

Ionic, développement de Web Components



Intervenant :



Renaud Dubuis

[View profile](#)

 LinkedIn®

Approche pédagogique.

Participants. (Tour de table)

Le tour de table initial favorise la création du groupe et permet la contextualisation des réponses aux questions individuelles.

Evaluation des pré-requis.

Le tour de table initial et les premiers exercices ont pour but l'évaluation effective des participants au regard des pré-requis.

Récapitulatif (matinal).

Chaque journée commence par un exercice collectif de restitution des concepts abordés la veille. L'objectif étant de renforcer (répétition) les connaissances acquises pour les utiliser comme socle lors de la journée à venir.

Concertation personnelle.

Le formateur passera assister les participants individuellement aussi souvent que possible.

Les participants sont invités à le solliciter pendant la journée de formation ou pour revenir sur un point particulier en fin de journée.

Travaux Pratiques.

L'acquisition des concepts abordés est découpée proportionnellement:

- **60%** de manipulation pratique.
- **40%** d'appports théoriques.

La mise en oeuvre des travaux pratiques se déroule en 3 phases, permettant la reproduction en autonomie des exercices.

1. Enoncé > 2. Démonstration > 3. Manipulation

Version numérique du support de formation.

Pour renforcer le confort de lecture et de manipulation (**copier/coller, liens cliquables, coloration du code**) le support est également distribué en version numérique. **au format PDF.** (sous license Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)).

Présentation des participants.

Afin d'animer ensemble la formation et de constituer notre groupe il est utile de nous présenter en fournissant les informations lés pour le déroulement de ce stage.

Exemple

- Identité.
- Rôle au sein du groupe.
- Positionnement sur les pré requis.
- Attente de cette formation.
- Précision.

(identité) Bonjour je suis Renaud Dubuis.

(rôle) Je suis l'animateur de notre formation. (pré requis) En tant qu'architecte Font-End je suis amené à en maîtriser les différents outils. Je connais très bien HTML, CSS et JavaScript.

(attente) Je souhaite partager avec vous ces compétences par cette formation.

(précision) Etant dyslexique, je parfois des soucis d'orthographe, je vous prie de m'en excuser. Je tutoie également les développeurs avec qui je code, merci de me dire si vous préférez le vouvoiement.

Presentation personnelle

identité :

rôle :

Organisation pratique : repas, pause, temps de questions/réponses.

Lors d'une formation INTRA (dans les locaux de l'entreprise) les horaires sont adaptés en fonction de votre rythme habituels.

9:00 : Début de formation.

10:30: pause de la matinée (15 à 20 minutes)

Pause déjeuner: (choisir un horaire et une durée)

15:30: pause de l'après midi. (15 à 20 minutes)

17:00 : Temps des questions spécifiques.

17:30 : fin de journée.

Le récapitulatif matinal est noté et tenu à jour par le formateur dans un fichier nommé **RECAPITULATIF.md**. Ce fichier vous sera remis à la fin de la formation.

Les questions posées font soit:

1. L'objet d'une réponse immédiate.
2. L'objet d'une prise de note dans un fichier nommé **QUESTIONS.md** pour une réponse ultérieure avant les pauses ou en fin de journée, ou plus tard dans la formation.

Introduction du modèle de formation :

L'acquisition des concepts abordés est découpée proportionnellement :

- 60% de manipulation pratique.
- 40% d'apports théoriques.

La mise en oeuvre des travaux pratiques se déroule en 3 phases, permettant la reproduction en autonomie des exercices.

1. Ennoncé > 2. Démonstration > 3.Manipulation.

1. Ennoncé Verbal, avec l'appui du support de cours ou d'une documentation extérieure telle qu'une documentation en ligne ou croquis.

2. Démonstration Le formateur illustre pratiquement le point expliqué.

Durant ces deux premières phases il est recommandé de :

Se concentrer sur le concept abordé. Prendre des notes.

NE PAS essayer reproduire les manipulations en même temps que le formateur.

3. Manipulation Vous reproduisez les manipulations en vous appuyant sur vos prises de notes. ***N'hésitez pas à solliciter le formateur pour vous assister***



Présentation de la structure du support de formation :

Dans la mesure du possible le support se décompose en trois points.

1. Théorie > 2. Exemple > 3. Validation.

1. Théorie Explications théoriques internes au support ou basées sur une documentation en ligne.

Pourquoi utilisée une documentation en ligne :

Les solutions de développement telles que **ionic** évoluent très rapidement pour répondre au besoins techniques du marché et aux attentes des développeurs.

Les traductions peuvent souvent être approximative et très rapidement obsolètes. Beaucoup de solutions et de langages sont créés en langues anglaises **les concepts y sont souvent plus aisément présentés.**

Votre formation s'appuiera sur l'utilisation, la compréhension et l'usage de la documentation en ligne

2. Exemple Exemple local ou illustré en ligne.

3. Validation Espace de validation basé sur des questions écrites ou orales.



Exemple de structure :

Utiliser une documentation en anglais

Si les documentations techniques sont rédigées dans un soucis de simplicité, certaines expressions peuvent échapper au développeur non anglophone.

Installons **Google Translate** dans notre navigateur.

Saisir **google translate chrome** dans un moteur de recherche ou suivre le lien précédent.

Cliquez sur le lien Ajouter à Chrome



Pour effectuer une traduction sur une page en anglais :

1. Sélectionner les **texte désiré sur la page**.
2. Cliquez sur l'icône **Google Translate** qui apparaît à proximité du texte.

1. Install Ionic

```
npm install -g cordova ionic
```



Windows Developers: We recommend Visual Studio Community, which comes with everything you need, including starter templates!



First, install [Node.js](#). Then, install the latest Cordova and Ionic command-line tools in your terminal. Follow the [Android](#) and [iOS](#) platform guides to install required tools for development.

If you're new to the command line, read our [Terminal tutorial](#) >

Le texte traduit apparaît dans une "Info-Bulle" comme sur l'image ci-après.

The screenshot shows the Ionic website's 'Code with the CLI' section. It includes a button to download the CLI, a note about using the CLI tool, and a 'Code with the CLI' icon. A tooltip provides instructions for installing Node.js, Cordova, and Ionic command-line tools. It also includes a French translation of the instructions, options for extension settings, and a link to the 'PLUS' section.

Code with the CLI
Use our CLI tool to build your apps.

Angular

First, install Node.js. Then, install the latest Cordova and Ionic command-line tools in your terminal. Follow the Android and iOS platform guides to install required tools for development.

D'abord, installez Node.js. Ensuite, installez les derniers outils de ligne de commande Cordova et Ionic dans votre terminal. Suivez les guides de la plate-forme Android et iOS pour installer les outils requis pour le développement.

FRANÇAIS

OPTIONS D'EXTENSION

PLUS »

First, install Node.js. Then, install the latest Cordova and Ionic command-line tools in your terminal. Follow the [Android](#) and [iOS](#) platform guides to install required tools for development.

If you're new to the command line, read our [Terminal tutorial](#) >

1 Install Ionic

```
npm install -g cordova ionic
```



Windows Developers: We recommend [Visual Studio Community](#), which comes with everything you need, including [starter templates!](#)

Introduction :

Cette formation vous permettra de maîtriser la dernière version du framework/SDK Ionic basé sur Cordova ou Capacitor. Vous développerez des applications mobile et multi-plateformes en utilisant et concevant des Web Components portables pour les frameworks leaders du marché (Angular, React, Vue...).

Objectif(s) pédagogique(s) :

- Maîtriser l'environnement de développement hybride.
- Comprendre les Web Components.
- Mettre en œuvre les applications multi-plateforme.
- Développer une application mobile basée sur Ionic.
- Exploiter les outils de productivité proposés par Node.js.

Pré-requis :

Bonnes connaissances des technologies du Web et des outils modernes de développement Front-End.
Connaissances de base de JavaScript.



Configurer un environnement de développement moderne.

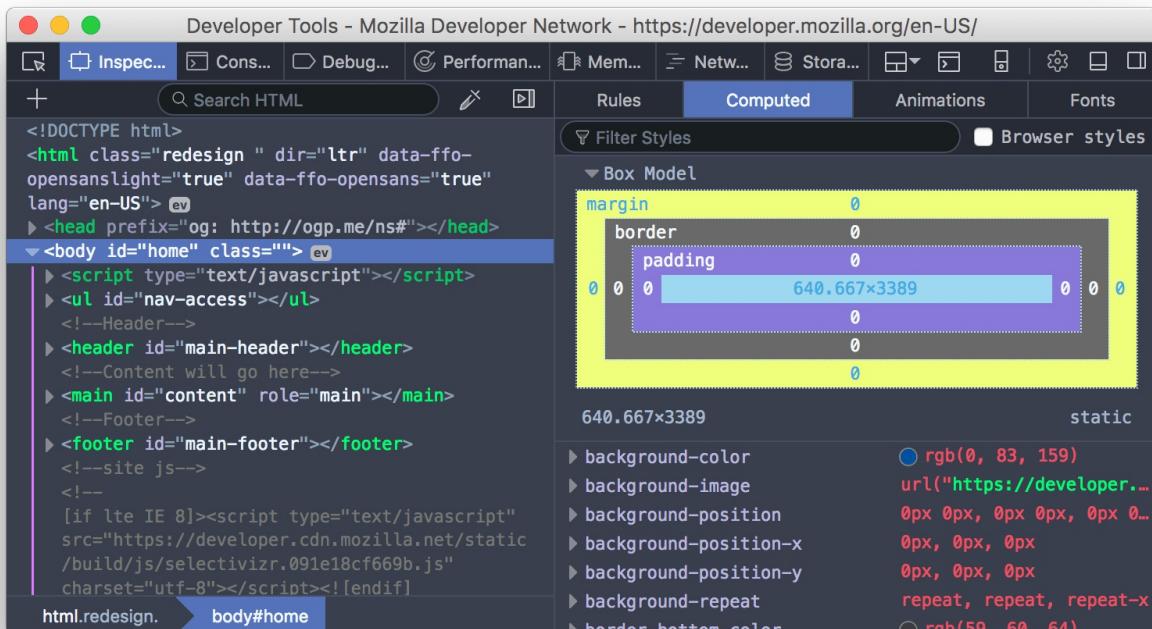
Configurer un environnement de développement moderne.

Pour programmer en JavaScript, un éditeur de texte suffit, il est intéressant de connaître l'offre en outillage autour de Node.

Un bon environnement telle que Chrome Examinez, modifiez, et déboguez du HTML, du CSS, et du JavaScript sur ordinateur, et sur mobile.

- Inspecteur
- Console web
- Débogueur JavaScript
- Moniteur réseau
- Performances

Inspecteur



Permet de voir et modifier une page en HTML et en CSS. Permet de

Environnement de développement. IDE et plug-ins.

Outils de développement les autres logiciels :

Pour programmer avec angular en JavaScript, un éditeur de texte suffit, il est intéressant de connaître l'offre en outillage autour de Node.

Nous utiliserons VS Code.

cf. pratique

- Git cf. ressources attention à ajouter **git au PATH windows!**
- Node.js cf. ressources
- VS Code cf. ressources

Vérifier des installations :

cf. pratique

Vérifier l'installation de git, node et npm (installation par node), ouvrir une invite de commande et saisir les commandes suivantes :

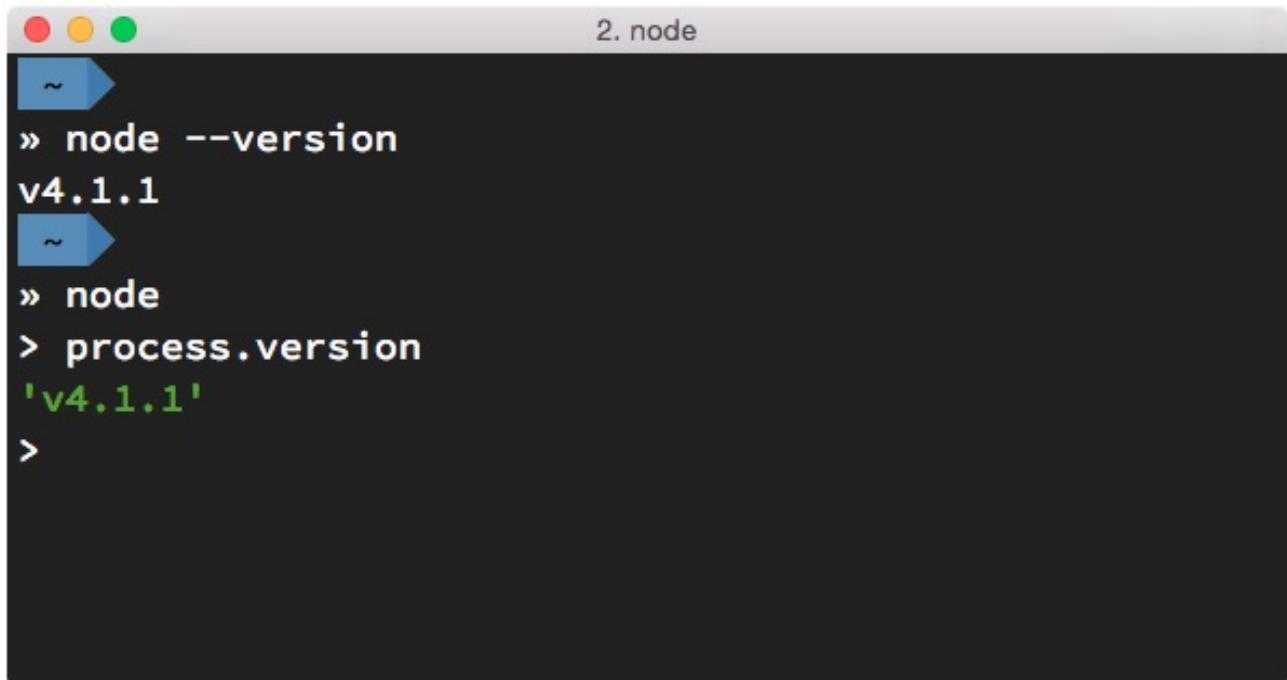
```
$> node --version  
x.x.x  
  
$> npm --version  
x.x.x  
  
$> git --version  
x.x.x
```

✓ Installation réussie !

Node.js utilitaire de développement.

Installation de Node.JS :

Il est fortement recommandé d'utiliser un interpréteur de commandes (terminal ou shell). Les systèmes d'exploitation modernes en proposent un, y compris les versions récentes de Windows.



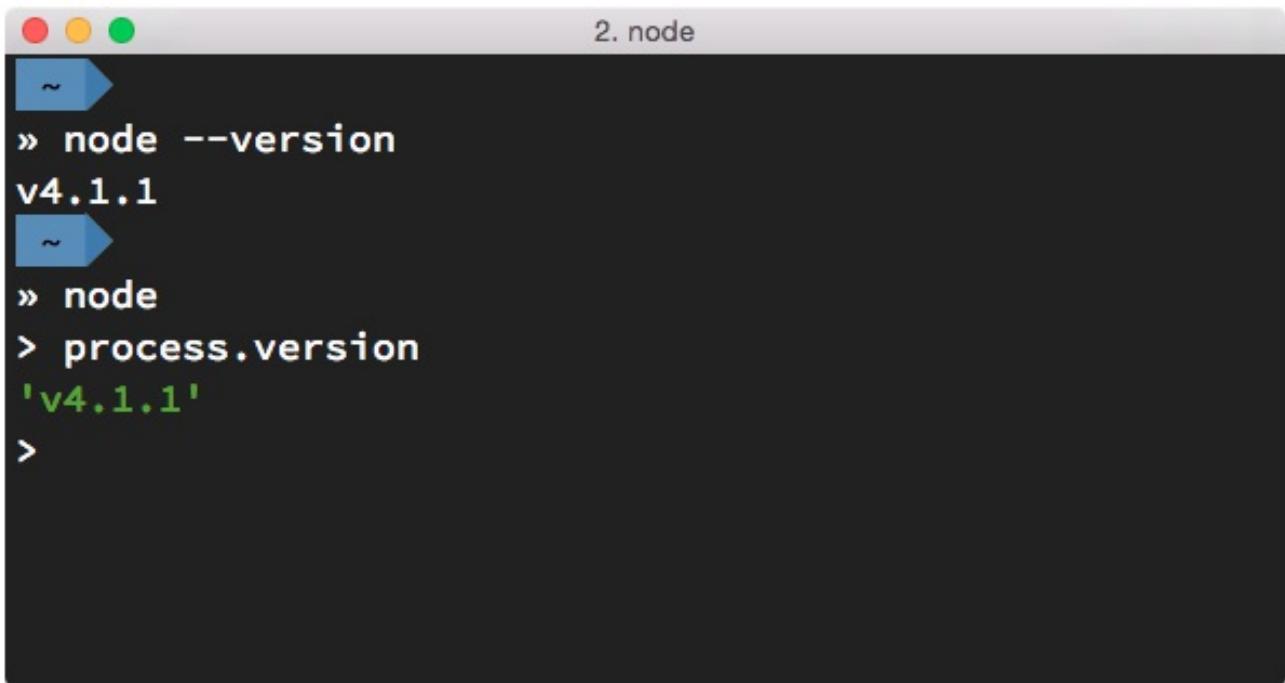
A screenshot of a macOS terminal window titled "2. node". The window has the standard OS X title bar with red, yellow, and green buttons. The terminal itself is black with white text. It shows the following command-line session:

```
~ » node --version
v4.1.1
~ » node
> process.version
'v4.1.1'
>
```

Si vous n'utilisez pas encore de terminal, voici une liste de recommandations non exhaustive pour vous aider :

- *OS X* : iTerm2, Terminal.app.
- *Linux* : GNOME Shell, Terminator.
- *Windows* : PowerShell, Console.

Aisance avec l'invite de commande windows.



```
2. node
~
» node --version
v4.1.1
~
» node
> process.version
'v4.1.1'
>
```

Il est utile de pouvoir ouvrir rapidement une invite de commande pointant directement sur le répertoire voulu.

Manipulation 1

MAJ + CLIQUE DROIT > Ouvrir une invite de commande au dossier

Manipulation 2

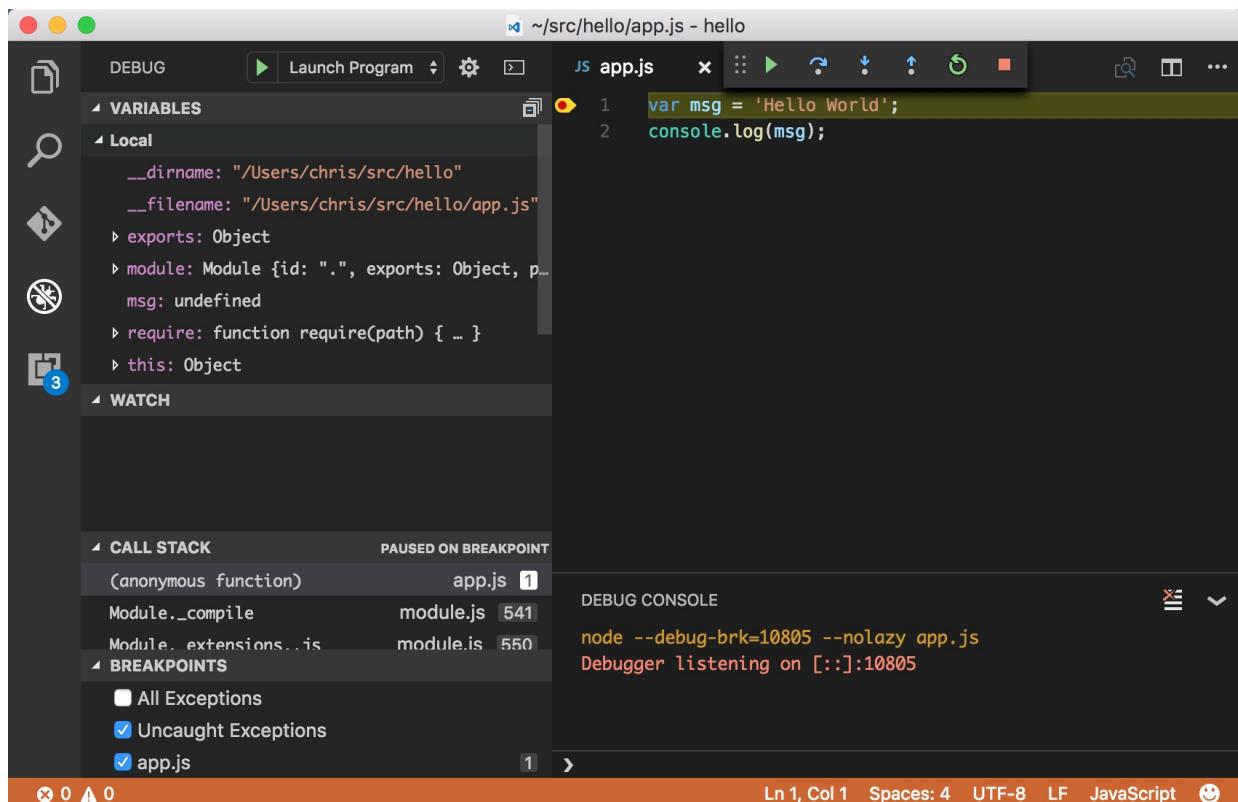
A l'aide de l'explorateur windows, se placer dans le dossier **workshops** Saisir **cmd + ENTER** dans la barre d'adresse

Manipulation 3

Depuis **VS Code** choisir **menu>Afficher>Terminal Intégré**

Présentation de l'éditeur, les plug-ins indispensables.

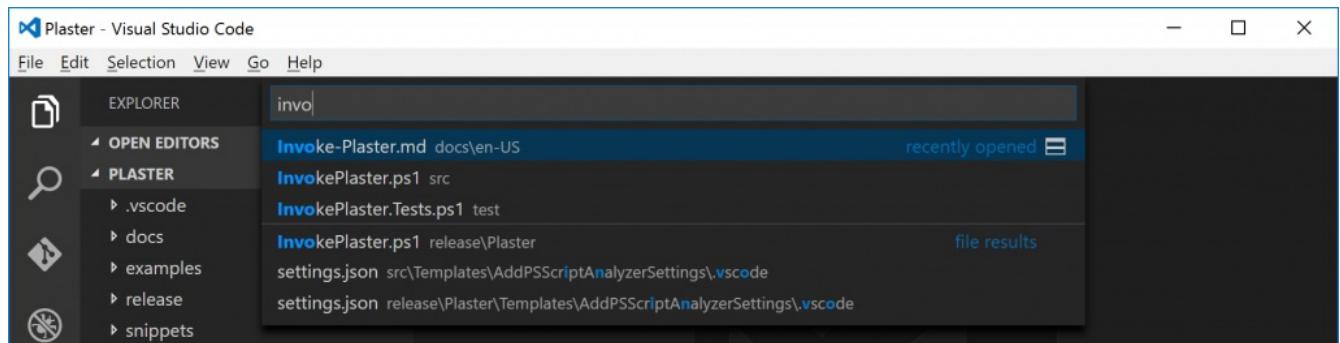
VS Code apporte toutes les fonctionnalités d'un éditeur moderne, sans nécessiter un investissement de formation.



Fonctionnalités principales:

- Palette de commande
- Gestion des fichiers et des projets
- "Snippets"
- Console
- Débuggeur
- Terminal intégré
- Intégration des plugins

La palette de commande



C'est le centre de contrôle de votre IDE Pour ouvrir la palette de commande, appuyez sur **Ctrl+Maj+P**, tapez le nom d'une commande, les suggestions les plus cohérentes s'affichent dans une liste, validez avec la touche ENTRÉE.

- Utilisation des commandes
- Saisie intuitive

Préparer son environnement.

Installer Node n'est pas très compliqué. Il existe cependant plusieurs mécanismes d'installation. Ces mécanismes vont du téléchargement d'un installateur à une compilation manuelle *via* un terminal.

Les installateurs, les binaires et les sources de Node sont disponibles sur le site officiel <https://nodejs.org/en/>

Téléchargez l'installateur adapté.

