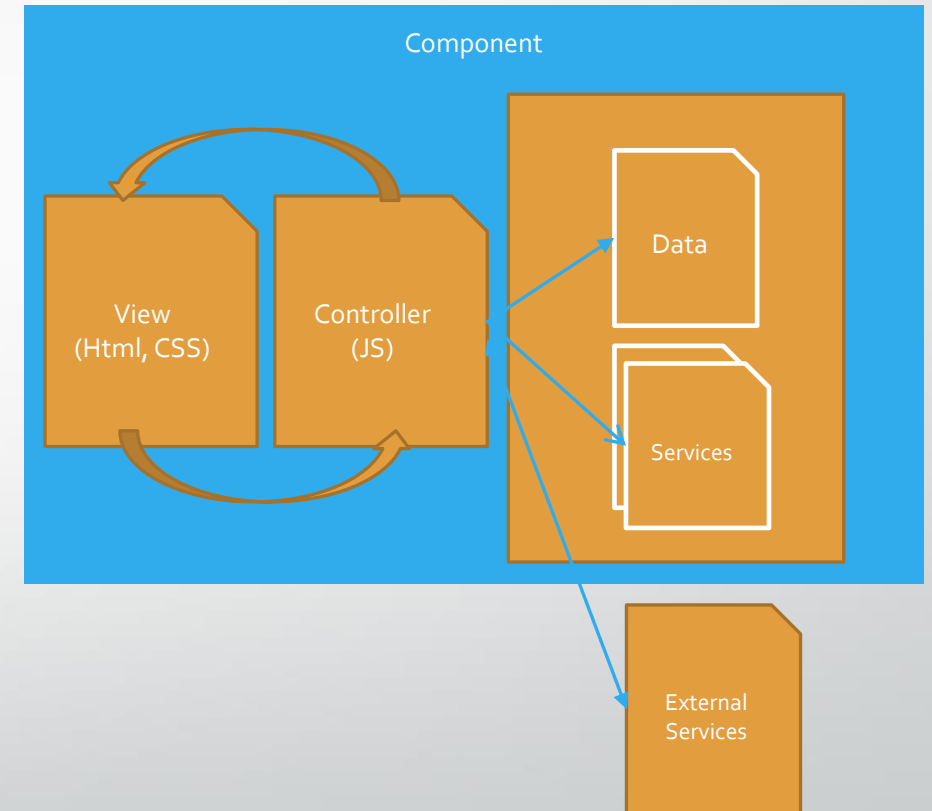
<https://www.typescriptlang.org/docs/handbook/declaration-files/by-example.html>

Component based Architecture App

- Angular/React est basé sur une architecture « Component-based »
- Les composants sont des petits bout de software indépendants les uns des autres, avec leurs propres comportement, leurs APIs, leurs UI. Chaque composant utilise en interne le pattern MVC. Les vues, le contrôleur et l'accès aux données sont obligatoirement séparés dans plusieurs fichiers.
- Un composant contient toutes les méthodes (API, UI, Helper, contrôles) lui permettant de se rafraichir, d'avoir l'autonomie nécessaire pour demander les données à exposer, et de modifier l'UI selon les besoins. La principale différence par rapport au MVC classique ou toutes ces méthodes sont explosées dans les différentes couches.