Vérifier l'installation **node et npm** (installation par node), ouvrir une invite de commande et saisir les commandes suivantes :

À noter il est recommandé une version de **Node 10+ et NPM 6+**

```
$> node --version  
x.x.x
```

```
$> npm --version  
x.x.x
```

Installer des modules tiers

Il existe deux modes d'installation avec npm :

global (machine)

```
$> npm install --global MODULE_NAME  
$> npm i -g MODULE_NAME
```

local (dossier courant)

```
$> npm install MODULE_NAME  
$> npm i MODULE_NAME
```

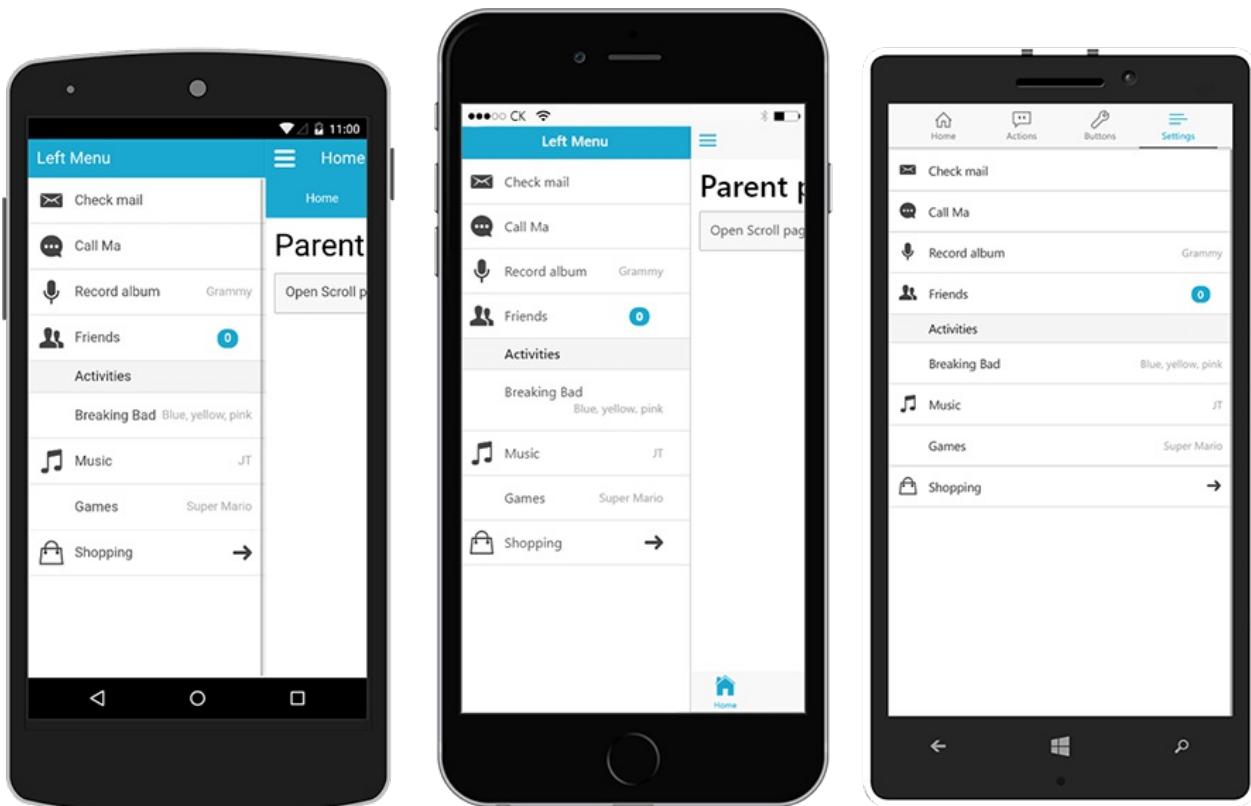
Installation des modules (CTRL+C ?)

Installer les modules suivants :

```
$> npm i -g lite-server
```

lite-server facilite le développement

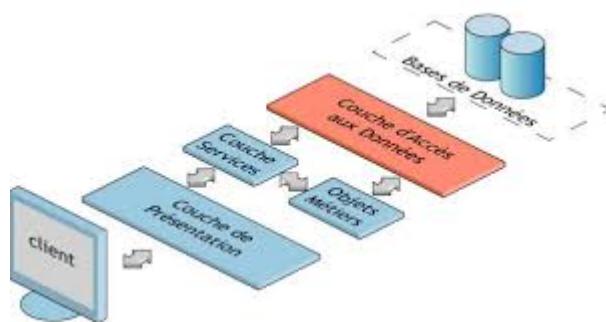
Synchronisation “multi-device”.



Il est utile de pouvoir afficher une application sur plusieurs terminaux.
lite-server fournit cette fonctionnalité par défaut

Architectures n-tiers, applications et composants distribués. Théorème de Brewer (CAP).

Architectures n-tiers



Aussi appelée *architecture à trois niveaux ou architecture à trois couches*, est l'application du modèle plus général qu'est le **multi-tiers**.

L'architecture logique du système est divisée en trois niveaux ou couches :

- Présentation ;
- Traitement ;
- Accès aux données.

C'est une architecture basée sur l'environnement client-serveur.

Architecture distribuée

Les architectures distribuées reposent sur la possibilité d'utiliser des objets qui s'exécutent sur des machines réparties sur le réseau et communiquent par messages au travers du réseau, les ressources ne se trouvent pas au même endroit ou sur la même machine.

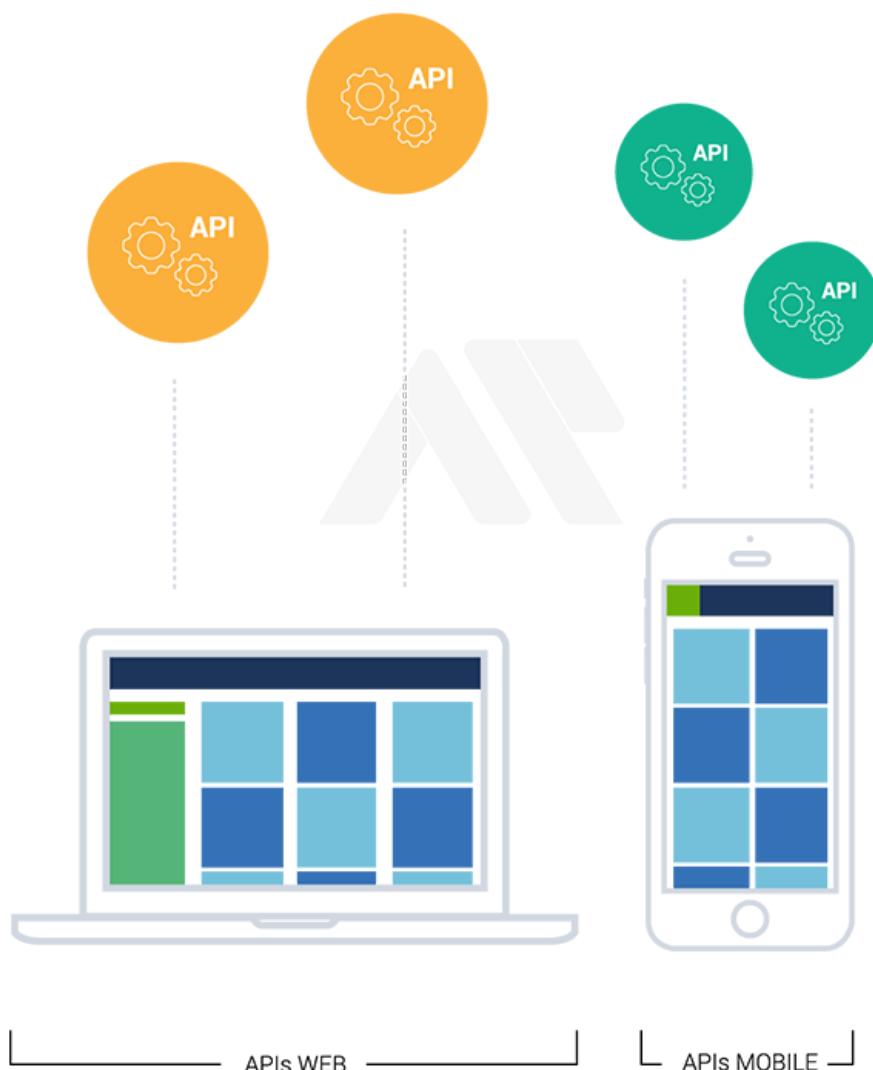
Internet est un exemple de réseau distribué puisqu'il ne possède aucun noeud central.

Composants distribués

Un composant logiciel est un élément constitutif d'un logiciel destiné à être incorporé en tant que pièce détachée dans des applications.

Les paquets, les bibliothèques logicielles, les exécutables, les fichiers, les bases de données ou encore des éléments de configuration (paramètres, scripts, fichiers de commandes) sont des composants logiciels1.

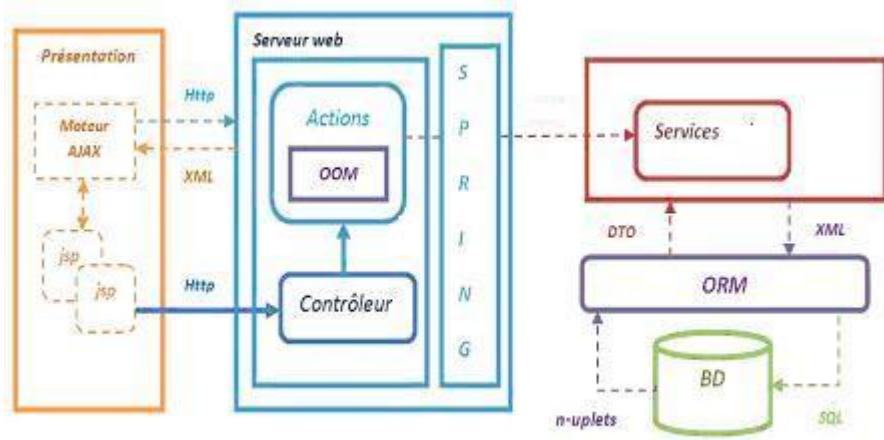
API Mobiles.



Les API mobiles sont basées sur l'utilisation de la technologie WOA. La création de ces services est devenue plus simple en utilisant des protocoles Web simplifiés, par ex. REST et JSON (JavaScript Object Notation).

Ces protocoles sont beaucoup plus faciles pour les développeurs web, car ils nécessitent moins de CPU et de bande passante. Ils sont plus reconnus à cause des grandes plates-formes sociales, telles que **Facebook, Amazon et Twitter**, etc.

Les trois couches



Couche de présentation

Ou **interface homme-machine**, elle peut être réalisée par une application graphique ou textuelle, être représentée en HTML pour être exploitée par un navigateur web ou être utilisée par un téléphone portable.

On conçoit facilement que cette interface peut prendre de multiples facettes sans changer la finalité de l'application.

La couche de présentation relaie les requêtes de l'utilisateur à destination de la couche de traitement, et en retour lui présente les informations renvoyées par les traitements de cette couche.

Il s'agit donc ici d'un assemblage de services métiers et applicatifs offerts par la couche inférieure.

Couche de traitement

Partie fonctionnelle de l'application, implémentant la logique métier, et décrivant les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs.

Les différentes règles de gestion et de contrôle du système sont mises en œuvre dans cette couche. **La couche de traitement offre des services applicatifs et métier à la couche de présentation.**

Pour fournir ces services, elle s'appuie, sur les données du système, accessibles au travers des services de la couche inférieure.

Couche d'accès aux données

Partie gérant l'accès aux données de l'application. Ces données peuvent être propres à l'application, ou gérées par une autre application.

La couche de traitement n'a pas à s'adapter, l'accès aux données est et uniforme (couplage faible).

Données propres à l'application Ces données sont pérennes, car destinées à durer dans le temps, de manière plus ou moins longue, voire définitive.

Les données peuvent être stockées indifféremment dans de simples fichiers dans différents formats textuels (XML) ou binaires.

Le pattern Data Access Object (DAO), consiste à représenter les données du système sous la forme d'un modèle objet.

La représentation du modèle de données objet en base de données (appelée persistance) peut s'effectuer à l'aide d'outils tels que Hibernate.

Données gérées par une autre application Les données ne sont pas fournies par l'application qui s'appuie sur une autre application à fournir ces informations.

Quelles fonctionnalités attendre d'un Backend Mobile ?

Pour édifier votre projet mobile, il vous est possible de choisir entre une application locale et une application connectée. Ce choix influencera alors votre besoin en terme de solution d'administration, de développement et d'animation de votre app.

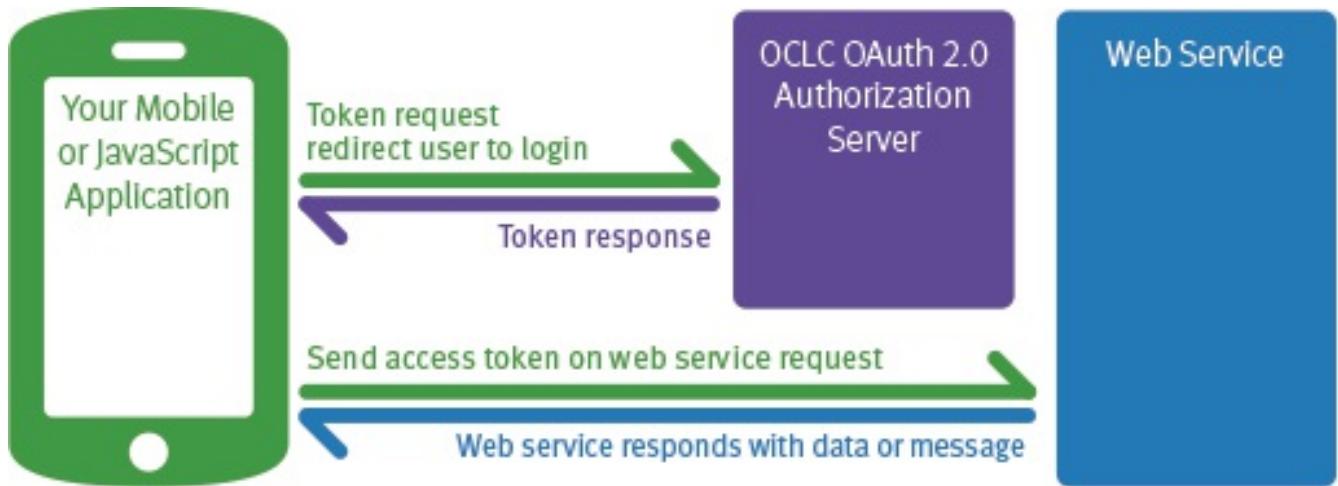
- **Les apps locales (ex : petits jeux)** : pas d'interaction entre le serveur de données et le client, elles sont stockées intégralement sur l'appareil de l'utilisateur. Les modifications de l'app (contenus, MAJ,...) se font par republication et MAJ par les mobinautes.
- **Les apps connectées au Cloud** : une connectivité complète avec le serveur de l'app, les utilisateurs, et les utilisateurs entre eux. C'est un véritable canal de communication qui vous permet d'interagir avec votre communauté, envoi de push, data-live etc.

Une solution de Mobile Backend est composée d'une plateforme, d'un hébergement, de datas et d'une application mobile. Pour pouvoir fournir la bonne information aux utilisateurs (géolocalisation, tri, filtre...), les applications ont donc besoin de recroiser leurs informations internes avec d'autres bases de données.

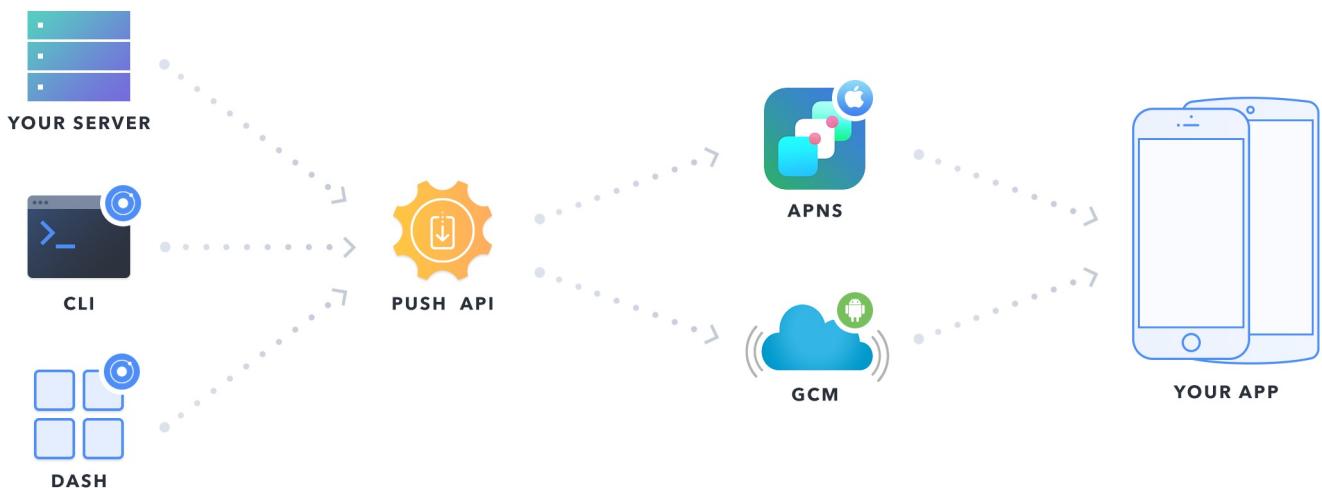
Pour répondre à ce besoin, les contenus de l'application mobile interagissent et sont mis à jour sur des serveurs « cloud ».

Une fois à connecté à votre application, le Mobile Backend vous permet de gérer la synchronisation des informations sur les différents supports (Web, Mobile, Facebook, Twitter...), gérer les campagnes de notifications et étudier leur ROI, ou encore obtenir les statistiques de l'utilisation des apps...

Les services d'authentification et de “push notification”.



Firebase propose des services d'authentification et de push.



Utilisation :

```
npm install firebase angularfire2 --save  
ionic generate provider Auth
```

[Suivre le guide](#)

Style d'architecture.

Un style d'architecture est un ensemble de contraintes qui permettent, lorsqu'elles sont appliquées aux composants d'une architecture, d'optimiser certains critères propres au cahier des charges du système à concevoir.

REST

REST a été décrit par Roy Thomas Fielding dans sa thèse « Architectural Styles and the Design of Network-based Software Architectures ». REST est l'acronyme de Representational State Transfer. C'est un style d'architecture.

L'information de base, dans une architecture REST, est appelée ressource.

Une ressource est identifiée par un identificateur de ressource. URI (Uniform Resource Identifier).

HTTP Verb	Path	Controller#Action	Used for
GET	/photos	photos#index	display a list of all photos
GET	/photos/new	photos#new	return an HTML form for creating a new photo
POST	/photos	photos#create	create a new photo
GET	/photos/:id	photos#show	display a specific photo
GET	/photos/:id/edit	photos#edit	return an HTML form for editing a photo
PATCH/PUT	/photos/:id	photos#update	update a specific photo
DELETE	/photos/:id	photos#destroy	delete a specific photo

Les composants de l'architecture manipulent ces ressources en transférant des représentations de ces ressources.

L'interface entre les composants est simple et uniforme. En HTTP, cette interface est implantée par les verbs GET, PUT, POST, DELETE, pour manipuler les ressources de manière simple et « intuitive »

REST Design Principles – URI / Resource



Rule1: Plural for collections, ID for single element

Rule2: No verbs! `./petstore/dogs/add`

Rule3: Query params only for meta-info (filtering, format)

brunelot

Une architecture REST doit respecter les six contraintes suivantes :

Client-serveur : les responsabilités sont séparées entre le client et le serveur. L'interface utilisateur est séparée de celle du stockage des données. Cela permet aux deux d'évoluer indépendamment.

À serveur sans état : chaque requête d'un client contient toute l'information nécessaire pour permettre au serveur de traiter la requête, sans dépendre d'un contexte conservé sur le serveur en relation avec des requêtes antérieures. **Il n'y a pas de session entre le client et le serveur.**

Avec cache : chaque réponse d'un serveur contient un marquage explicite ou implicite de la possibilité ou non de la mettre en cache côté client ou côté serveur. Si une réponse peut être mise en cache, elle peut être réutilisée pour des requêtes ultérieures équivalentes.

À interface uniforme :

- identification des ressources : chaque ressource possède un identifiant ;
- manipulation des ressources par des représentations : les ressources ont des représentations définies ;
- messages auto-descriptifs : les messages expliquent leur nature. Par exemple, si une représentation en HTML est encodée en UTF-8, le message contient l'information nécessaire pour dire que c'est le cas ;
- hypermédia comme moteur d'état de l'application : chaque accès aux états suivants de l'application est décrit dans le message courant.

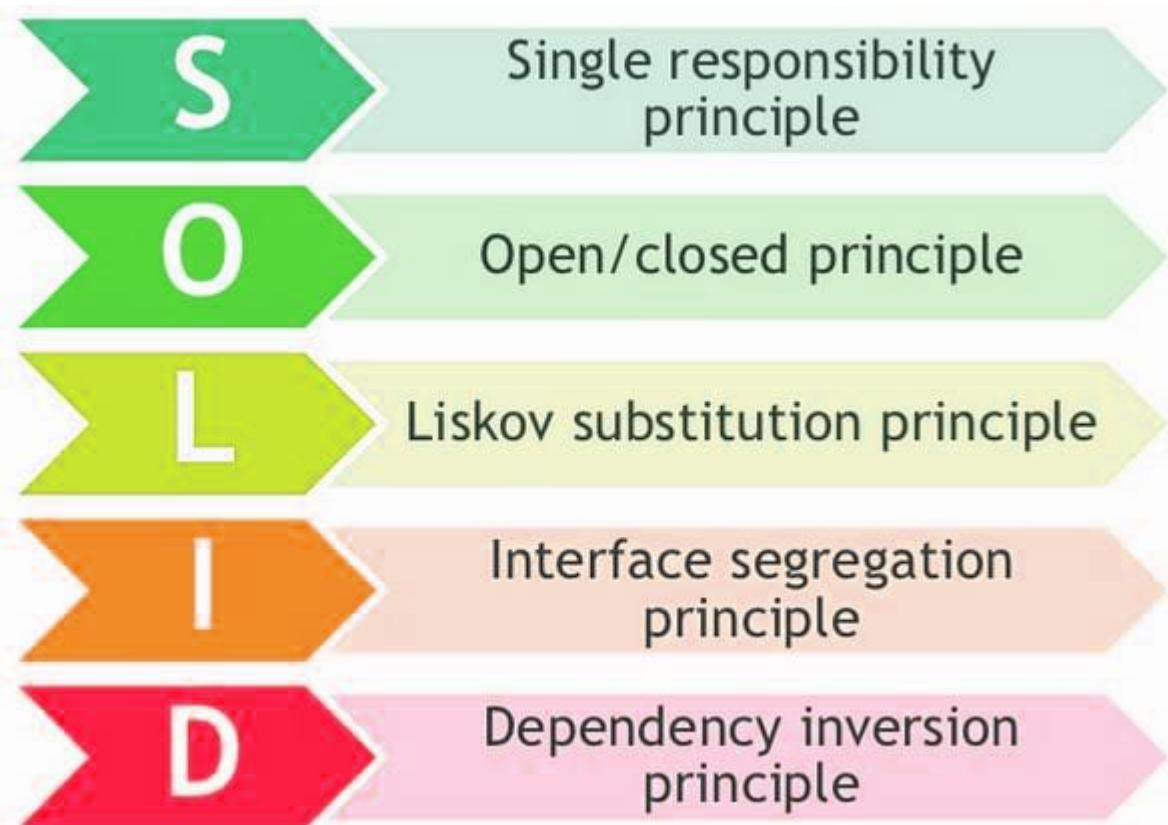
LDP : Linked Data Platform

Cette spécification décrit l'utilisation de HTTP pour accéder, mettre à jour, créer et supprimer des ressources provenant de serveurs qui exposent leurs ressources en tant que données liées. :

- URI comme noms pour les choses
- URI HTTP pour que les gens puissent consulter ces noms
- Lorsque quelqu'un recherche un URI, fournir des informations utiles, en utilisant les normes (RDF *, SPARQL)
- Inclure des liens vers d'autres URI, afin qu'ils puissent découvrir plus de choses

SOLID

SOLID est un acronyme représentant cinq principes de bases pour la programmation orientée objet, introduits par Michael Feathers et Robert C. Martin au début des années 2000.



(Single Responsibility Principle) Responsabilité unique

une classe, une fonction ou une méthode doit avoir une et une seule responsabilité.

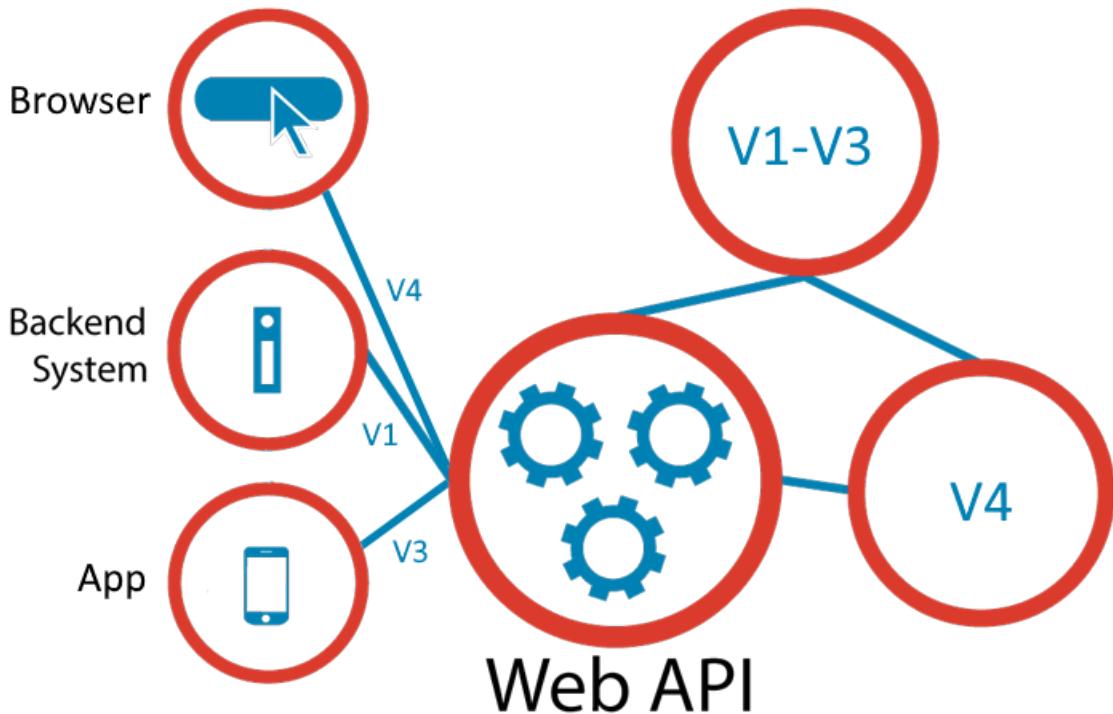
(Open/closed principle) Ouvert/fermé une classe doit être ouverte à l'extension, mais fermée à la modification.

(Liskov substitution Principle) Substitution de Liskov une instance de type T doit pouvoir être remplacée par une instance de type G, tel que G sous-type de T, sans que cela ne modifie la cohérence du programme.

(Interface segregation principle) Ségrégation des interfaces préférer plusieurs interfaces spécifiques pour chaque client plutôt qu'une seule interface générale.

(Dependency Inversion Principle) Inversion des dépendances

Versioning



Toute API sera amenée à évoluer dans le temps. Une API peut être versionnée de différentes manières:

- Par timestamp, par numéro de version,...
- Dans le path, au début ou à la fin de l'URI
- En paramètre de la request
- Dans un Header HTTP
- Avec un versioning facultatif ou obligatoire

La version d'une API est une information essentielle. Faire figurer un numéro de version obligatoire, sur un digit, au plus haut niveau du path de URI.

API Design

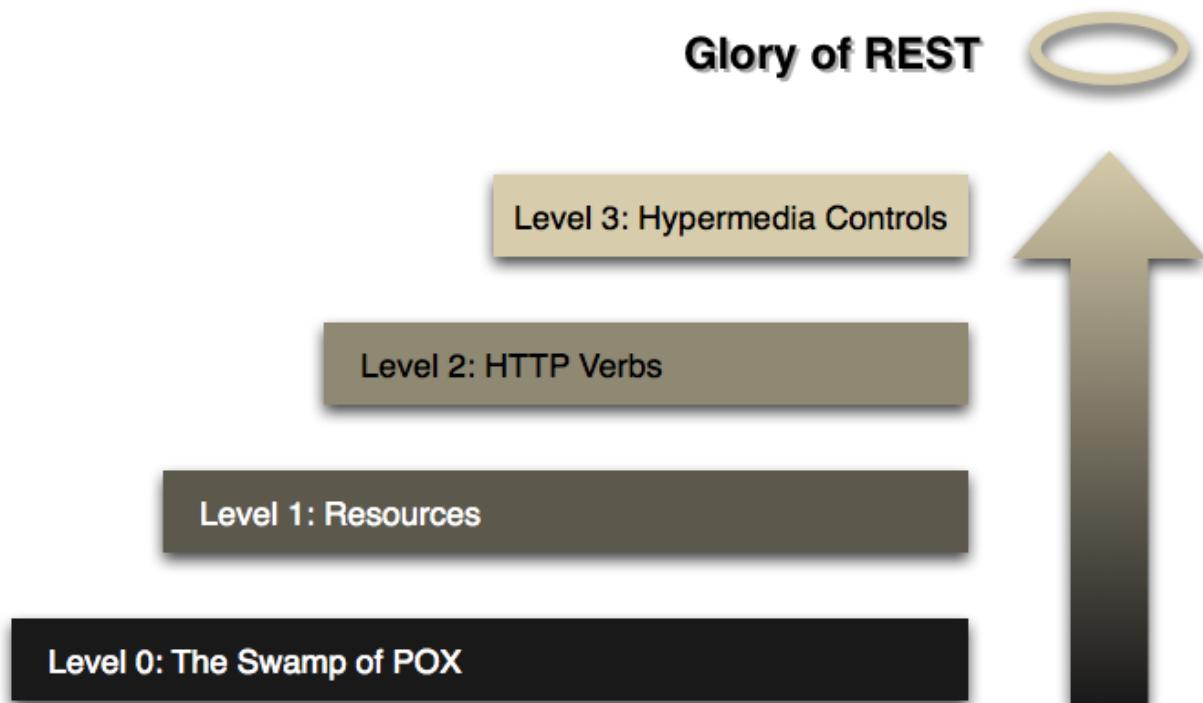
KISS – « Keep it simple, stupid »

Il est primordial que l'API soit la plus simple possible et auto-descriptive, de manière à ce que l'utilisateur ait à consulter la documentation le moins possible.

- **Termes usuels et concrets** : clients, orders, addresses, products,...
- **Ne pas proposer aux utilisateurs de pouvoir faire quelque chose de plusieurs manières différentes.**
- L'API est "designée" pour les clients : les développeurs, et non pour les "data".

<https://developer.github.com/v3/>
<https://developers.google.com/youtube/v3/live/authentication>
<https://developer.paypal.com/docs/api/>
<https://developers.facebook.com/docs/graph-api/>
<http://instagram.com/developer/endpoints/likes/>

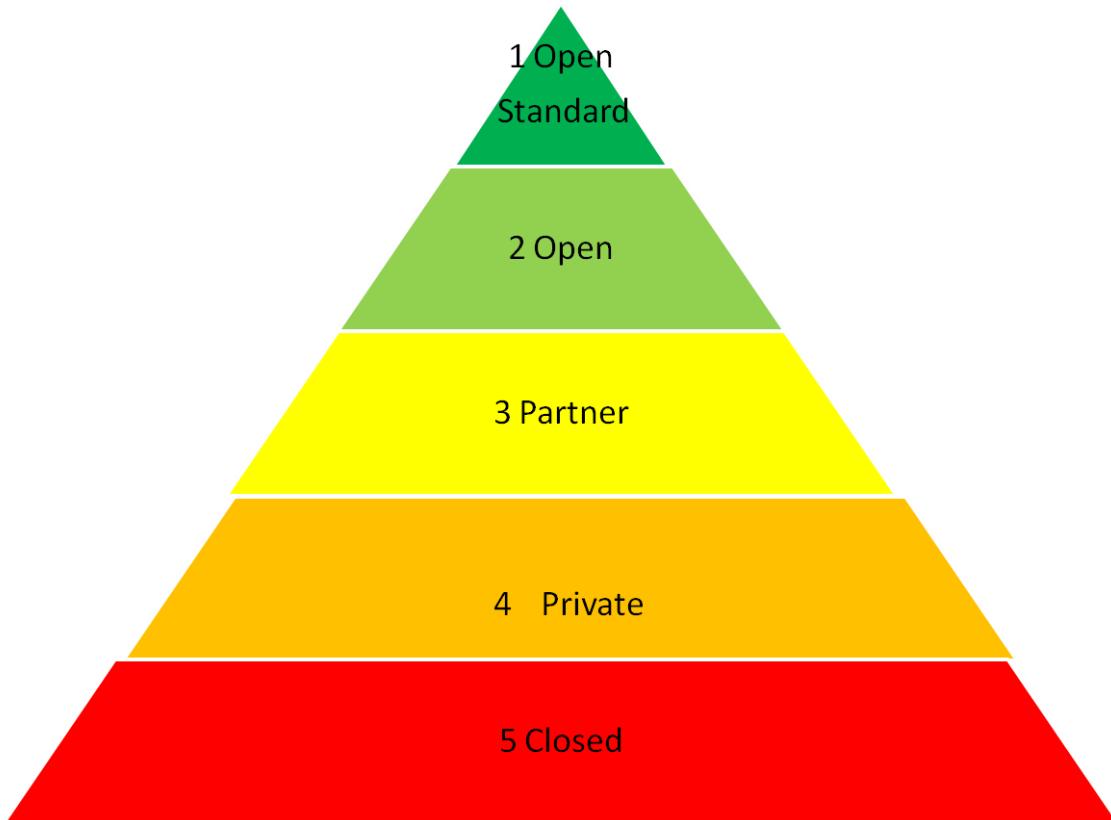
Niveaux d'ouverture d'une API



En marge du modèle de maturité REST de Richardson, distinguons trois niveaux d'ouverture d'une API :

- **Niveau 1 « Private API »** : l'API est destinée à être consommée par les applications développées au sein de l'entreprise.
- **Niveau 2 « Partners API »** : l'API est destinée à être consommée par les applications développées au sein de l'entreprise ou par des partenaires.

- **Niveau 3 « Open API »** : l'API est destinée à être consommée par tout type de développeurs.



La différence entre chaque niveau réside principalement dans la qualité du processus d'industrialisation de consommation des services qu'il convient de mettre en œuvre :

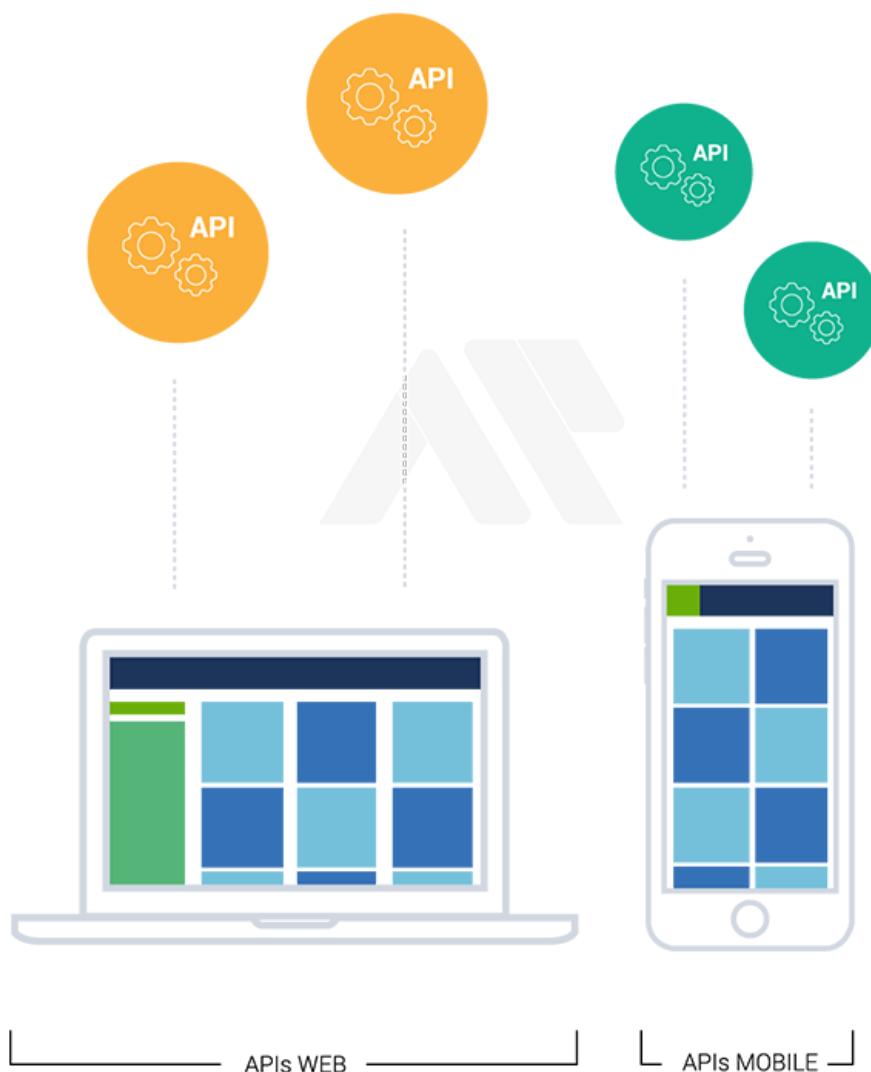
- niveau 1, un développeur d'application est contraint de consommer l'API et peut directement contacter l'équipe API.
- niveau 3, le succès d'une API dépend de la facilité du processus d'enrôlement et de sa documentation.

(Liskov substitution Principle) Substitution de Liskov une instance de type T doit pouvoir être remplacée par une instance de type G, tel que G sous-type de T, sans que cela ne modifie la cohérence du programme.

(Interface segregation principle) Ségrégation des interfaces préférer plusieurs interfaces spécifiques pour chaque client plutôt qu'une seule interface générale.

(Dependency Inversion Principle) Inversion des dépendances

API Mobiles.



Les API mobiles sont basées sur l'utilisation de la technologie WOA. La création de ces services est devenue plus simple en utilisant des protocoles Web simplifiés, par ex. REST et JSON (JavaScript Object Notation).

Ces protocoles sont beaucoup plus faciles pour les développeurs web, car ils nécessitent moins de CPU et de bande passante. Ils sont plus reconnus à cause des grandes plates-formes sociales, telles que **Facebook, Amazon et Twitter**, etc.

Ajax : consommation de services Web.

AJAX est l'acronyme d'Asynchronous JavaScript And XML, autrement dit JavaScript Et XML Asynchrones.

AJAX n'est ni une technologie ni un langage de programmation ; AJAX est un concept de programmation Web reposant sur plusieurs technologies comme le JavaScript et le XML - d'où le nom AJAX.

L'idée même d'AJAX est de faire communiquer une page Web avec un serveur Web sans occasionner le rechargement de la page. C'est la raison pour laquelle JavaScript est utilisé, car c'est lui qui va se charger d'établir la connexion entre la page Web et le serveur.

```
// Requête réseau en JavaScript

// Construire l'objet de requête
let xhr = new XMLHttpRequest();
// Paramétriser la requête
xhr.open('GET', 'http://jsonplaceholder.typicode.com/users');
// Déclencher la requête
xhr.send();
// Traiter le résultat à la fin de la requête
xhr.onload = function(){
    // Transformer les données JSON
    var result = JSON.parse(xhr.responseText);
    console.log(result)
}
```

Choisir et paramétrer un “workflow” mobile.



Un “workflow” ou *cyle de travail* est l'ensemble des outils choisis pour assuré une tâche.

“workflow” mobile.

1. Idée ou demande client.
2. Proposition.
3. Prototype.
4. Développement.
5. Livraison.

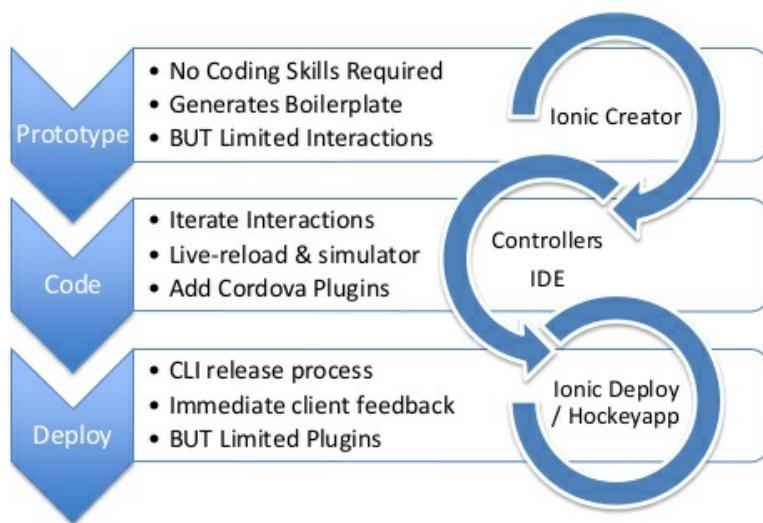
Ionic propose avant tout un workflow mobile en réunissant un ensemble d'outils dans l'**ionic-cli**.

cli désigne **command line interface** c'est à dire un logiciel utiliser en ligne de commande.

Analysons le “workflow” ionic

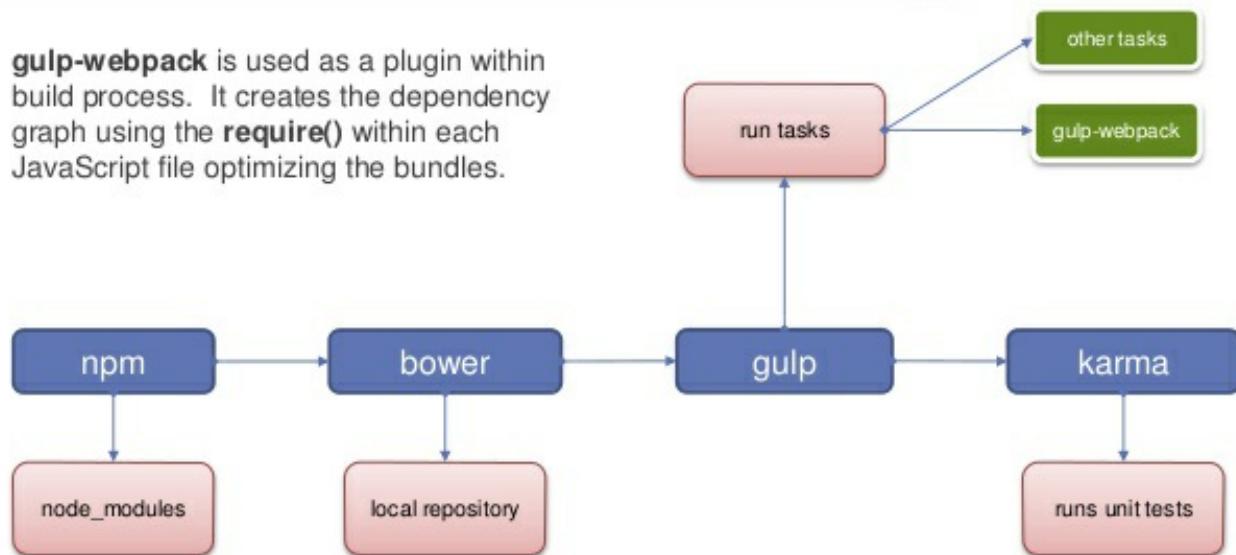
1. Prototyper l'application avec **ionic-creator**.
2. Développer avec **ionic-cli**.
3. Déployer avec **ionic-push**.

Our Workflow with Ionic



Webpack workflow

gulp-webpack is used as a plugin within build process. It creates the dependency graph using the **require()** within each JavaScript file optimizing the bundles.



WebPack. est un **module bundler**, un utilitaire manipulant les ressources afin de les préparer pour leur distribution. L'ensemble des transformations sont ainsi réalisées en tâche de fond et immédiatement répercutées dans le navigateur du développeur.

JavaScript ES6/ES2015 avec TypeScript .

TypeScript est un **superset** d'Ecma Script 6/+.

Utilisé conjointement avec des **polyfills** il permet de tirer parti des évolutions du langages.

L'apprentissage d' **TypeScript** passe par celui de **JavaScript**.

TypeScript étant. lui même un **superset ES6/2015**



ES6/2015 : Evolutions syntaxiques fondamentales.

Transpilers

Avec l'évolution rapide des caractéristiques, un problème se pose pour les développeurs JavaScript qui désirent utiliser les nouvelles fonctionnalités tout en maintenant ses applications pour les anciens navigateurs.

Comment anticiper le support natif ?

La réponse est l'outillage, en particulier une technique appelée **transpiling** (transformation + compilation).

TypeScript est un enrichissement de **JavaScrit** ET un **Tranpisler**.

Il convertit le code **.ts** en **.js**

Les transpilers assure cette transformation comme part du **workflow** de la même manière que les linter ou les opérations de minification.

L'idée est d'utiliser un outil pour transformer votre code ES6 à un équivalent (ou presque!) ES5.

Promise easy basics creation chaining `then()` new the API	Array Array.from() Array.of() [].fill() [].find() [].findIndex() [].entries() [].keys() [].values()	Class easy creation accessors easy static easy extends more extends super in method super in constructor	Destructuring easy array easy string easy object easy defaults parameters assign	Generator creation iterator yield expressions send value to a generator send function to a generator `return` inside a generator function	Map easy Basics map.get() map.set() initialize easy map.has()
Reflect easy Basics Reflect.apply() Reflect.getPrototypeOf() Reflect.construct() Reflect.defineProperty()	Set easy basics set.add() easy set.delete() easy the API easy set.clear()	Iterator array string protocol usage	Object literal basics computed properties easy getter easy setter	String easy string.includes() easy string.repeat(count) easy string.startsWith() easy string.endsWith()	
Template strings easy basics easy multiline tagged template strings 'raw' property	Symbol basics Symbol.for() Symbol.keyFor()	Arrow functions easy basics easy function binding	Block scope easy 'let' declaration easy 'const' declaration	Rest operator as parameter with destructuring	Spread operator with arrays with strings
Default parameters easy Basics	Modules easy 'import' statement	Number easy Number.isInteger()	Object easy Object.is()	Unicode in strings	

Constantes et variables de bloc.

Variables de bloc.

L'instruction **let** permet de déclarer des variables dont la portée est limitée à celle du bloc dans lequel elles sont déclarées.

Au niveau le plus haut (la portée globale), let crée une variable globale alors que var ajoute une propriété à l'objet.

```
var x = 'global';
let y = 'global2';
console.log(this.x); // "global"
console.log(this.y); // undefined
console.log(y);     // "global2"
```

Rappel : Le mot-clé **var** permet de définir une variable globale ou locale à une fonction (sans distinction des blocs utilisés dans la fonction).

```
for (let i = 1; i <= 5; i++) {
  setTimeout(function(){console.info(i)},1000)
}

for (var i = 1; i <= 5; i++) {
  setTimeout(function(){console.warn(i)},1000)
}
```

Redéclarer une même variable au sein d'une même portée de bloc entraîne une exception **TypeError**Contrairement au corps de fonction.

```
if (x) {  
    let myVar;  
    let myVar; // TypeError  
}  
  
function foo() {  
    let myVar;  
    let myVar; // Cela fonctionne.  
}
```

Faire référence à une variable dans un bloc avant la déclaration de celle-ci avec **let** entraînera une exception **ReferenceError**. La variable est placée dans une « zone morte temporaire » entre le début du bloc et le moment où la déclaration est traitée.

Zone morte temporaire (temporal dead zone / TDZ) concerne les erreurs liées à **let**

```
function foo() {  
    console.log(myVar); // ReferenceError  
    let myVar = true;  
}
```

Constantes

La déclaration **const** permet de déclarer un identifiant constant.

Attention: Cela ne signifie pas que la valeur contenue est immuable, uniquement que l'identifiant ne peut pas être réaffecté.

Les constantes font partie de la portée du bloc (comme les variables définies avec **let**).

```
const myVar = true;
myVar = false;

const myObj = {};
myObj.property = false;
myObj = {};
```

- Il est nécessaire d'initialiser une constante lors de sa déclaration.
- Il est impossible d'avoir une constante qui partage le même nom qu'une variable ou qu'une fonction.(Au sein d'une même portée)

Usage: On préférera **let** pour les variables et **const** pour les fonctions.

```
const foo = function foo() {
  let myVar = true;
  console.log(myVar);
};
```

Fonction, paramètres par défaut.

Paramètres par défaut.

En JavaScript, les paramètres de fonction non renseignés valent **undefined**. EN ES6 il est possible de définir une valeur par défaut différente.

```
//ES5
function multiplier(a, b) {
    b = typeof b !== 'undefined' ? b : 1;

    return a*b;
}

multiplier(5); // 5
```

Code allégé avec ES6

```
function multiplier(a, b = 1) {
    return a*b;
}

multiplier(5); // 5
```

"Arrow Function" : portée lexicale. Usages.

Les **Arrows Function** sont un raccourci syntaxique pour la création de fonction utilisant le symbole `=>`.

Les fonctions ainsi créées sont syntaxiquement similaire à la fonction en C#, Java 8 et CoffeeScript.

Elles définissent le corps (bloc de code) et la valeur de retour.

Contrairement aux fonctions classiques, les **Arrows function** partagent la même valeur lexicale pour le mot clé `this` que leur code environnant.

```
var odds = evens.map(v => v + 1);
var nums = evens.map((v, i) => v + i);
var pairs = evens.map(v => ({even: v, odd: v + 1}));
```

```
nums.forEach(v => {
  if (v % 5 === 0)
    fives.push(v);
});
```

```
var bob = {
  _name: "Bob",
  _friends: [],
  printFriends() {
    this._friends.forEach(f =>
      console.log(this._name + " knows " + f));
  }
}
```

Une expression de fonction fléchée permet d'utiliser une syntaxe plus concise que les expressions de fonctions classiques. La valeur `this` est alors **liée lexicalement**. Les fonctions fléchées sont nécessairement anonymes.

Les **Arrows** sont un raccourci pour la création de fonction en utilisant le => . Elles définissent le corps (bloc de code) et la valeur de retour.

Modèles de classe et héritage.

Les classes ont été introduites dans JavaScript avec ECMAScript 6 et sont un **sucré syntaxique** de l'héritage prototypal. Le mot clé **class** fournit une syntaxe plus claire pour utiliser un modèle et gérer l'héritage.

Cette syntaxe n'introduit pas un nouveau modèle d'héritage dans JavaScript ! ** JavaScript est un langage à paradigme multiple : Orienté Objet (prototypal), impératif et fonctionnel. **ES6 n'introduit pas de paradigme Orienté Objet basé sur les classes

Les classes sont des *fonctions spéciales* de la même façon qu'il y a des expressions de fonctions et des déclarations de fonctions, on aura deux syntaxes :

Déclarations de classes

```
class Polygone {  
    constructor(hauteur, largeur) {  
        this.hauteur = hauteur;  
        this.largeur = largeur;  
    }  
}
```

Expressions de classes

Si on utilise un nom dans l'expression, ce nom ne sera accessible que depuis le corps de la classe.

```
// anonyme
var Polygone = class {
    constructor(hauteur, largeur) {
        this.hauteur = hauteur;
        this.largeur = largeur;
    }
};

// nommée
var Polygone = class Polygone {
    constructor(hauteur, largeur) {
        this.hauteur = hauteur;
        this.largeur = largeur;
    }
};
```

Héritage de classes

Le mot-clé **extends**, utilisé dans les déclarations ou les expressions de classes, permet de créer une classe qui hérite d'une autre classe (on parle aussi de « sous-classe » ou de « classe-fille »).

Le mot-clé **super** fait référence de la classe parente.

```
class Animal {
    constructor(nom) {
        this.nom = nom;
    }
}
```

```
parle() {
    console.log(this.nom + ' fait du bruit.');
}
}

class Chien extends Animal {
parle() {
    super.parle();
    console.log(this.nom + ' aboie.');
}
}
```

Une classe ECMAScript ne peut avoir qu'une seule classe parente.

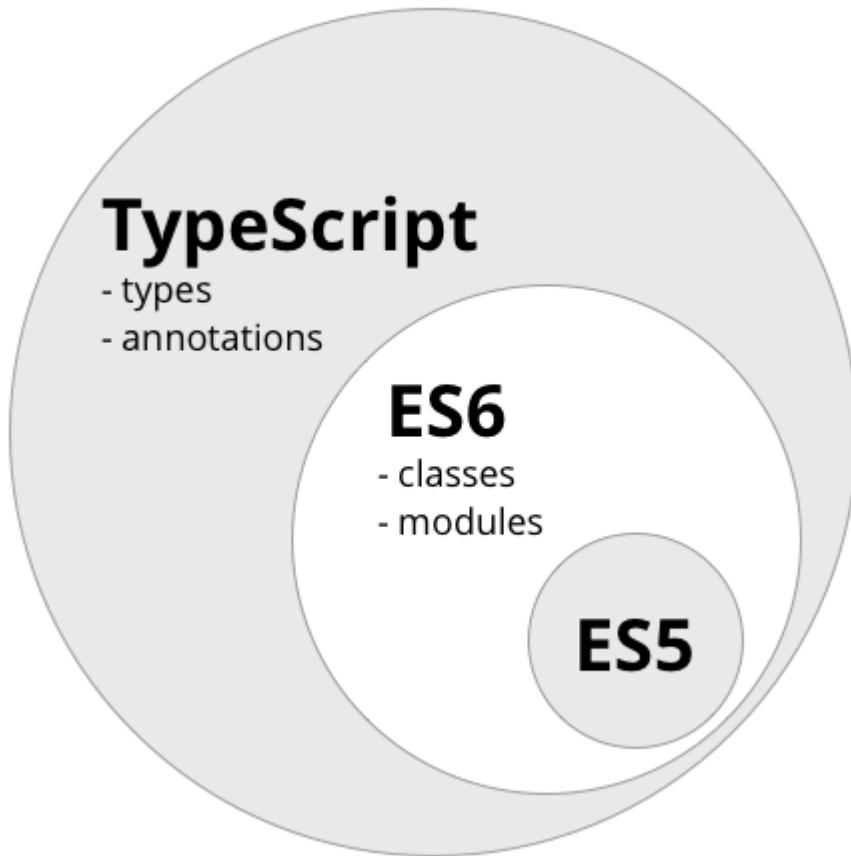
Créer une classe statique

```
class Manager {  
    constructor() {  
        return Manager;  
    }  
  
    static create(value) {  
        Manager.collection.push(value);  
    }  
  
    static remove(value) {  
  
        Manager.collection.splice(Manager.collection.indexOf(  
            value), 1);  
    }  
}  
Manager.collection = []  
  
Manager.create();
```

TypeScript en détail, configuration.

TypeScript est un langage facilitant le développement d'applications larges et "scalables", écrites en JavaScript.

TypeScript apporte un [support supérieur à 58%](#) des nouveautés syntaxique ES6/2015



Le **transpiler** TypeScript permet l'ajout de concepts *propre* à la programmation orientée objet par **classe** tels que :

- Les interfaces
- Les génériques
- Le typage statique
- Décorateurs

C'est une surcouche de JavaScript : tout le code JavaScript est valide en TypeScript ce qui permet de l'ajouter de façon transparente à n'importe quel projet.

Parceque le code TypeScript est *transcomplié* en JavaScript en tant que language cible on le qualifie de [Superset JavaScript](#).

Le [Playground](#) permet de tester la syntaxe TypeScript en illustrant le mécanisme de transpilation.

Installation et mise en oeuvre.

Les deux principaux mode d'installation de TypeScript :

1. En autonomie sur le socle NodeJS via npm
2. Par [plugin selon votre éditeur](#)

Installation avec NPM

```
npm install -g typescript
```

Premier Script

A noter les fichier typescript utilisent l'extension `.ts`

```
// user.ts

class User {
    userName:string;

    constructor(name:string){
        this.userName = name;
    }
}

let currentUser = new User('Linus');
```

Créer un fichier de configuration

```
tsc --init
```

La commande d'initialisation permet de générer simplement le fichier `tsconfig.json`

```
{
  "compilerOptions": {
    "module": "commonjs",
    "target": "es5",
    "noImplicitAny": false,
    "sourceMap": false
  }
}
```

A noter : Pour un projet JavaScript on préférera les options suivante.

```
rm tsconfig.json & tsc --init --moduleResolution node --target ES5 --sourceMap --modu
```

Apports Syntaxiques

L'usage de TypeScript renforce la syntaxe JavaScript :

- Support (partiel) de la syntaxe ES6/2015
- Typage fort (variable, fonction, types génériques)
- Enrichissement des “patterns OO” (class, interface, héritage)
- Modularité (namespace)

Grâce à ces enrichissement et sa phase de compilation TypeScript offre une plus grande prédictibilité du code.

Typage des variables.

Declaration Type	Namespace	Type	Value
Namespace	X		X
Class		X	X
Enum		X	X
Interface		X	
Type Alias		X	
Function			X
Variable			X

TypeScript propose des types basiques :

- Boolean
- Number
- String
- Array
- Tuple
- Enum
- Any
- Void
- Null et Undefined
- Never

Les 3 premiers sont des représentations standards JavaScript.

```
//boolean
let complete: boolean = false;
//number
let decimal: number = 6;
let hex: number = 0xf00d;
let binary: number = 0b1010;
let octal: number = 0o744;
//string
let color: string = "blue";
```

Array

Les tableaux peuvent être annoté de deux façons :

- Type des éléments suivis de []
- Notation générique **Array<elemType>**

```
//array
let list: number[] = [1, 2, 3];
let list: Array<number> = [1, 2, 3];
```

Tuple

Les **tuple** sont l'expression d'un **array** dont le type des éléments est connu bien potentiellement différents.

```
// Declare un tuple
let x: [string, number];
// Initialisation
x = ["hello", 10]; // OK
// Error d'initialisation
x = [10, "hello"]; // Error
```

Enum

Les **enum** ou un type énuméré est un type de données qui consiste en un ensemble de constantes indiquées.

```
enum Color {Red, Green, Blue};  
let c: Color = Color.Green;
```

Any et Void, Null et Undefined et Never

any tous les types acceptés.

```
let notSure: any;  
notSure = "maybe a string instead";  
notSure = false; // okay, definitely a boolean  
  
let list: any[] = [1, true, "free"];  
  
list[1] = 100;
```

Les types **any** et **void** apportent un comportement opposé.

void aucun type accepté.

```
function warnUser(): void {  
    alert("This is my warning message");  
}
```

Null et **Undefined** représentation peu utiles.

never particulièrement utilisé pour les fonctions proposant une erreur ou sans valeur de retour.

```
// Function returning never must have unreachable end point
function error(message: string): never {
    throw new Error(message);
}

// Inferred return type is never
function fail() {
    return error("Something failed");
}
```

Inférence de type.

L'[inférence de types](#) est un mécanisme qui permet à un compilateur de rechercher automatiquement les types associés à des expressions, sans qu'ils soient indiqués explicitement dans le code source.

Il s'agit pour le compilateur ou l'interpréteur de trouver le type le plus général que puisse prendre l'expression.

```
let x = 3; // Type Inferred to number
x = ''; // Error : Type 'string' is not assignable to type 'number'.
```

Best common type lorsque l'inférence repose sur plusieurs types la représentation la plus commune est recherchée.

```
let x = [0, 1, null];
x.push('');// Error : Argument of type 'string' is not assignable to parameter of type 'number'

let zoo = [new Rhino(), new Elephant(), new Snake()];
let zoo: Animal[] = [new Rhino(), new Elephant(), new Snake()];
```

Si aucun type ne peut être déterminé l'inférence sera résolue sur `object {}`

Erreur de contexte :

```
window.onmousedown = function(mouseEvent) {
    console.log(mouseEvent.button); //<- Error
};

//versus

window.onmousedown = function(mouseEvent: any) {
    console.log(mouseEvent.button); //<- Now, no error is given
};
```

Assertion de type.

L'assertion de type permet de renseigner le compilateur selon deux syntaxes.

```
let someValue: any = "this is a string";  
  
let strLength: number = (<string>someValue).length;
```

Ou préférablement :

```
let someValue: any = "this is a string";  
  
let strLength: number = (someValue as string).length;
```

Typage des fonction.

Les [définitions de fonctions](#) reçoivent également des information de types sur :

- Les arguments.
- Les arguments optionnels.
- Les valeurs de paramètres par défaut.
- Les valeurs de retour.

```
function add(x: number, y?: number = 1 ): number {  
    return x + y;  
}
```

Surcharge de fonction.

Puisqu 'il est possible d'exprimer un typage fort dans les signature de fonction, on peut utiliser cette différentiation pour **surcharger la définition de fonction**

```
let suits = ["hearts", "spades", "clubs", "diamonds"];  
  
function pickCard(x: {suit: string; card: number;}[]): number;  
function pickCard(x: number): {suit: string; card: number;};  
function pickCard(x): any {  
  
    if (typeof x == "object") return Math.floor(Math.random() * x.length);  
  
    if (typeof x == "number") return { suit: suits[Math.floor(x / 13)], card: x % 13 }  
}  
  
let myDeck = [{ suit: "diamonds", card: 2 }, { suit: "spades", card: 10 }, { suit: "he  
let pickedCard1 = myDeck[pickCard(myDeck)];  
alert("card: " + pickedCard1.card + " of " + pickedCard1.suit);  
  
let pickedCard2 = pickCard(15);  
alert("card: " + pickedCard2.card + " of " + pickedCard2.suit);
```

Patterns de programmation orientée objet.

Enrichissement des `class`.

TypeScript enrichie la syntaxe **ES6/2015**

- Interface
- Déclaration de propriétés

Interface.

En programmation orientée objet, une interface décrit à minima les méthodes accessibles depuis l'extérieur de la classe, par lesquelles on peut modifier l'objet.

Interfaces Typecript en détail

il est commun à toutes les Interfaces de déclarer chacune de ses méthodes sous la forme d'une signature :

```
nomDeFonction (argument1:Type, ...):typeRetour;
```

Pour rappel la différenciation **publique/privée** ou portée de variable permet :

- d'éviter de manipuler l'objet de façon indue, en limitant ses modifications à celles autorisées comme publiques par le concepteur de la classe
- à ce concepteur, de modifier l'implémentation interne de ces méthodes de manière transparente.

De part le typage structurel dynamique ou "["Duck Typing"](#)" de TypeScript, les interfaces peuvent également **décrire la structure d'un objet en incluant les propriétés.**

```
interface User {  
    connect(credential: string): boolean;  
    name: string;  
    email: string;  
    score: number;  
}
```

Bien qu'il soit parfois d'usage de préfixer une interface d'un I majuscule cet usage n'est pas recommandé. (voir Best Practices)

Une fonction pourra par exemple utiliser l'interface pour vérifié la structure de l'objet à manipuler :

```
function increaseUserScore(user:User):boolean{  
    if(!user.connect('12345678')) return false;  
    user.score += 1;  
}
```

"Duck Typing" Le type dans le contexte où il est utilisé, est déterminée par l'ensemble de ses méthodes et de ses attributs.

Il est possible de décrire la structure attendue dans la signature de fonction :

```
function increaseUserScore(user:{  
    connect(credential: string): boolean;  
    name: string;  
    email: string;  
    score:number;  
} ):boolean{  
    if(!user.connect('12345678')) return false;  
    user.score += 1;  
}
```

Avec la même notation que les paramètres de fonction certains membres peuvent être rendus optionnels

```
interface User {  
    connect(credential: string): boolean;  
    disconnect?(): boolean; //Optionnel  
    name: string;  
    email?: string; //Optionnel  
    score:number;  
}
```

readonly identifie les propriétés modifiables uniquement par le constructeur.

TypeScript implémente les **ReadonlyArray<T>** (Array sans mutateurs)

```
interface User {  
    connect(credential: string): boolean;  
    readonly name: string;  
    readonly email?: string; //Optionnel  
    score:number;  
}
```

Utiliser **const** pour les variables et **readonly** pour les propriétés.s

Les sont **interface** utilisables pour les `casting` :

```
let me = {connect(credential){return true;},name:'Me',score:0} as User;
```

Les interfaces peuvent également décrire des type de fonction ou de collections.

```
interface User {
  readonly name: string;
  readonly email?: string; //Optionnel
  score:number;
}

class Player implements User{}
```

Class.

Les classes peuvent implémenter une ou plusieurs interface(s).

A noter une class peut également servir d'interface.

```
interface User {
  name: string;
  email?: string; //Optionnel
  score:number;
}

interface AppUser {
  ...
}

class AdminUser implements User, AppUser {
  ...
}
```

Héritage.

Une classe ou une interface peuvent en **étendre** d'autre(s).

```
interface Shape {  
    color: string;  
}  
  
interface PenStroke {  
    penWidth: number;  
}  
  
interface Square extends Shape, PenStroke {  
    sideLength: number;  
}
```

On notera qu'une interface peut également étendre une classe.

```
class Control {  
    private state: any;  
}  
  
interface SelectableControl extends Control {  
    select(): void;  
}  
  
class Button extends Control {  
    select() {}  
}
```

TypeScript : le mécanisme des annotations.

```
// Décoration par annotation
@decoratorExpression
class MyClass { }
```

La fonction de décoration serait :

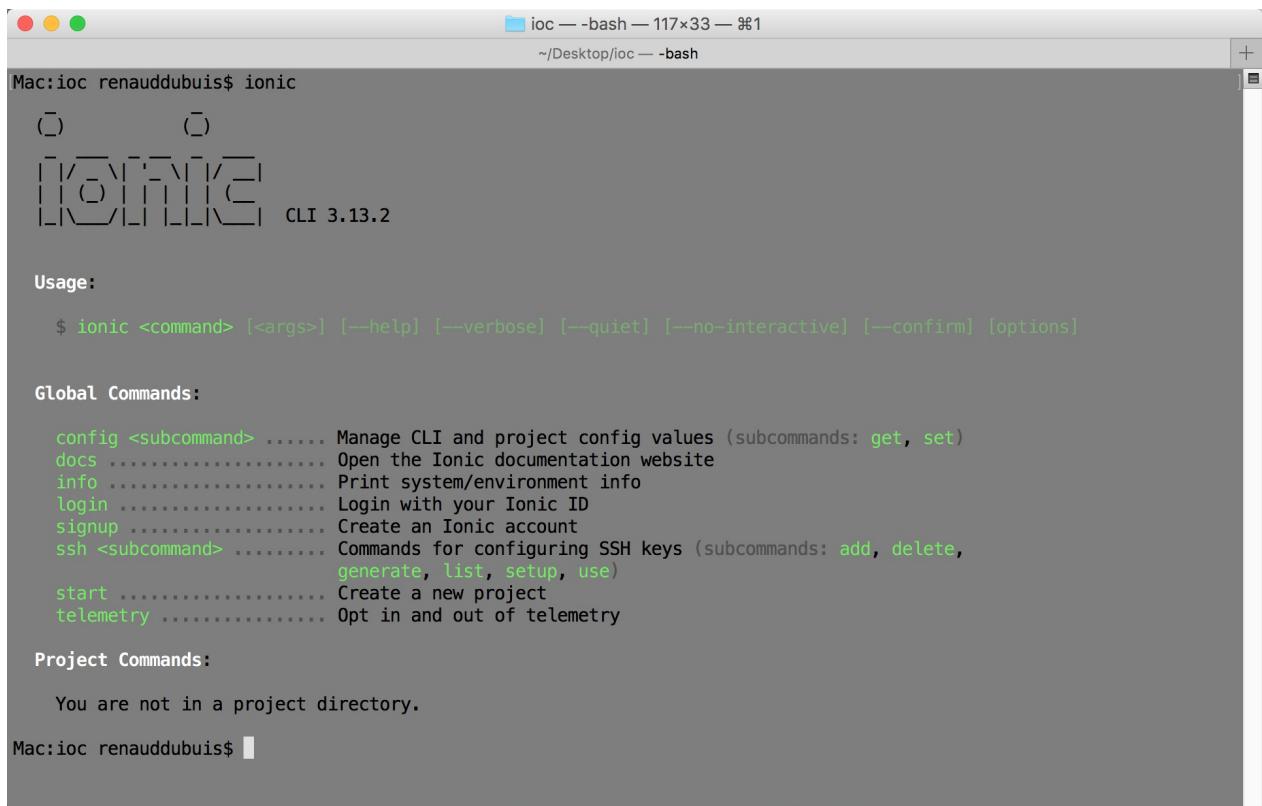
```
function decoratorExpression(target) {
    // Add a property on target
    target.annotated = true;
}
```

Le résultat est proche du mécanisme d'annotations ES5.

```
MainCtrl.annotations=['dep1','dep2']
function MainCtrl(dep1,dep2) {
}
```

Ionic CLI : Utiliser le “scaffoldeur” de projet.

Dans une invite de commande saisir **npx onic start**



```
Mac:ioc renaudubuis$ ionic
Ionic CLI 3.13.2

Usage:
$ ionic <command> [<args>] [--help] [--verbose] [--quiet] [--no-interactive] [--confirm] [options]

Global Commands:
config <subcommand> ..... Manage CLI and project config values (subcommands: get, set)
docs ..... Open the Ionic documentation website
info ..... Print system/environment info
login ..... Login with your Ionic ID
signup ..... Create an Ionic account
ssh <subcommand> ..... Commands for configuring SSH keys (subcommands: add, delete,
                      generate, list, setup, use)
start ..... Create a new project
telemetry ..... Opt in and out of telemetry

Project Commands:
You are not in a project directory.

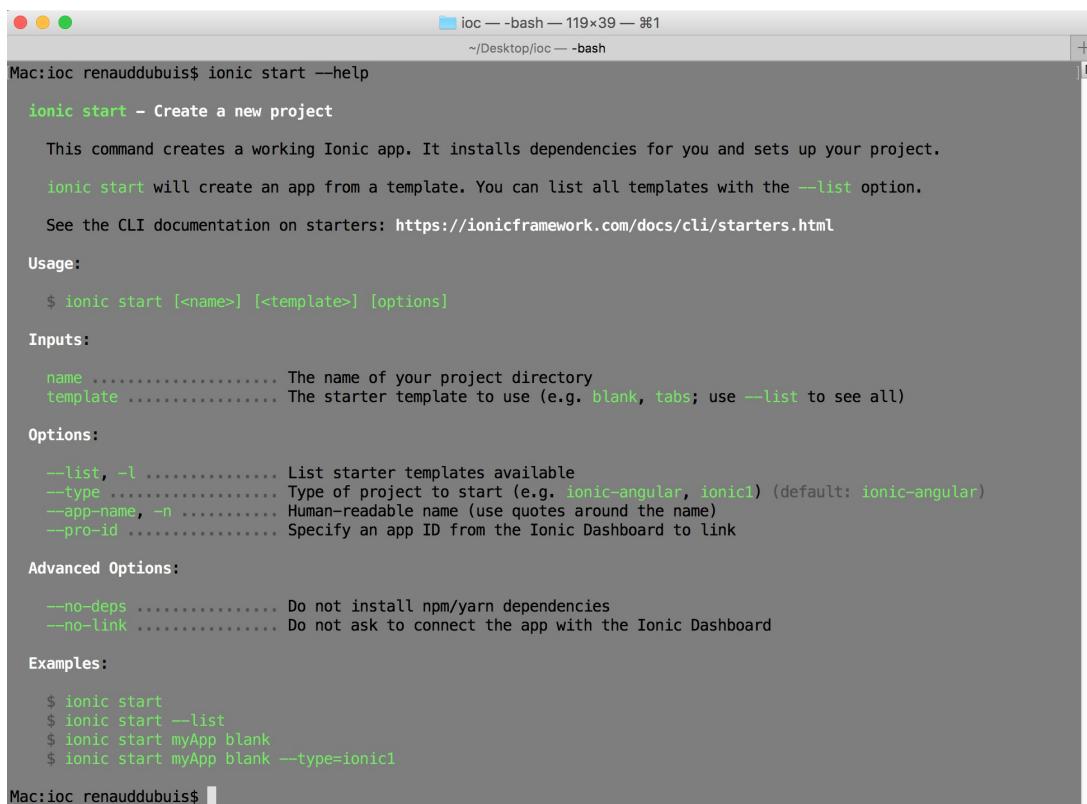
Mac:ioc renaudubuis$
```

```
config <subcommand> ..... Manage CLI and project
config values (subcommands: get, set)
docs ..... Open the Ionic
documentation website
info ..... Print
system/environment info
login ..... Login with your Ionic
ID
signup ..... Create an Ionic
account
ssh <subcommand> ..... Commands for
configuring SSH keys (subcommands: add, delete,
                      generate, list, setup,
use)
```

start Create a new project
telemetry Opt *in* and out of
telemetry

Activer l'aide sur une commande

Saisir ionic COMMAND --help comme par exemple ionic start --help



```
Mac:ioc renauddubuis$ ionic start --help
ionic start - Create a new project

This command creates a working Ionic app. It installs dependencies for you and sets up your project.

ionic start will create an app from a template. You can list all templates with the --list option.

See the CLI documentation on starters: https://ionicframework.com/docs/cli/starters.html

Usage:
$ ionic start [<name>] [<template>] [options]

Inputs:
name ..... The name of your project directory
template ..... The starter template to use (e.g. blank, tabs; use --list to see all)

Options:
--list, -l ..... List starter templates available
--type ..... Type of project to start (e.g. ionic-angular, ionic1) (default: ionic-angular)
--app-name, -n ..... Human-readable name (use quotes around the name)
--pro-id ..... Specify an app ID from the Ionic Dashboard to link

Advanced Options:
--no-deps ..... Do not install npm/yarn dependencies
--no-link ..... Do not ask to connect the app with the Ionic Dashboard

Examples:
$ ionic start
$ ionic start --list
$ ionic start myApp blank
$ ionic start myApp blank --type=ionic1
```

Démarrer un nouveau projet

Saisir **ionic start PROJECT_NAME TEMPLATE_NAME** ou :

- **PROJECT_NAME** est le nom du répertoire et de l'application.
- **TEMPLATE_NAME** est le nom du modèle que vous souhaitez scaffolder.

A noter pour connaître la liste des templates il est possible de saisir la commande : **ionic start --list**



Ionic SDK : présentation et mise en œuvre.

Ionic SDK : présentation et mise en oeuvre.



Introduction

Everything you need to know to get started with Ionic.



UI Components

A comprehensive preview of our mobile UI components - everything you'll need.



API

Explore our API for info on component methods, properties, and events.



Ionic Native

Integrate native device plugins, like Bluetooth, Maps, HealthKit, and more.



Theming

Learn how to easily customize and modify your app's design to fit your brand.



Ionicons

Over 900 custom-designed font/SVG icons. MIT licensed & ready to use with Ionic.



CLI

All about the primary tool used during the process of developing an Ionic app.



FAQ

Quickly find answers to some of the most commonly asked Ionic questions.



Forum

Drop by and say Hi. Ask a question or share something interesting.

ionic émule une interface utilisateur native d'application en utilisant les Kits de développement. **ionic** utilise Capcitor / Cordova pour déployer en mode natif, ou s'exécute dans le navigateur comme une Progressive Web App.

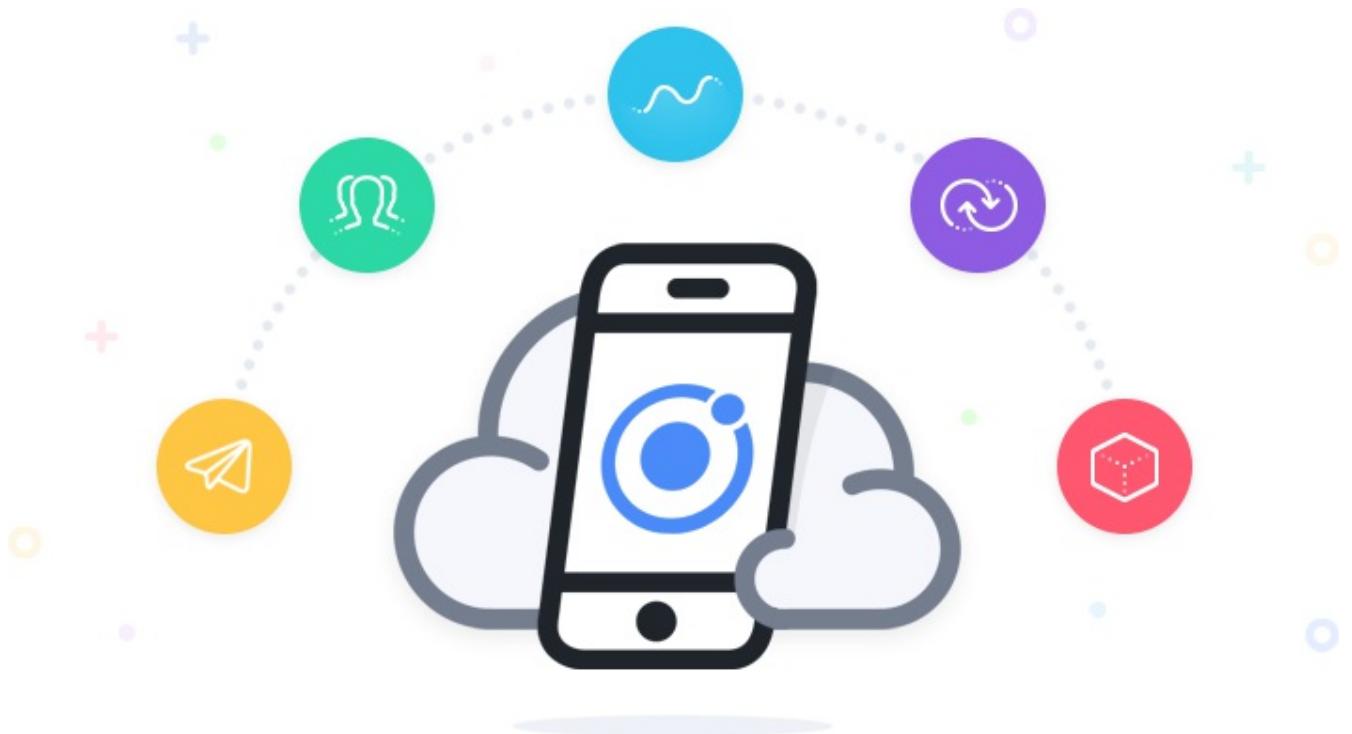
Free & Open Source Ionic Framework est 100% gratuit et open source, sous licence MIT. Il restera toujours libre d'utilisation, alimenté par une communauté mondiale .

Fully Cross-Platform Construire des progressive web et des applications mobiles natives pour chaque app store, avec une base de code.

Premier Native Plugins Utilisez plus de 120 fonctions natives de l'appareil telles que le Bluetooth, HealthKit, Finger Print Auth, et plus avec les plugins Cordova/PhoneGap.

First-class Documentation Construite avec de vrais exemples d'application, composant des démos, des guides, et comment faire pour le faire fonctionner avec des applications mobiles plus rapidement que jamais.

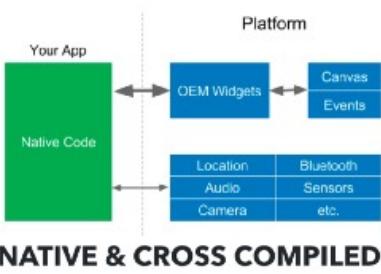
Présentation de l'offre de services Ionic.



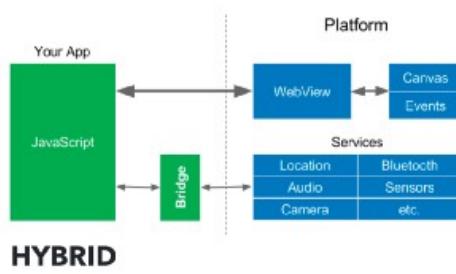
L'offre Pro regroupe quatre services principaux.

- [Ionic Deploy](#)
- [Ionic Package](#)
- [Ionic Docs](#)
- [Ionic Studio](#)

Framework de développement “hybride”, positionnement.

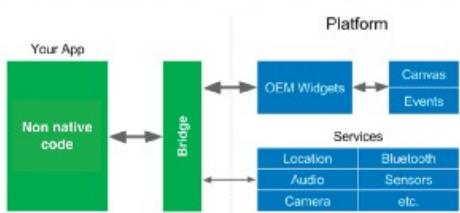


NATIVE & CROSS COMPILED

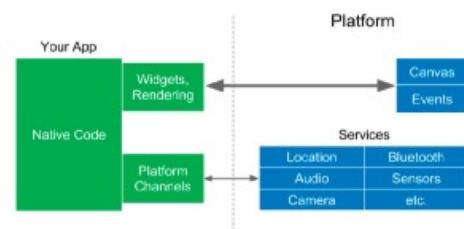


HYBRID

WEB NATIVE & CROSS COMPILED



FLUTTER





Cordova

Cordova

Cordova anciennement PhoneGap, a ouvert la route vers une uniformisation de l'accès aux fonctions natives des plates-formes mobiles. Non sans surprise, il reste aujourd'hui l'un des grands piliers du mobile hybride.

Non seulement il continue d'être maintenu et amélioré très régulièrement (sortie de la version 6 le 28 janvier 2016 !), mais surtout il sert de base à de nombreux frameworks dont Ionic dont nous parlons ci-dessous.

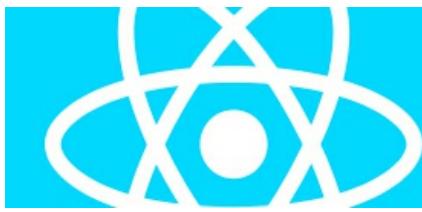
Clarification, PhoneGap est aujourd'hui le nom du service propriétaire d'Adobe, construite sur Cordova.

Points positifs

- compatible avec de très nombreuses plates-formes (Android, iOS, Windows Phone, Blackberry, Ubuntu mobile, Firefox OS, LG web OS, FireOS) ;
- liberté technologique (utilisez le framework qui vous convient, ou pas du tout de framework d'ailleurs !) ;
- performances entièrement dépendantes de ce que l'on en fait ;
- une base de plugins complète

Points négatifs

demande un travail plus conséquent pour avoir un résultat utilisable en production ;
interface graphique non comprise ;
absence de visualisation hors émulateur / périphérique par défaut.



React Native

React Native

React Native est le penchant mobile du framework de Facebook, React. Encore en cours de finalisation, il apporte une autre réponse au fameux mythe du “écrire une fois et déployer partout”.

L'intérêt principal est qu'il utilise la même bibliothèque (donc même syntaxe) que React, et permet donc, modulo les éléments spécifiques aux plates-formes mobiles, de réutiliser le code pour une version bureau ou web, tout en créant des versions natives pour Android et iOS.

Points positifs

- performances natives ;
- framework édité par Facebook, donc bien maintenu ;
- logique de composants ;
- même syntaxe que React (donc possibilité de sortir une version web).

Points négatifs

*framework jeune ; *compatibilité IOS et Android (partielle pour l'instant), mais pas de Windows Phone prévu ; *nécessité d'utiliser des conditions pour différencier le code spécifique aux composants natifs des différentes plates-formes. *n'est pas du web à proprement parler et donc demande une adaptation pour les développeurs.



Ionic

Ionic est aujourd’hui l’un des frameworks les plus en vogue pour faire des applications mobiles hybrides rapidement et de qualité. Depuis sa création en 2014, l’outil n’a cessé de s’améliorer et de fédérer autour de lui.

L’attrait principal du framework est qu’il offre toute une interface prête à utiliser, au sein d’un système de routage et d’outils complets. Construit autour Cordova, il profite de ses plugins et offre donc une surcouche à celui-ci.

Points positifs

- UI complète (avec un système de templating unifié pour toutes les plates-formes) ;
- création rapide pour la production ;
- construit avec AngularJS ;
- CLI très puissante (ex: visualisation dans le navigateur avec du live-reload) ;
- dépôt de plugins cordova spécifiquement maintenu par l’équipe du framework.

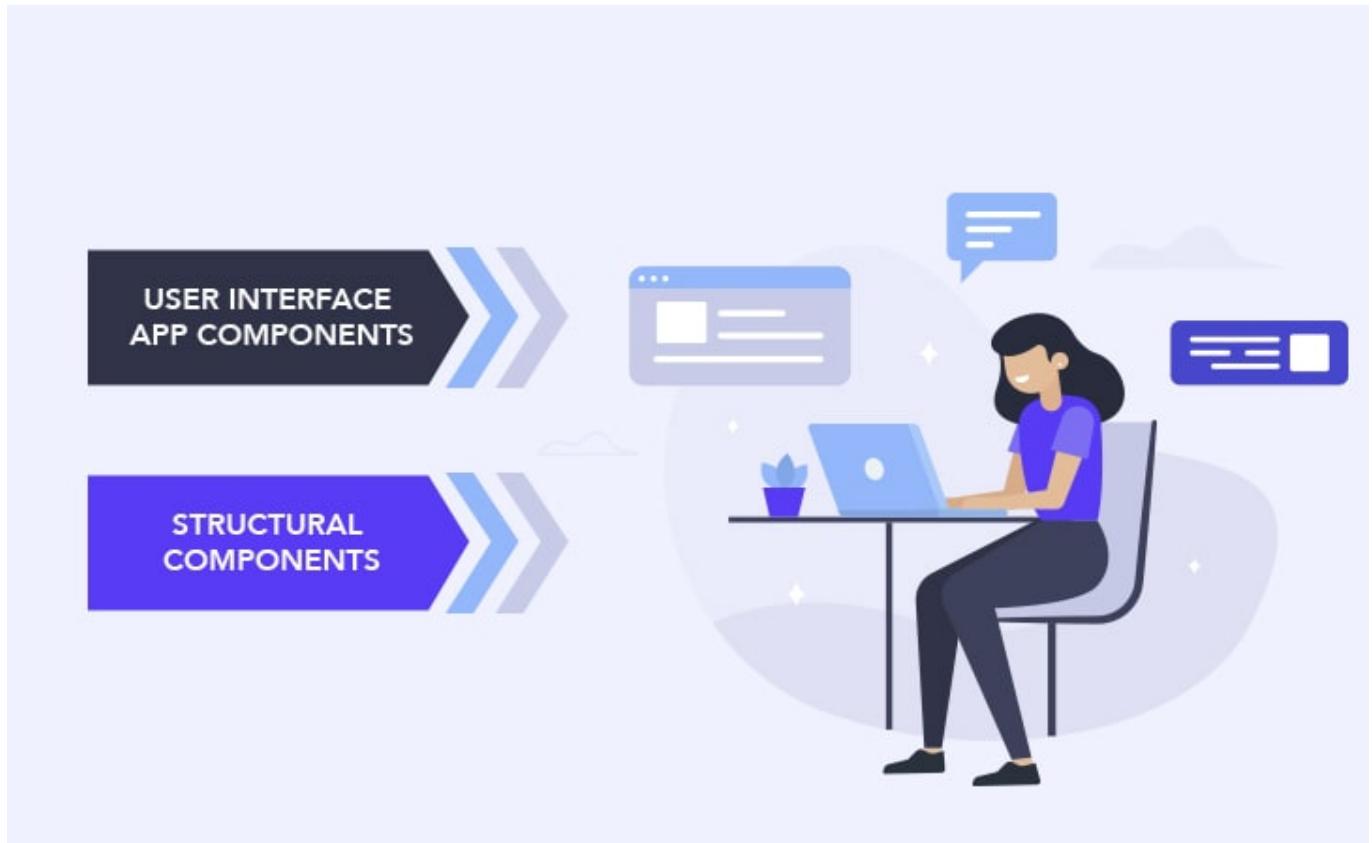
Points négatifs

- compatible uniquement iOS et Android (Windows possible mais difficile) ;
- obligation d'utiliser Angular ;

Composants

Les architectures d'applications Web comprennent divers composants qui sont séparés en deux catégories de composants:

- « **User Interface Components.** »
- « **Structural Components.** »



Composants d'interface utilisateur

Référence aux pages Web ayant pour rôle d'afficher des données, des paramètres et des configurations.

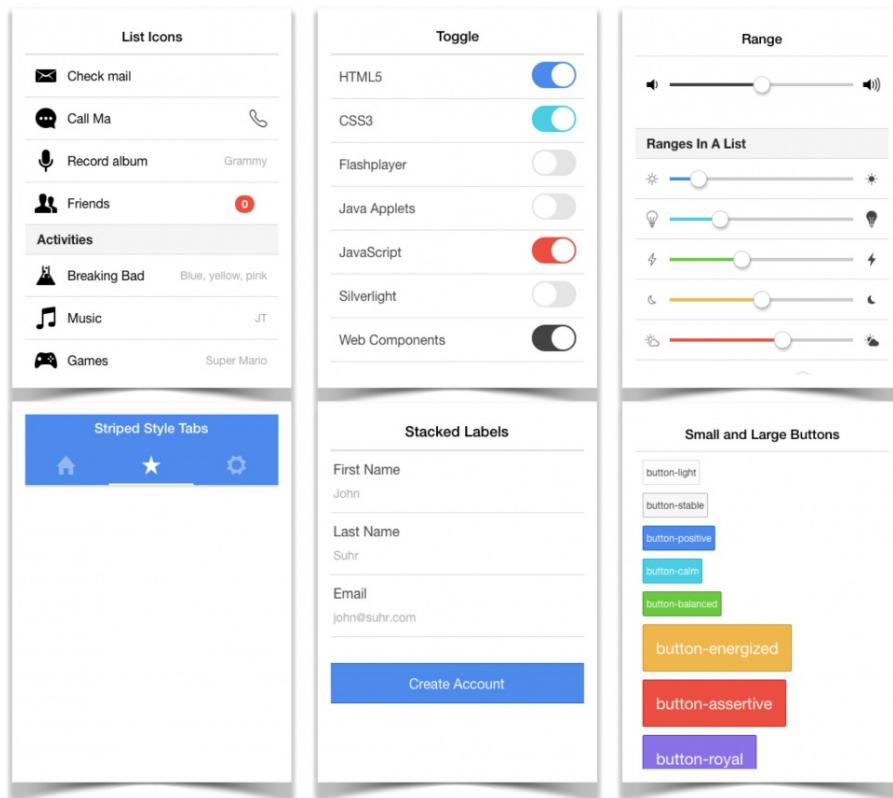
Liés à l'interface / expérience plutôt qu'au développement et, par conséquent, traite des tableaux de bord d'affichage, des paramètres de configuration, des notifications, des journaux, etc.

Apport du Framework CSS.

Les applications **Ionic** sont constituées de blocs de construction de haut niveau appelés composants. Les composants vous permettent de créer rapidement une interface pour votre application.

Ionic est livré avec un certain nombre de composants, y compris les modaux, les popups et les cartes. Consultez les exemples ci-dessous pour voir à quoi ressemble chaque composant et pour apprendre à les utiliser.

Une fois que vous connaissez les bases, rendez-vous sur [les documents de l'API](#) pour savoir comment personnaliser chaque composant.



Composantes : Utilitaires, CSS, JavaScript, Services.

Ionic va vous permettre de développer une application web qui aura l'air d'une application native. Elle en partagera également certaines capacités comme l'accès aux éléments systèmes. C'est ce qu'on appelle une application hybride. La majeure partie de son code est en technologie du web (javascript/html/css) et pour tout le reste, il y a des plugins qui font l'interface entre le smartphone et votre application.

[Ionic 1](#) est basé sur [Cordova](#) et [Angular 1](#). Il fournit des briques qui vous permettront de créer une application proche du natif.

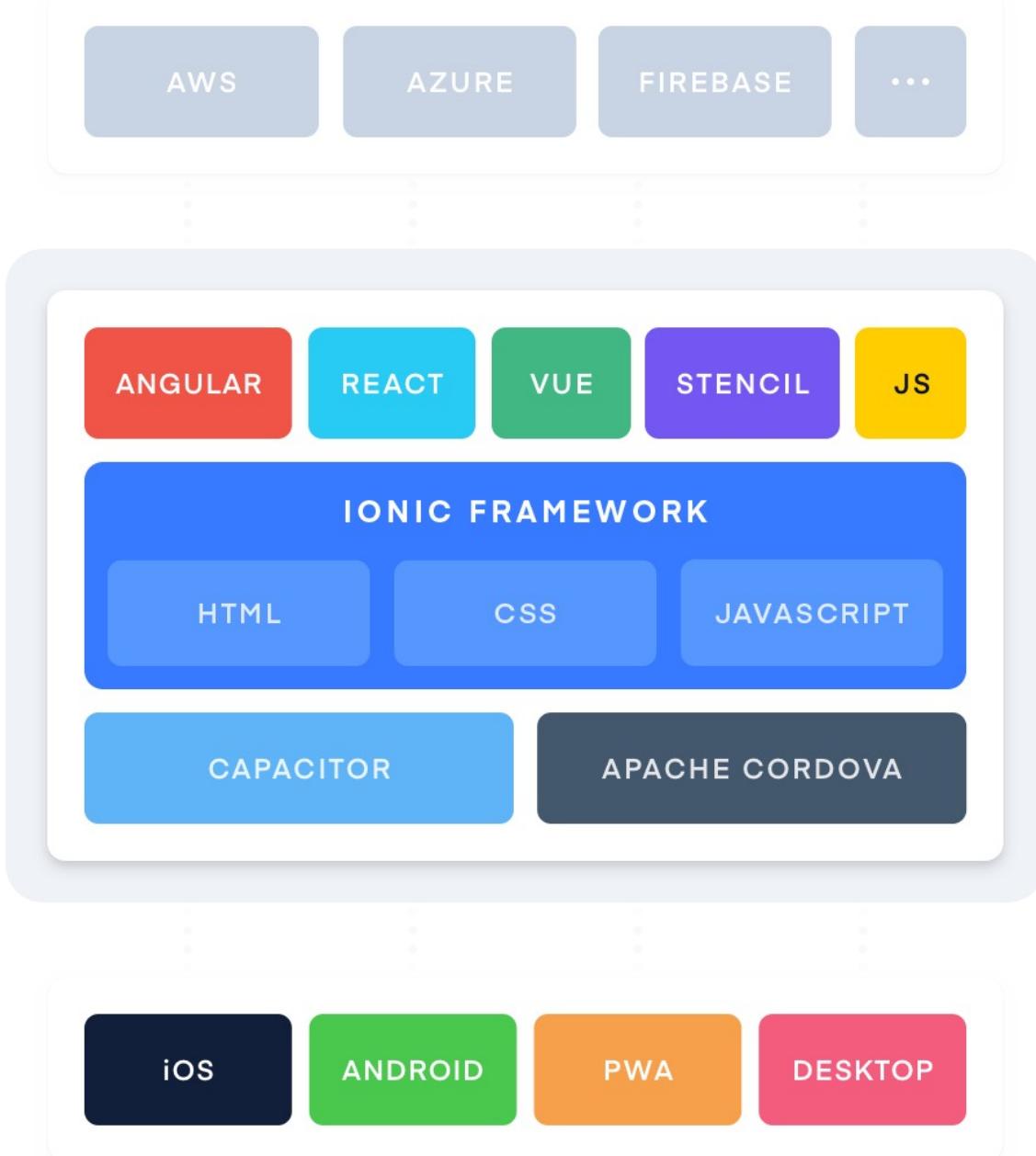
[Les équipes de Drifty](#) ont mis tout en œuvre pour faciliter la vie du développeur mobile hybride en lui fournissant un environnement simple et un framework pratique. Leur framework a tellement fait sensation qu'ils ont [levé un million de dollars](#) en 2014, alors que la startup n'avait que 2 ans d'existence.

Cela a permis à leurs équipes de repenser leur création, en apprenant de leurs erreurs, des demandes de la communauté et en mettant à jour les briques sur lesquelles reposent le framework.

La Stack Ionic, le choix d'Angular.

Ionic permet également d'utiliser [les évolutions de javascript](#) avec les syntaxes d'ES6/ES2015 et d'ES7.

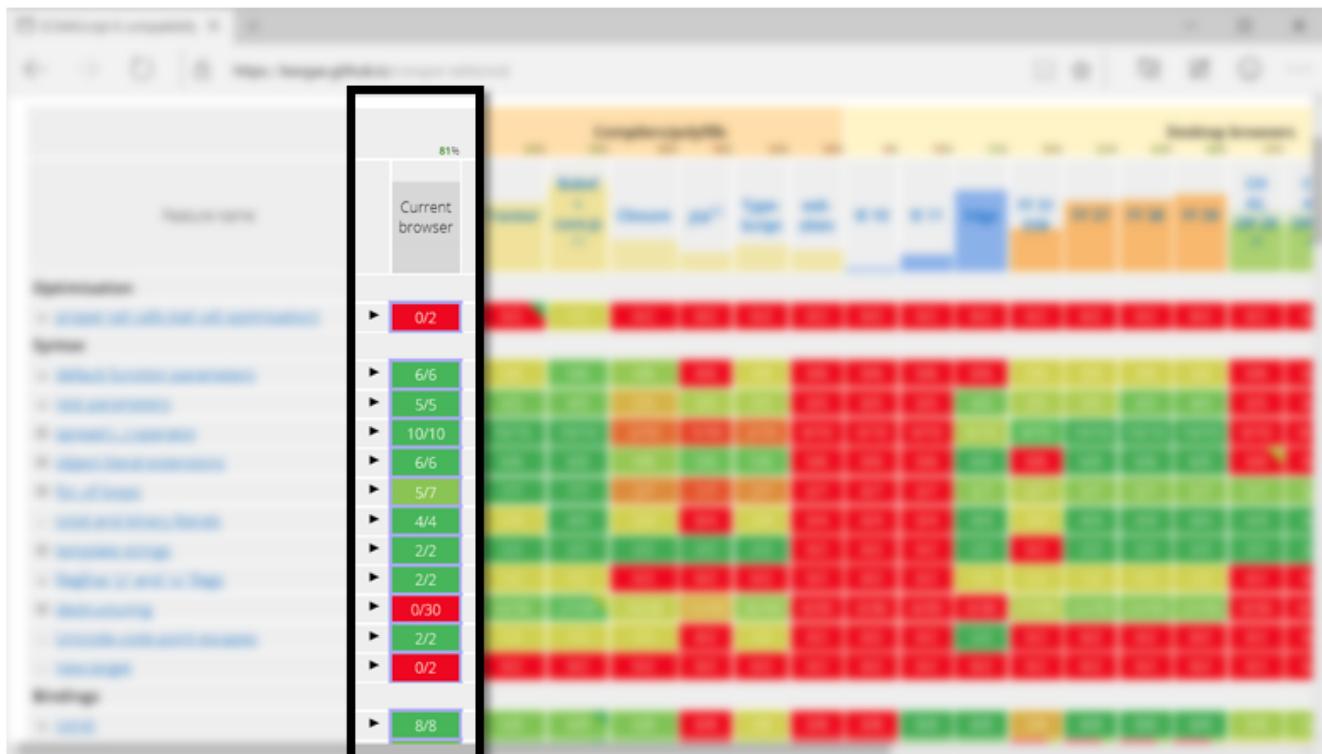
Ionic est fortement inspiré des concepts architecturaux Angular sans pour autant être exclusif.



- **Ionic CLI** : Les commandes très complètes peuvent par exemple permettre de configurer, build, tester et déployer ses apps.
- **Ionic Market** : C'est un peu comme un Play Store ou un App Store, mais il permet ici de trouver et rajouter des extensions 'gratuites ou non) à notre application, que ce soit des archétypes déjà prêts, des plugins ou encore des thèmes.
- **Ionic Creator** : Un IDE en ligne pour créer une application. Utilisation très intuitive avec le drag&drop pour ajouter des composants.
- **Ionic Native** : Ce sont des plugins permettant d'utiliser les plugins de Cordova en AngularJS, et donc pouvoir utiliser les fonctions natives de l'appareil. (Photo, Geolocalisation, etc)

Support courant pour ES6 : compilateurs, polifills, navigateurs serveurs.

La meilleure façon de suivre l'évolution du support ES6 sont les [tableaux de compatibilité ECAMScript](#) maintenu par **Juriy Zaytsev** (@kangax)



Configurer une application.

Fichiers

Les valeurs de Configuration sont stockés dans des fichiers **JSON**. Le **Ionic CLI** maintient un fichier de configuration global, généralement situé dans `~/.ionique/config.json`, et les fichiers de configuration du projet, généralement à la racine du répertoire dans `ionic.config.json`.

La CLI fournit des commandes pour la définition des valeurs de configuration de projet.

Voir `ionic config --help` ou la documentation.

Les Variables D'Environnement

La CLI va chercher les variables d'environnement suivantes:

- `IONIC_CONFIG_DIRECTORY`: Le répertoire de la mondial de de la CLI de config. Par défaut `~/.ionic`.
- `IONIC_HTTP_PROXY`: Définir l'URL de proxy .
- `IONIC_TOKEN`: Automatiquement authentifie avec **Ionic Appflow**.

Flags

Ce sont des options globales qui modifient le comportement d'une commande CLI.

- `--help`: Au lieu de l'exécution de la commande, consultez la page d'aide.
- `--verbose`: Afficher tous les messages du journal des fins de débogage.
- `--quiet`: Afficher uniquement les AVERTIR et les messages du journal des ERREURS.
- `--no-interactive`.
- `--confirm`: Activez auto-confirmation des invites de confirmation.

Configurer une application.

Fichiers

Les valeurs de Configuration sont stockés dans des fichiers **JSON**. Le **Ionic CLI** maintient un fichier de configuration global, généralement situé dans `~/.ionique/config.json`, et les fichiers de configuration du projet, généralement à la racine du répertoire dans `ionic.config.json`.

La CLI fournit des commandes pour la définition des valeurs de configuration de projet.

Voir `ionic config --help` ou la documentation.

Les Variables D'Environnement

La CLI va chercher les variables d'environnement suivantes:

- **IONIC_CONFIG_DIRECTORY**: Le répertoire de la mondial de de la CLI de config. Par défaut `~/.ionic`.
- **IONIC_HTTP_PROXY**: Définir l'URL de proxy .
- **IONIC_TOKEN**: Automatiquement authentifie avec **Ionic Appflow**.

Flags

Ce sont des options globales qui modifient le comportement d'une commande CLI.

- **--help**: Au lieu de l'exécution de la commande, consultez la page d'aide.
- **--verbose**: Afficher tous les messages du journal des fins de débogage.
- **--quiet**: Afficher uniquement les AVERTIR et les messages du journal des ERREURS.
- **--no-interactive**.
- **--confirm**: Activez auto-confirmation des invites de confirmation.

APIs HTML 5, les applications hybrides.

Les **applications Web progressives** utilisent des **API Web modernes** ainsi qu'une stratégie d'amélioration progressive traditionnelle pour créer des applications Web multiplateformes.

- **Stockage côté client** — un long guide montrant comment et quand utiliser le stockage Web, IndexedDB et les Service Workers.
- **Utiliser les Service Workers** — un guide plus détaillé couvrant l'API Service Worker.
- **Utiliser IndexedDB** — les principes fondamentaux de IndexedDB, expliqués en détail.
- **Utiliser l'API Web Storage** — l'API Web storage expliquée facilement. Chargement instantané des applications Web avec l'architecture Application Shell — un guide pour mettre en œuvre l'architecture App Shell pour créer des applications qui se chargent rapidement.
- **Utiliser l'API Push** — apprendre l'essentiel de l'API Web Push.
- **Utiliser l'API Notifications** — les notifications Web en bref.
- **Les éléments pour une conception adaptative** — apprendre les bases du responsive design, un sujet essentiel pour les mises en page des apps modernes.
- **Mobile first** — souvent lors de la création de mises en page d'applications réactives, il est judicieux de créer la mise en page mobile par défaut et de construire des mises en page plus larges ensuite.
- **Ajouter à l'écran d'accueil** — apprendre comment vos applications peuvent tirer profit de l'option Ajouter à l'écran d'accueil (A2HS : Add to Home Screen).

Avantages des PWA

Les PWA doivent être repérables, installables, interconnectables, indépendantes de l'état du réseau, progressives, ré-engageables, réactives aux différentes tailles d'affichage, et sûres.

A propos des Web Components:



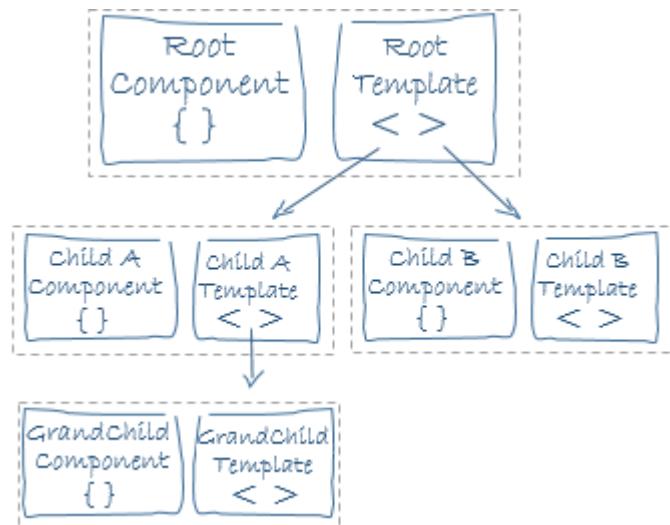
Composants d'interface graphique réutilisables, qui ont été créés en utilisant des technologies Web libres.

Les [Composants Web](#) sont constitués de plusieurs technologies distinctes. Ils font partie du navigateur, et donc ils ne nécessitent pas de bibliothèques externes comme jQuery ou Dojo.

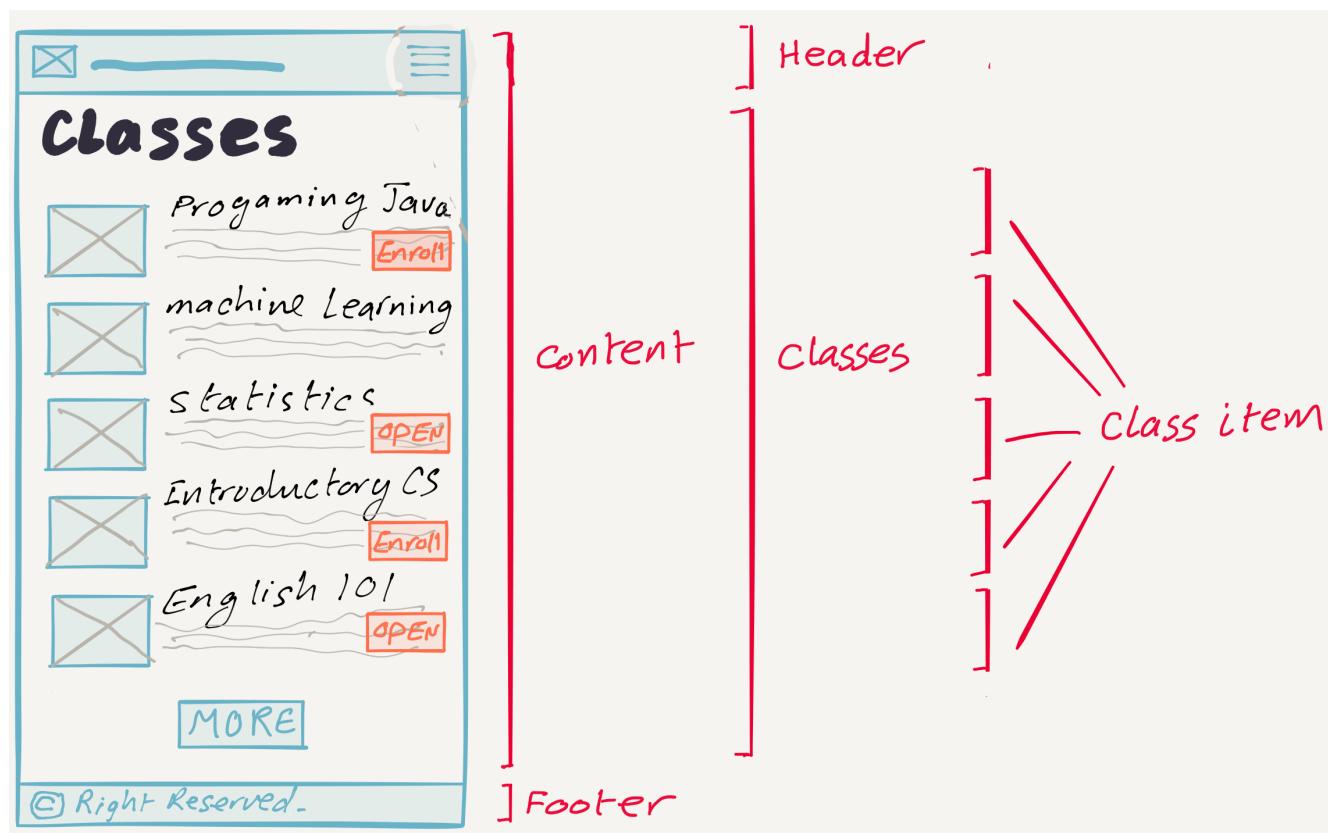
Un composant Web existant peut être utilisé sans l'écriture de code, en ajoutant simplement une déclaration d'importation à une page HTML.

Les Composants Web utilisent les nouvelles capacités standards de navigateur, ou celles en cours de développement.

Le framework angular permet de concevoir une application comme une arborescence de composants.



Les composants définissent des unités fonctionnelles plus ou moins réutilisables au sein de l'interface utilisateur.



La norme des Web Components.

Les Composants Web sont constitués de plusieurs technologies distinctes permettant de créer des composants d'interface graphique personnalisés et réutilisables, qui ont été créés en utilisant des technologies web libres.

Ils font partie du navigateur, et donc ne nécessitent pas de bibliothèque externe comme jQuery ou Dojo.

Un Composant Web existant peut être utilisé sans l'écriture de code, en ajoutant simplement une déclaration d'importation à une page HTML.

Les Composants Web sont constitués de quatre technologies (bien que chacune puisse être utilisée séparément) :

***Custom Elements**: pour créer et enregistrer de nouveaux éléments HTML et les faire reconnaître par le navigateur. ***HTML Templates**: squelette pour créer des éléments HTML instanciables. ***Shadow DOM**: permet d'encapsuler le JavaScript et le CSS des éléments. ***HTML Imports**: pour packager ses composants (CSS, JavaScript, etc.) et permettre leur intégration dans d'autres pages.

Les spécifications des Composants Web définissent les éléments suivants :

- De nouveaux éléments HTML : `<template>`, `<content>` et `<shadow>` (`<element>` et `<decorator>` ont été retirés).
- Les interfaces d'API associées pour les nouveaux éléments: `HTMLTemplateElement`, `HTMLContentElement` et `HTMLShadowElement`.
- Des extensions à l'interface `HTMLLinkElement` et l'élément . *Une API pour enregistrer les custom elements, `Document.registerElement()`, et des modifications de `Document.createElement()` et `Document.createElementNS()`.
- De nouvelles fonctions liées au cycle de vie ("`lifecycle callbacks`") peuvent être ajoutées à un prototype sur lequel est basé un custom element.
- Une nouvelle pseudo-classe CSS pour les éléments de style par défaut, `:unresolved`.
- Le Shadow DOM : `ShadowRoot` et `Element.createShadowRoot()`, `Element.getDestinationInsertionPoints()`, `Element.shadowRoot`.
- Une extension à l'interface `Event`, `Event.path`.
- Une extension à l'interface `Document`.
- Pour le style des Composants Web:
 - de nouvelles pseudo-classes : `:host`, `:host()`, `:host-context()`.
 - de nouveaux pseudo-elements : `::shadow` et `::content`.
 - un nouveau combinateur, `/deep/`.

Rappels DOM & AJAX.

DOM

Le DOM (Document Object Model) est une API qui représente et interagit avec tous types de documents HTML ou XML. Le DOM est un modèle de document chargé dans le navigateur. La représentation du document est un arbre nodal. Chaque nœud représente une partie du document (par exemple, un élément, une chaîne de caractères ou un commentaire).

Le DOM est l'une des API les plus utilisées sur le Web parce-qu'il autorise du code exécuté dans un navigateur à accéder et interagir avec chaque nœud dans le document. Les nœuds peuvent être créés, déplacés et modifiés. Des auditeurs d'évènements (event listeners) peuvent être ajoutés à des nœuds et déclenchés par un évènement donné.

AJAX

AJAX (Asynchronous JavaScript + XML) n'est pas une technologie en soi, mais un terme désignant une « nouvelle » approche utilisant un ensemble de technologies existantes, dont : **HTML ou XHTML, les feuilles de styles CSS, JavaScript, le modèle objet de document (DOM), XML, XSLT, et l'objet XMLHttpRequest**.

- La méthode AJAX peut être résumée comme un compromis : elle évite un rechargement complet de la page mais n'autorise pas davantage que ce qu'apporte une requête HTTP GET ou POST (ou HEAD) classique.
- La méthode AJAX a comme qualité de rester dans les standards HTTP, en plus d'être dans du côté client : c'est donc une méthode qui est totalement transparente dans les échanges standards entre un client et un serveur.
- La méthode AJAX est connue et reconnue, pouvant être utilisées à large échelle dans des bibliothèques (comme JQuery), dont l'intérêt de ces bibliothèques est d'accélérer la vitesse de développement.

```
<button id="ajaxButton" type="button">Faire une requête</button>

<script>
(function() {
    var httpRequest;
    document.getElementById("ajaxButton").addEventListener('click',
makeRequest);

    function makeRequest() {
        httpRequest = new XMLHttpRequest();

        if (!httpRequest) {
            alert('Abandon :( Impossible de créer une instance de XMLHttpRequest');
            return false;
        }
        httpRequest.onreadystatechange = alertContents;
```

```
httpRequest.open('GET', 'test.html');
httpRequest.send();
}

function alertContents() {
    if (httpRequest.readyState === XMLHttpRequest.DONE) {
        if (httpRequest.status === 200) {
            alert(httpRequest.responseText);
        } else {
            alert('Il y a eu un problème avec la requête.');
        }
    }
})();
</script>
```

Template HTML à chargement différé.

L'élément **HTML <template>** (ou Template Content ou modèle de contenu) est un mécanisme utilisé pour stocker du contenu HTML (côté client) qui ne doit pas être affiché lors du chargement de la page mais qui peut être instancié et affiché par la suite grâce à un script JavaScript.

Cet élément est un fragment de contenu mis de côté pour être utilisé par la suite dans le document.

Lorsque le moteur traite le contenu de l'élément **<template>** lors du chargement de la page, il ne fait que vérifier la validité du contenu, ce dernier n'est pas affiché.

```
<table id="producttable">
  </thead>
  <tbody>
    <!-- VIDE -->
  </tbody>
</table>

<template id="productrow">
  <tr>
    <td class="record"></td>
    <td></td>
  </tr>
</template>
<script>
// On vérifie si le navigateur prend en charge l'élément HTML template
if ("content" in document.createElement("template")) {

  var template = document.querySelector("#productrow");
  var tbody = document.querySelector("tbody");

  // On prépare une ligne pour le tableau
  var clone = document.importNode(template.content, true);
  var td = clone.querySelectorAll("td");
  td[0].textContent = "1235646565";
  td[1].textContent = "Stuff";

  // On clone la ligne et on l'insère dans le tableau
  tbody.appendChild(clone);

}

</script>
```

HTML Imports

HTML Imports est censé être un moyen de livrer des Composants Web dans une page, mais il est aussi possible d'utiliser HTML Imports seul.

On importe un fichier HTML cible à l'aide de la balise dans un document HTML source de la manière suivante :

```
<link rel="import" href="myfile.html">
```

La valeur d'attribut **import** de la balise link est un nouvel ajout.

HTML Imports n'est pas suffisamment supporté.

Shadow DOM, et CSS, les fragments de documents.

Le Shadow DOM permet de définir du comportement interne à notre DOM sans qu'il interfère sur les autres parties de notre application.

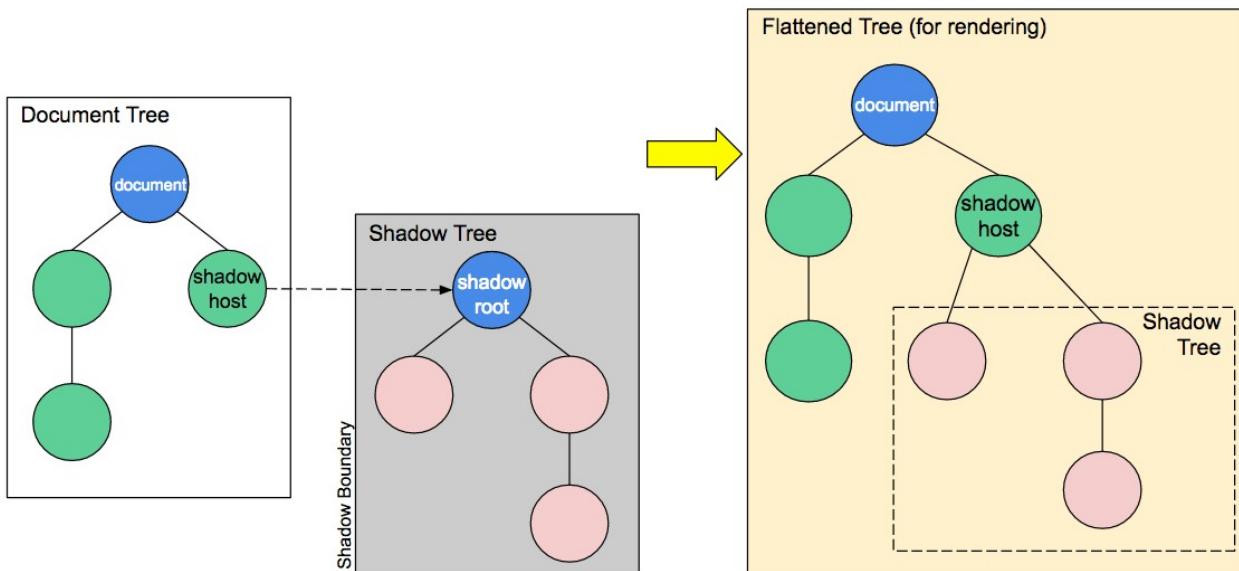
Il permet aussi de définir du style spécifique.

Un DOM dans un DOM

Vous pouvez considérer le Shadow DOM comme un "DOM dans un DOM". C'est son propre arbre DOM isolé avec ses propres éléments et styles, complètement isolé du DOM original.

Pour attacher un **Shadow DOM** à un hôte, on utilise la méthode `attachShadow()`.

```
const shadowEl = document.querySelector(".shadow-host");
const shadow = shadowEl.attachShadow({mode: 'open'});
const link = document.createElement("a");
link.href = shadowEl.querySelector("a").href;
link.innerHTML =
  `</span>
    ${shadowEl.querySelector("a").textContent}
  `;
shadow.appendChild(link);
```



CSS : le besoin d'encapsulation.

Le Shadow DOM correspond au « DOM caché » qui est encapsulé dans un composant.

L'isolation CSS est nécessaire à la cohabitation de composant différents.

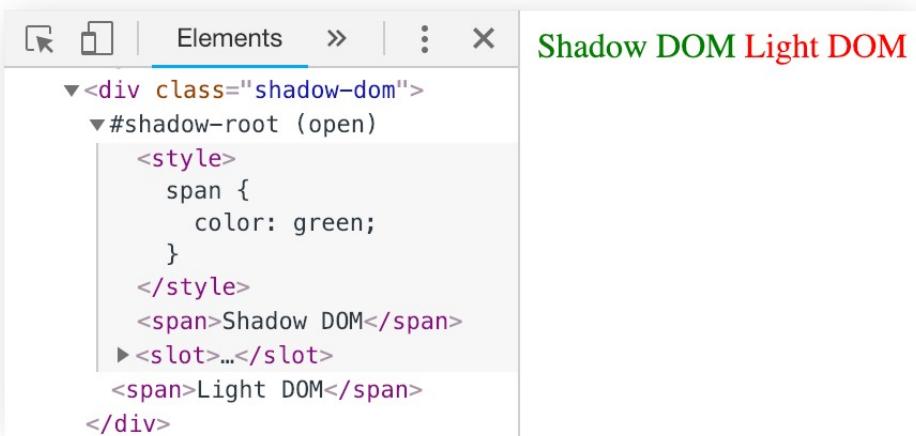
Ce DOM est isolé du reste de la page au niveau CSS et JavaScript, c'est à dire que les styles de la page courante ne lui seront pas appliqués, et que la fonction `querySelector` n'y accèdera pas non plus.

Mais si cela est nécessaire, la page hôte aura tout de même la possibilité de modifier le style du composant à l'aide de sélecteurs spécifiques :

- `:host(<selector>)` qui permet de styliser le host d'un élément ciblé par son sélecteur (optionnel)
- `:host-context(<selector>)` : pour styliser un élément en se basant sur ses éléments englobants.
- Le pseudo élément `::shadow` qui permet de cibler l'élément shadow-root
- Le mot clé `/deep/` qui, utilisé avec d'autres sélecteur, permet de passer au travers de n'importe quel nombre d'imbrication de shadow DOM.

```
:host {  
    opacity: 0.4;  
    transition: opacity 420ms ease-in-out;  
}  
:host(:hover) {  
    opacity: 1;  
}  
:host-context(.different) {  
    color: red;  
}  
#host::shadow span {  
    color: red;  
}  
body /deep/ .library-theme {  
    border: 1px solid red;  
}
```

Tous ces sélecteurs peuvent être utilisé dans le CSS, mais également en JS à l'aide de la méthode `querySelector`.



Custom Elements.

L'un des aspects les plus importants des composants web est la possibilité de créer des éléments personnalisés qui encapsulent bien vos fonctionnalités sur une page HTML.

Pour enregistrer un élément personnalisé sur la page, vous utilisez la méthode `CustomElementRegistry.define()`. Elle prend comme arguments :

- une DOMString représentant le nom que vous donnez à l'élément ;
- un objet de classe définissant le comportement de l'élément ;
- facultativement, un objet d'options contenant une propriété `extends`, qui indique l'élément intégré dont votre élément hérite, le cas échéant.

```
// L'objet de classe d'un élément personnalisé est écrit en utilisant la
syntaxe de classe ES 2015 standard.
customElements.define('word-count', WordCount, { extends: 'p' });
class WordCount extends HTMLParagraphElement {
    constructor() {
        // Toujours appeler "super" d'abord dans le constructeur
        super();

        // Ecrire la fonctionnalité de l'élément ici

        ...
    }
}
```

Notez que les noms d'éléments personnalisés doivent comprendre un tiret ; ils ne peuvent pas être des mots simples ;

Utilisation des rappels de cycle de vie

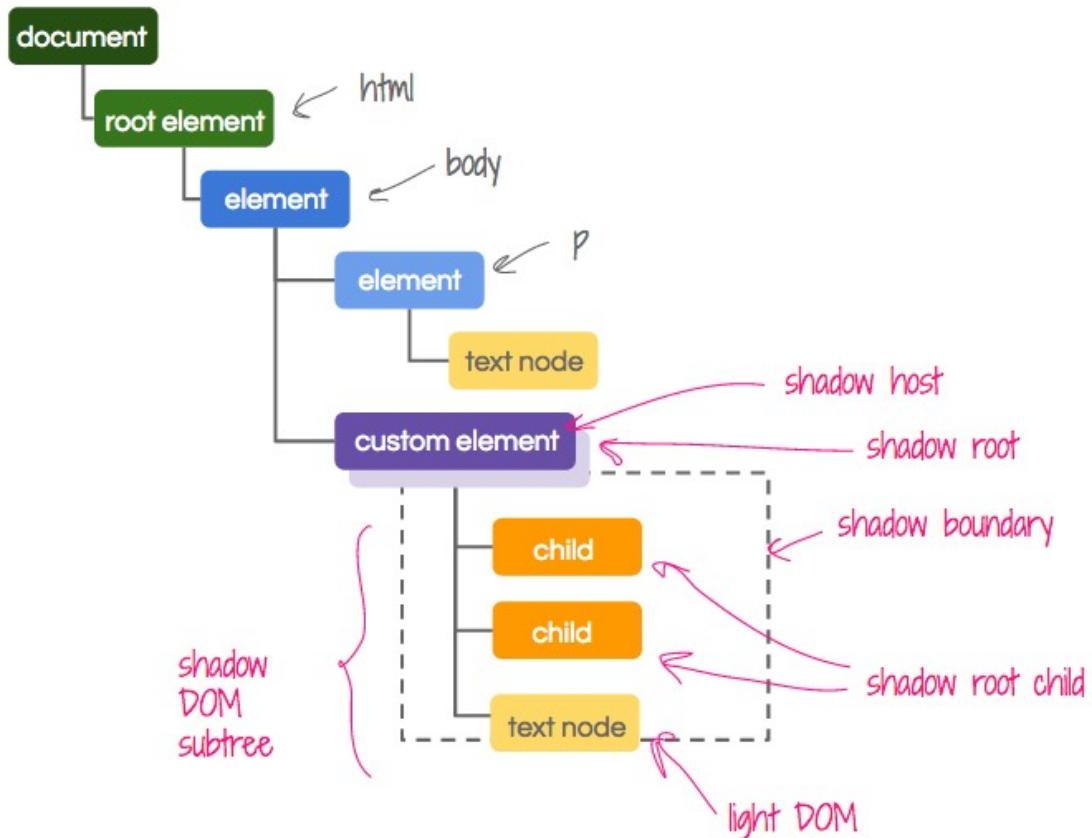
Vous pouvez définir plusieurs rappels différents dans le constructeur d'un élément personnalisé, qui se déclenchent à différents points du cycle de vie de l'élément :

- `connectedCallback` : appelé lorsque l'élément personnalisé est connecté pour la première fois au DOM du document ;
- `disconnectedCallback` : appelé lorsque l'élément personnalisé est déconnecté du DOM du document ;
- `adoptedCallback` : appelé lorsque l'élément personnalisé est déplacé vers un nouveau document ;
- `attributeChangedCallback` : appelé lorsque l'un des attributs de l'élément personnalisé est ajouté, supprimé ou modifié.

Notez que, pour déclencher le rappel `attributeChangedCallback()` lorsqu'un attribut change, vous devez observer les attributs.

Cela est réalisé en appelant le getter `observedAttributes()` dans le constructeur, en retournant un tableau contenant les noms des attributs que vous voulez observer :

```
static get observedAttributes() {return ['w', 'l']; }
```



<custom-element>

Structure

```
<div>
  <input>
  <p>
    <span></span>
  </p>
</div>
```

Behavior

```
tag.verifyAccount();
```

Styles

```
<style>
  p { color: red; }
</style>
```

StencilJS le compilateur de Web Component proposé par Ionic.

Stencil est un compilateur qui génère des **Web Components** (plus précisément, des Éléments Personnalisés). Stencil combine le meilleur des concepts des frameworks les plus populaires (**Angular**, **React**) en un simple outil de **build-time**.

Stencil takes features such as

- **Virtual DOM**
- **Async rendering** (inspired by React Fiber)
- **Reactive data-binding**
- **TypeScript**
- **JSX**

Conformes aux normes de composants web, ils peuvent fonctionner dans de nombreux frameworks car ce sont juste des composants web.

Starting a new project

Stencil nécessite une version LTS récente de NodeJS et npm.

```
npm init stencil
```

Composant

The **Stencil CLI** peut générer de nouveaux composants pour vous.

Vous pouvez simplement exécuter le script suivant dans votre projet, qui permettra de démarrer le générateur interactif.

```
npx stencil generate
```

La définition de composant suit l'API Stencil :

```
import { Component, Prop, h } from '@stencil/core';

@Component({
  tag: 'my-first-component',
})
export class MyComponent {

  // Indicate that name should be a public property on the component
  @Prop() name: string;
```

```
render() {
  return (
    <p>
      My name is {this.name}
    </p>
  );
}

}
```

Build

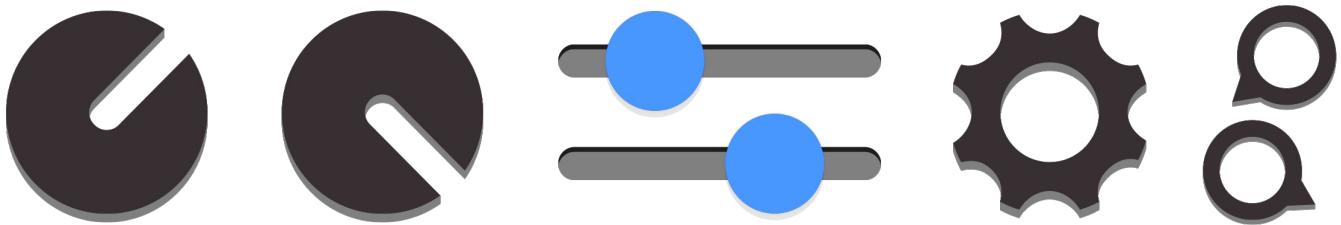
```
npx stencil build
```

Usage

```
<my-first-component name="Max"></my-first-component>
```



Le Framework Ionic en profondeur



Dans l'API, vous trouverez deux sortes de doc pages: Component et Service APIs.

- **Component APIs** inclre les classes et vous montre comment les utiliser, en plus de la liste de leurs sélecteurs, propriétés et événements.
- **Service APIs** sont des classes comme le **MenuController**, la configuration ou de la Plateforme. Ces services sont fournis par **Ionic** pour être injecté dans vos classes.

Nous allons maintenant explorer chacuns des composants et services.

Autres composantes.

- **IonIcon** est la liste des icônes fournies par le framework.
- **Storage**

Le **stockage** est un moyen facile de stocker des paires clé/valeur et les objets JSON. Le stockage utilise une variété de moteurs de stockage en dessous, choisi en fonction du meilleur disponible sur la plate-forme.

Lors de l'exécution dans une application native, le Stockage donnera la priorité à l'aide de SQLite, comme c'est l'un des plus stables et largement utilisé basé sur des fichiers de bases de données, et afin d'éviter certains des pièges de choses comme le localstorage et IndexedDB.

Lors de l'exécution dans le web ou Progressive Web App, le Stockage va tenter d'utiliser IndexedDB, WebSQL, et localstorage, dans cet ordre.

Présentation des composants, API et usages ergonomiques.

En s'appuyant sur la norme des [Web Component](#), [Ionic](#) offre désormais plus de 90 composants s'intégrant avec les différents frameworks modernes.

De part leurs nombres et les différentes intégrations un **énumération serait contre-productive** L'API propose un [index des composants](#)

Il faut se familiariser avec la richesse que le framework offre.

Heureusement ces composants entrent dans différentes catégories selon leurs usages.

Personnalisation ciblée de la plateforme (IOS/Android).

Ionic fournit des styles spécifiques en fonction de la plate-forme ou l'application est en cours d'exécution sur.

Le style des composants match celui des lignes directrices et permet à l'application d'être écrit une fois en gardant un look and feel natif en fonction du terminal.

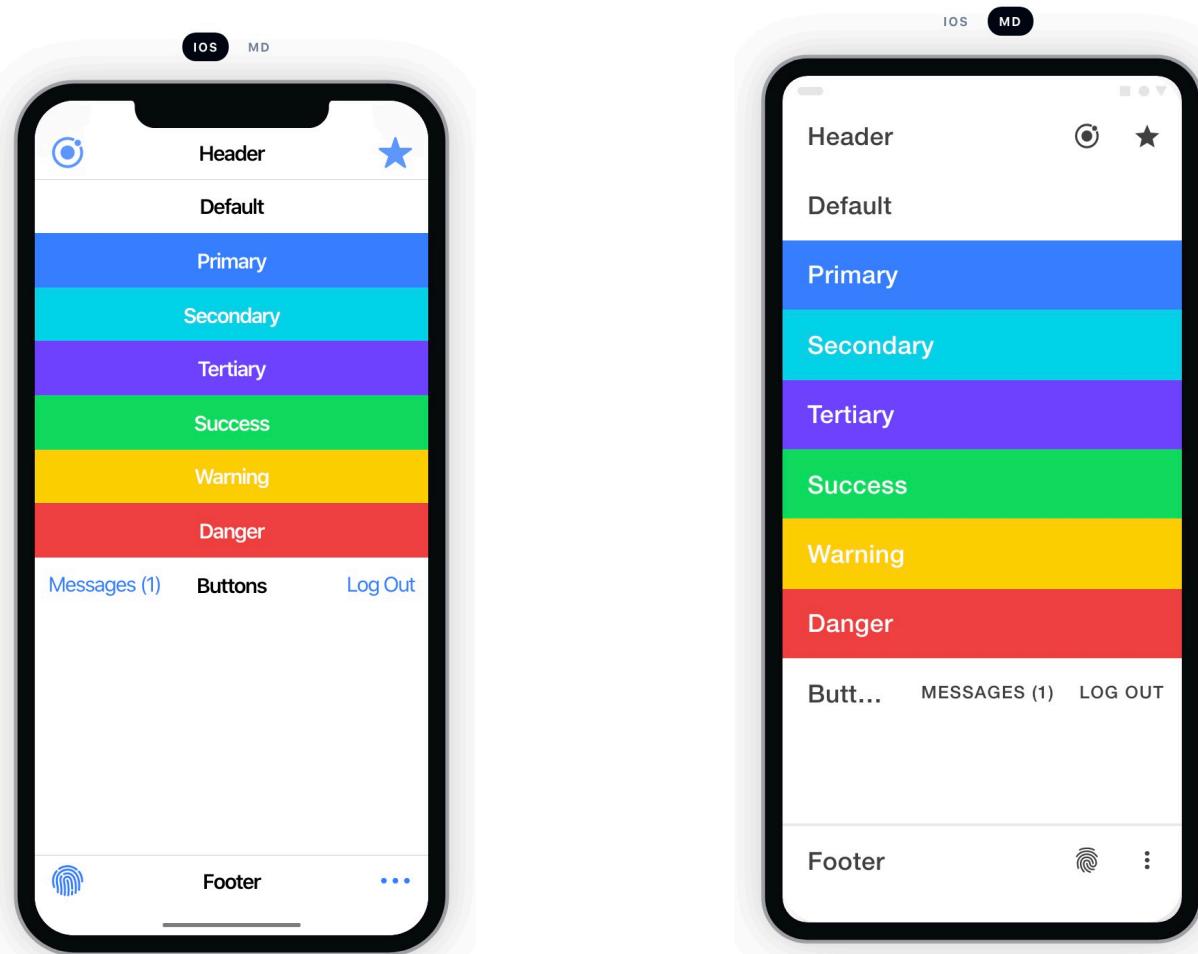
Chaque plate-forme dispose d'un mode par défaut, mais cela peut être modifié par le biais du global config.

La liste suivante affiche le **mode** par défaut qui est ajouté à chaque plate-forme:

- ios-**ios** :iPhone, iPad, iPod.
- android - **md**: Android.
- core - **md**

```
<html class="md|ios">
```

Le mode peut être redéfini dans la configuration de l'application.



Gestion du contenu.

CONTENT

- `ion-app`
- `ion-content`

GRID

- `ion-grid`
- `ion-col`
- `ion-row`

LIST

- `ion-list`
- `ion-list-header`
- `ion-virtual-scroll`

ITEM

- `ion-item`
- `ion-item-divider`
- `ion-item-group`
- `ion-item-sliding`
- `ion-item-options`
- `ion-item-option`
- `ion-label`
- `ion-note`

INFINITE SCROLL

- `ion-infinite-scroll`
- `ion-infinite-scroll-content`

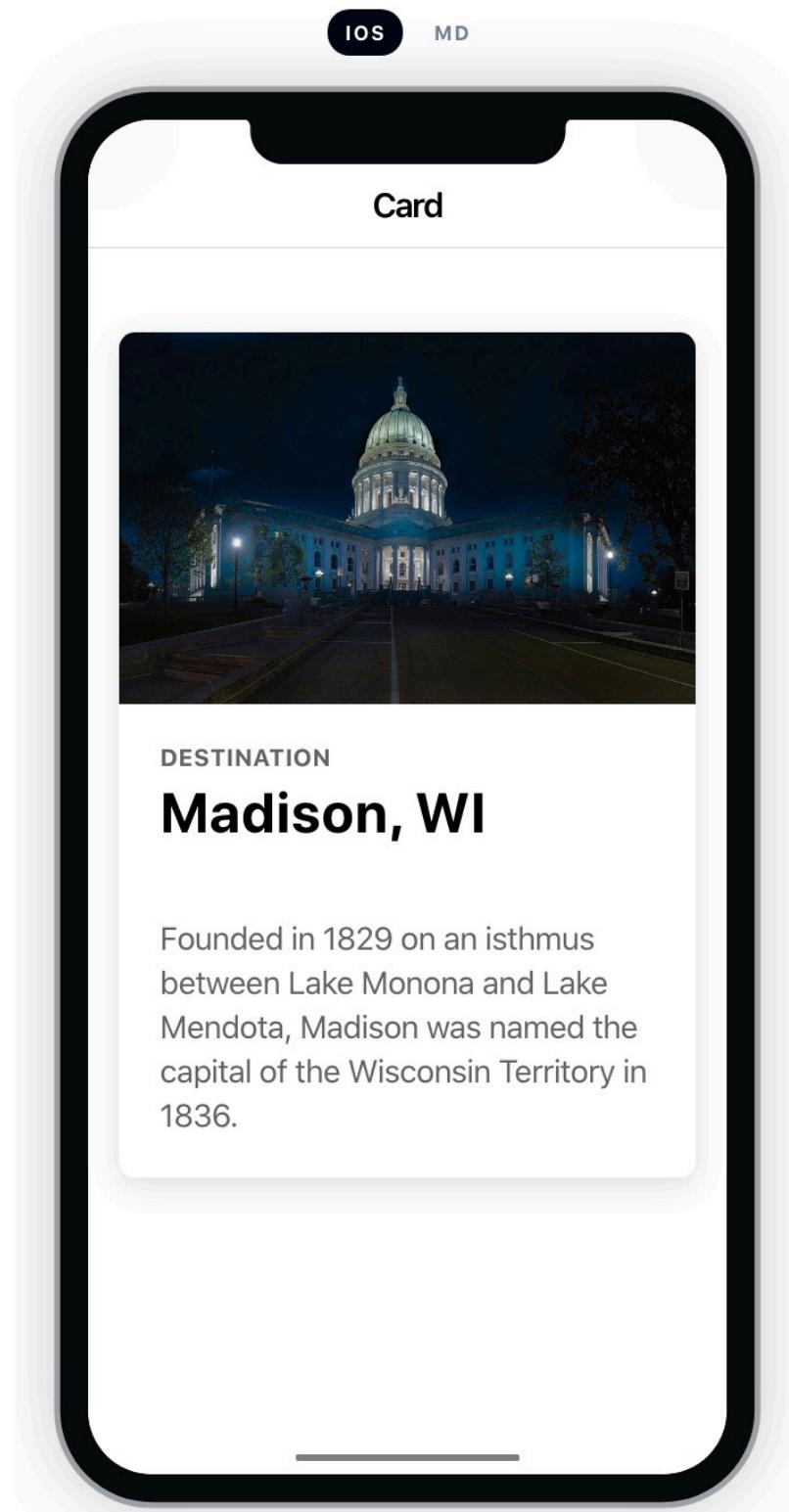
CARD

- `ion-card`
- `ion-card-content`
- `ion-card-header`
- `ion-card-subtitle`
- `ion-card-title`

BADGE

- `ion-badge`

CHIP



- `ion-chip`

ICONS

- `ion-icon`

MEDIA

- `ion-avatar`
- `ion-icon`
- `ion-img`
- `ion-thumbnail`

TYPOGRAPHY

- `ion-anchor`
- `ion-text`

TOOLBAR

- `ion-toolbar`
- `ion-header`
- `ion-footer`
- `ion-title`
- `ion-buttons`
- `ion-back-button`

Gestion de la navigation.

NAVIGATION

- `ion-nav`
- `ion-nav-pop`
- `ion-nav-push`
- `ion-nav-set-root`

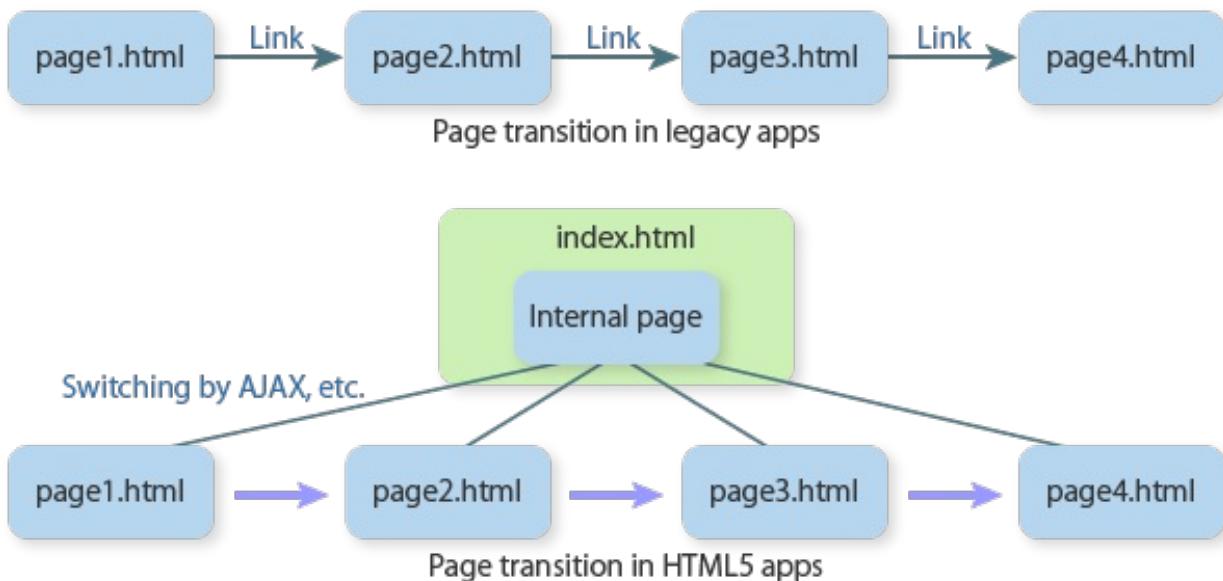
ROUTING

- `ion-router`
- `ion-router-link`
- `ion-router-outlet`
- `ion-route`
- `ion-route-redirect`

SPA (Single Page Application) routes et navigation.

Une application web monopage (en anglais single-page application ou SPA) est une application web accessible via une page web unique. Le but est d'éviter le chargement d'une nouvelle page à chaque action demandée, et de fluidifier ainsi l'expérience utilisateur. Deux méthodes existent pour ce faire : l'ensemble des éléments de l'application est chargé (contenu, images, CSS et JavaScript) dans un unique fichier HTML, soit les ressources nécessaires sont récupérées et affichées dynamiquement en fonction des actions de l'utilisateur. Le terme a été introduit par Steve Yen en 2005.

L'enregistrement en local de la page définissant une application web monopage et la possibilité de continuer à l'exécuter en local est l'une des propriétés importantes des applications web monopage qui les distingue des applications web standards qui reposent sur l'existence d'un serveur HTTP avec lequel elles échangent données, continuations applicatives et interfaces.

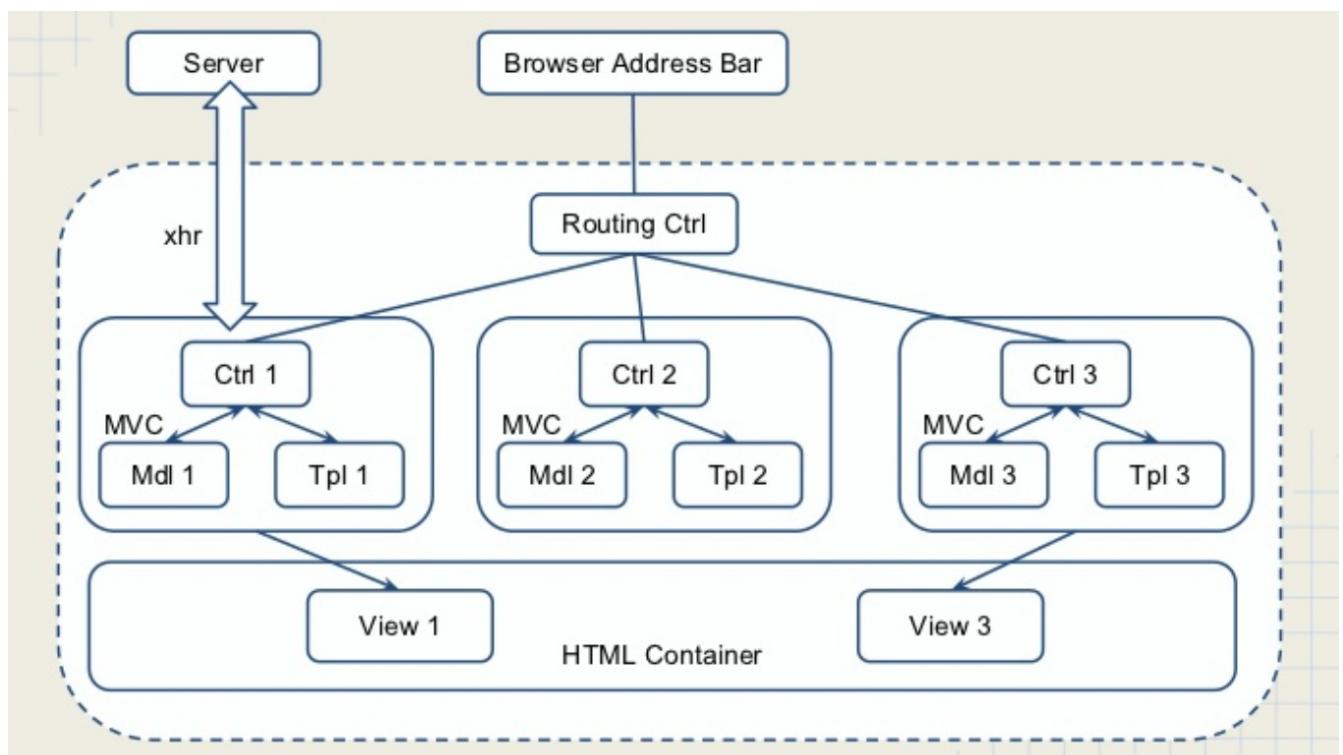


Qu'est-ce que le routage ?

Un système de routage permet de faire correspondre une URL donnée à une page précise.

Autrement dit, grâce à un système de routage, on sera toujours capable d'envoyer notre visiteur là où il le veut de façon automatisée.

De plus, vous pourrez établir une table de routage dans laquelle vous spécifierez ces correspondances ce qui vous permettra de complètement définir vos URL.



Composants interactifs.

ACTION SHEET

- `ion-action-sheet`
- `ion-action-sheet-controller`

ALERT

- `ion-alert`
- `ion-alert-controller`

MODAL

- `ion-modal`
- `ion-modal-controller`
- `ion-backdrop`

POPOVER

- `ion-popover`
- `ion-popover-controller`

PROGRESS INDICATORS

- `ion-loading`
- `ion-loading-controller`
- `ion-progress-bar`
- `ion-skeleton-text`
- `ion-spinner`

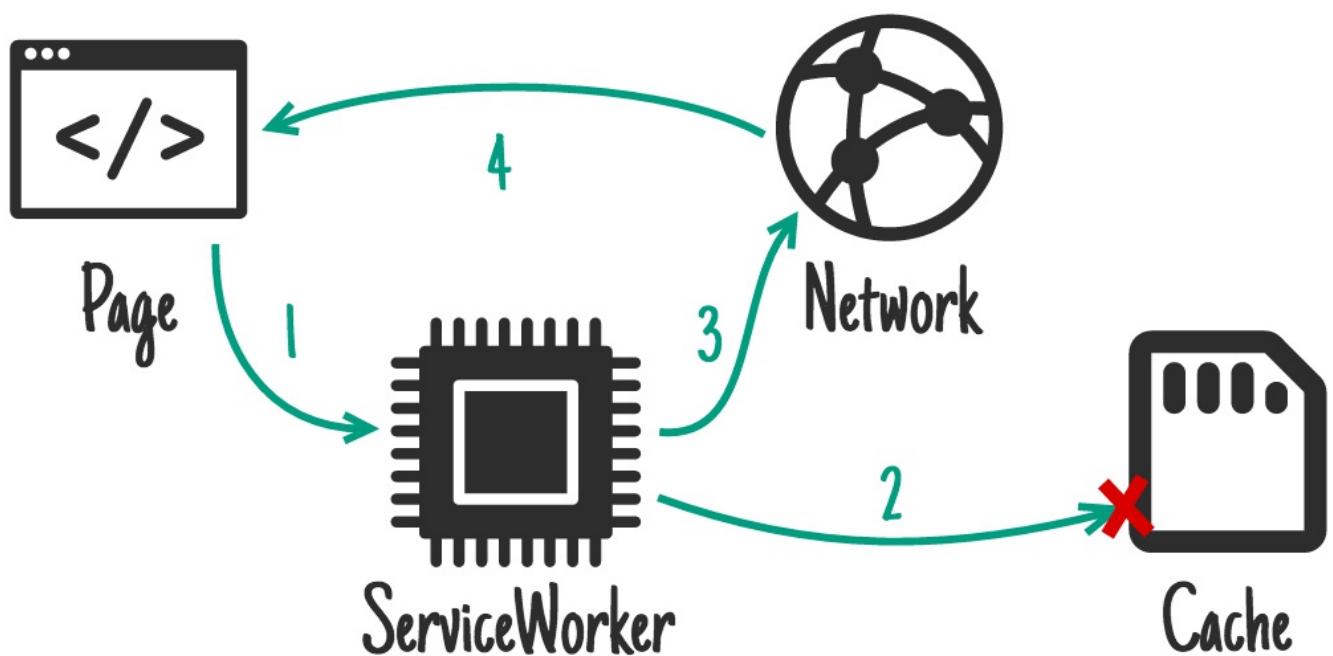
TOAST

- `ion-toast`
- `ion-toast-controller`

Adopter une stratégie “offline first”.

Le **Offline First** est un terme qui amène le développeur à privilégier le “offline” ou “hors ligne”. C'est une méthodologie qui nous enseigne que le fonctionnement d'un site ou d'une application ne devrait pas être dépendant d'une connexion.

Certaines applications, que vous connaissez bien, fonctionnent d'ores et déjà sur le modèle “offline”. Par exemple, les applications que vous utilisez tous les jours telles que le calendrier ou encore les contacts et les mails ont quelque chose en commun : elles synchronisent les données sur votre téléphone de façon locale, afin qu'elles puissent fonctionner même hors ligne.



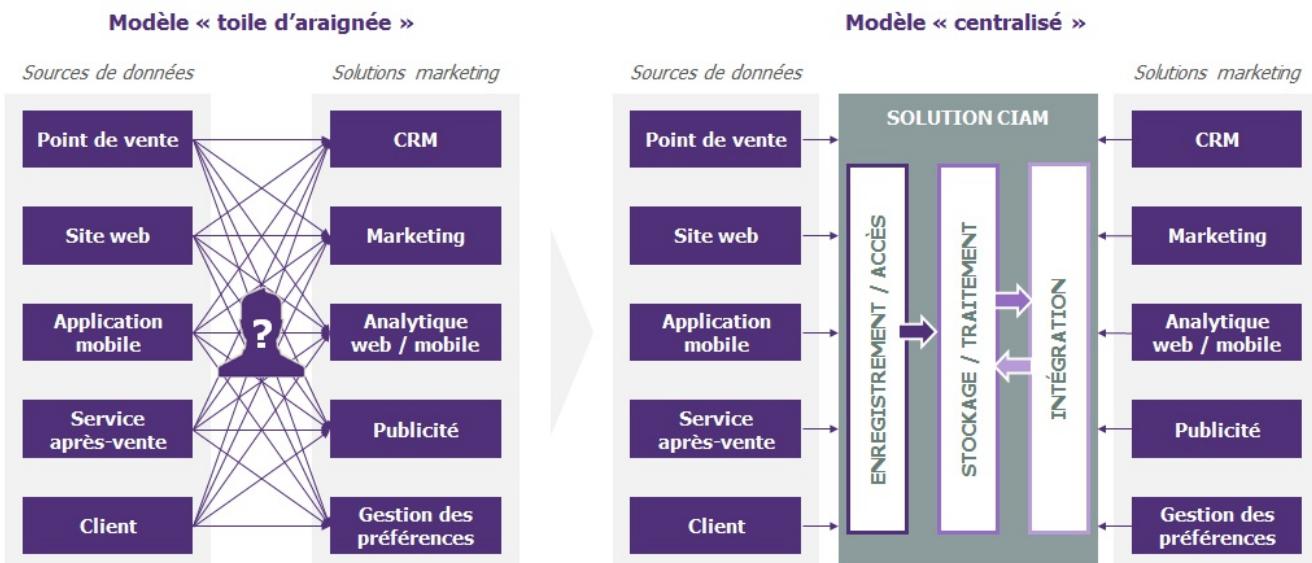
Les développeurs se doivent de penser “offline” pour plusieurs raisons :

- **Faire face aux problèmes de services** dûs aux surencombrements du réseau.
- **La rapidité** : la rapidité de navigation est parfois soit bien en dessous des espérances.

- **Faire face aux problèmes de batterie :** Certaines applications sont très gourmandes en énergie et par conséquent, elles vident votre batterie très rapidement.

Centraliser les données : redéfinir le cycle UX.

Quand on parle de centralisation des données, on parle d'un maximum de données récupérées et classées depuis tous les canaux possibles.



La collecte de ces données et la centralisation qui en découle doivent impérativement être conçues de façon cross canal, afin de réconcilier le plus d'informations possibles :

- les données nominatives (nom, prénom, raison sociale, etc.) ;
- les données socio-démographiques (situation familiale, nombre d'enfants, etc.) ;
- les comportements d'achat (historique des achats, montants et types de paiement, etc.) ;
- les données relationnelles (historique des sollicitations commerciales, SAV, etc.) ;
- les comportements de navigation (origine des visites, mots-clés, nombre de pages visitées, fréquence, taux de conversion, etc.) ;
- les données des études internes (enquêtes de satisfaction, etc.).

Gestuelle et interactions utilisateur.

BUTTON

- `ion-button`
- `ion-ripple-effect`

CHECKBOX

- `ion-checkbox`

DATE & TIME PICKERS

- `ion-datetime`
- `ion-picker`
- `ion-picker-controller`

FLOATING ACTION BUTTON

- `ion-fab`
- `ion-fab-button`
- `ion-fab-list`

INPUT

- `ion-input`
- `ion-textarea`

MENU

- `ion-menu`
- `ion-menu-button`
- `ion-menu-controller`
- `ion-menu-toggle`
- `ion-split-pane`

RADIO

- `ion-radio`
- `ion-radio-group`

RANGE

- `ion-range`

REFRESHER

- `ion-refresher`

- `ion-refresher-content`

REORDER

- `ion-reorder`
- `ion-reorder-group`

SEARCHBAR

- `ion-searchbar`

SEGMENT

- `ion-segment`
- `ion-segment-button`

SELECT

- `ion-select`
- `ion-select-option`

SLIDES

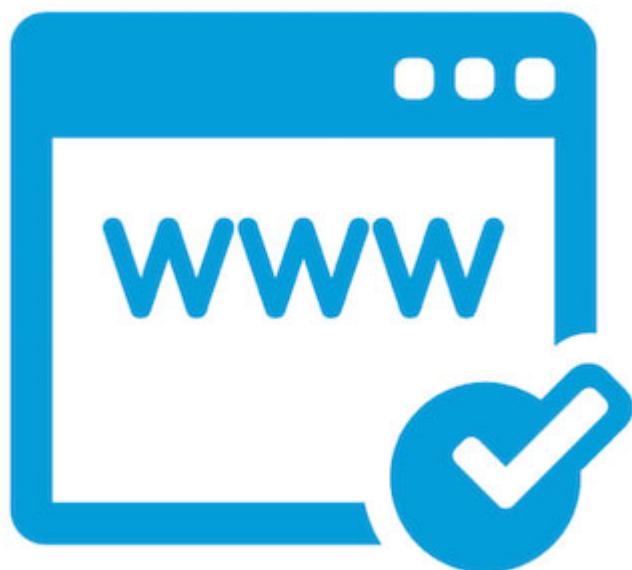
- `ion-slides`
- `ion-slide`

TABS

- `ion-tabs`
- `ion-tab`
- `ion-tab-bar`
- `ion-tab-button`

TOGGLE

- `ion-toggle`



Réutilisation : composants cross-frameworks

Architecture des applications Web: composants, modèles et types

Internet évolue progressivement vers un engagement actif des utilisateurs et des fonctionnalités étendues offertes par les applications Web. De plus en plus de développeurs cherchent à développer des applications Web et à attirer davantage de visiteurs vers leurs ressources Web.

L'un des défis que tout développeur peut rencontrer avant de se lancer dans le développement d'applications Web consiste à choisir le type et le modèle de composant de l'architecture Web.



Types d'architecture d'application Web.

Les applications Web comprennent deux ensembles différents de programmes qui s'exécutent séparément mais simultanément, l'objectif commun étant de travailler en harmonie pour fournir des solutions.

En règle générale, les deux ensembles de programmes incluent le code dans le navigateur qui fonctionne selon les entrées de l'utilisateur et le code dans le serveur qui fonctionne selon les demandes de protocoles, HTTPS.

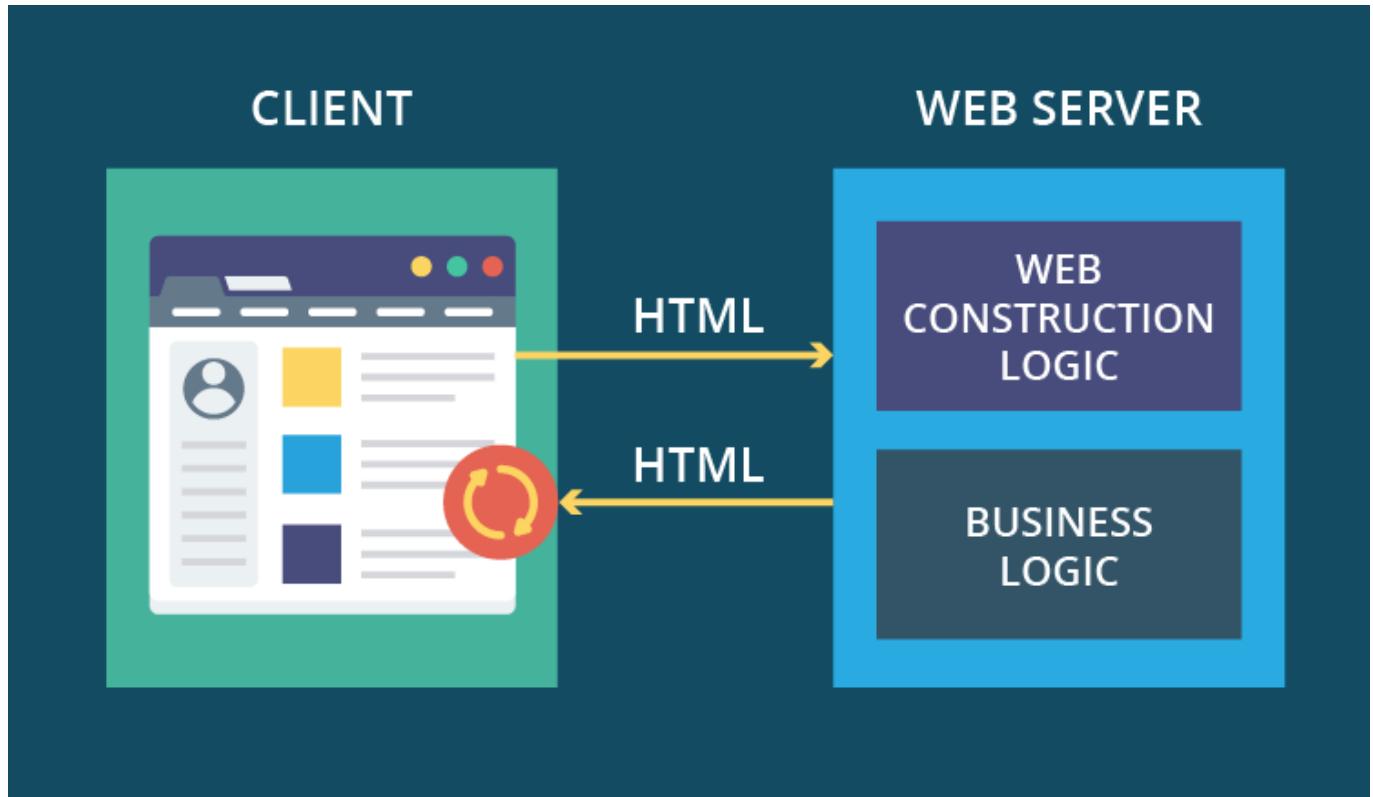
Quel que soit le modèle, tous les composants de l'application Web fonctionnent toujours simultanément et créent une application Web intégrale.

En fonction de la répartition de la logique de l'application entre le client et le serveur, il peut exister différents types d'architecture d'application Web.

Application Web HTML « Legacy »

Selon la toute première architecture de base d'applications Web, un serveur constitué d'une logique de construction de page Web et d'une logique métier interagit avec un client en envoyant une page HTML complète.

Pour voir une mise à jour, l'utilisateur doit entièrement recharger la page ou, en d'autres termes, demander au client d'envoyer une demande de page HTML au serveur et charger à nouveau son code complet.



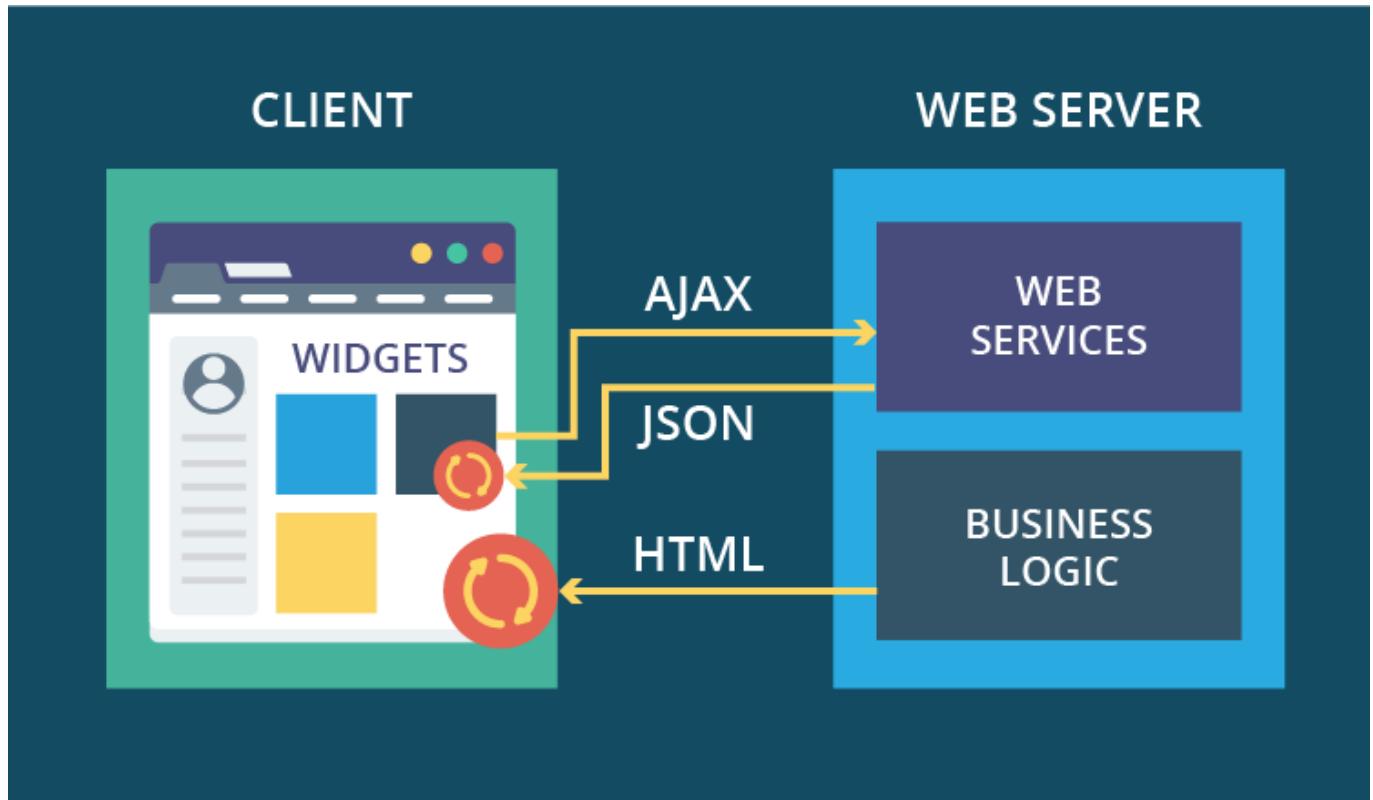
Étant donné que toutes les logiques et les données sont stockées sur le serveur et que l'utilisateur n'y a aucun accès, ce type d'architecture est hautement sécurisé.

Néanmoins, en raison du rechargement constant du contenu et de l'échange de données considérable, il est plus courant pour les sites Web statiques que pour les applications Web réelles.

Widget

Dans ce type, la logique de construction de page Web est remplacée par des services Web et chaque page du client possède des entités distinctes appelées widgets.

En envoyant des requêtes AJAX aux services Web, les widgets peuvent recevoir des fragments de données au format HTML ou JSON et les afficher sans recharger la page entière.



Avec les mises à jour des widgets en temps réel, ce type est plus dynamique, compatible avec les appareils mobiles et presque aussi populaire que le type suivant.

Cependant, cette architecture d'application Web nécessite un temps de développement plus long et est moins sécurisée en raison du transfert partiel de la logique de l'application vers le client exposé.

Développement 'cross-plateformes/cross-projet'.

Ionic CLI prend en charge une configuration **multi-app**, qui implique de multiples applications et le code partagé au sein d'un référentiel unique, ou **monorepo**.

Étapes De Configuration

Créer un répertoire et initialiser un monorepo.

Créer un fichier **ionic.config.json** à la racine du référentiel avec le contenu suivant :

```
{  
  "projects": {}  
}
```

Utilisation **ionic start** dans le monorepo pour créer des applications.

Project