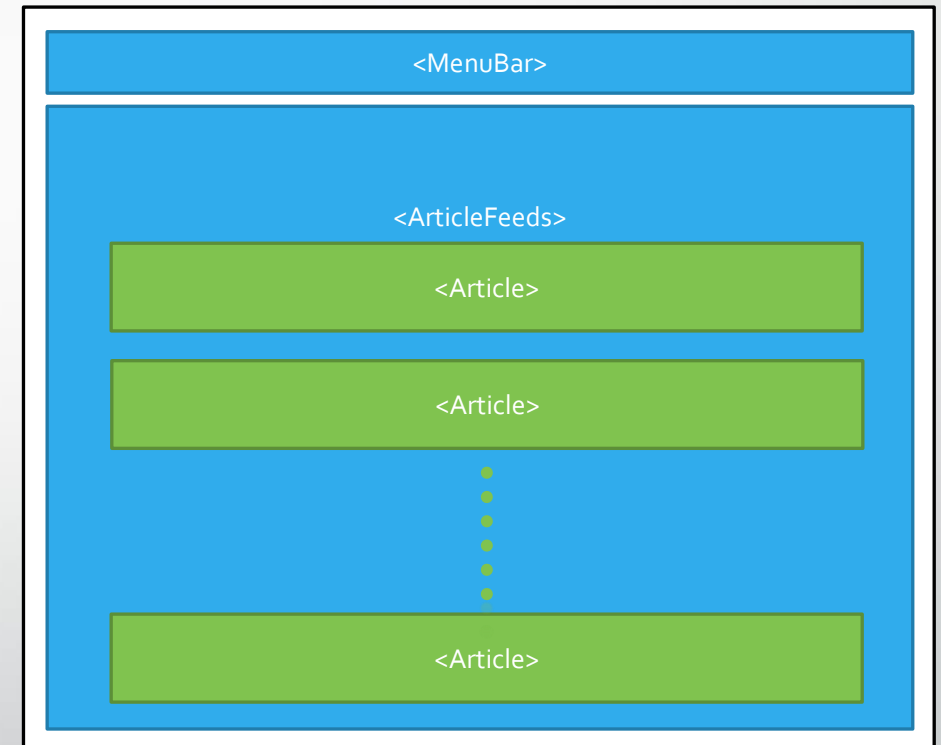
Component based Architecture App

- Angular est basé sur une architecture « Component-based »
- Les composants sont des petits bout de software indépendants les uns des autres, avec leurs propres comportement, leurs APIs, leurs UI. Chaque composant utilise en interne le pattern MVC. Les vues, le contrôleur et l'accès aux données sont obligatoirement séparés dans plusieurs fichiers.
- Un composant contient toutes les méthodes (API, UI, Helper, contrôles) lui permettant de se rafraichir, d'avoir l'autonomie nécessaire pour demander les données a exposer, et de modifier l'UI selon les besoins. La principale différence par rapport au MVC classique ou toutes ces méthodes sont exposées dans les différentes couches.



Component based Architecture App

- Les composants sont réutilisables, c'est-à-dire qu'un composant peut être copié une multitude de fois dans une interface pour afficher différentes informations (data), mais exactement de la même manière (design), et son comportement sera exactement le même.
- L'architecture CBA permet au différents composant de rafraichir l'interface utilisateur sans avoir besoin de rafraichir la page entière.



Component based Architecture App

- Un module angular est une concaténation des composants angular permettant d'implémenter une fonctionnalité.
- Certains module permette d'exposer des composants généraux utilisable par d'autres module
 - Un module de service peut service a exposer au composant qui le références des endpoints vers des API.
 - Un module d'utilitaire peut servir a exposer des composants généraux qui peuvent être des modèles de design afin d'encapsuler d'autres composant. Exemple : Un composant Card sur être créer et celui-ci permettra d'exposer des composant article, texte, widget, etc. encapsulé toujours sous le même design, et avec les actions spécifiques a ce composant (drag&drop par exemple) dans toute l'application.

Binding JSON

- Lors de l'appel à une WebAPI par un composant Angular, nous attendons un retour au format JSON
- Le langage TypeScript permet de mapper explicitement ce retour sous la forme d'un objet au sens langage objet, avec la description dans la classe correspondante. Les éléments du fichiers JSON sont mappés directement selon les noms des propriétés de la classe TypeScript.
- Les composants Angular utilisent l'objet TypeScript pour afficher les propriétés dans leur interface UI.
- Les objets TypeScript peuvent être observable, c'est-à-dire que lorsqu'une modification se fait dans le contrôleur du composant, ceci averti directement l'UI qui affiche la modification de l'objet TypeScript à l'utilisateur.

- Quel est le meilleur front ?
 - C'est comme la mode cela change tous les ans
- 2022 web framework among developers WW + adoption trend 2022

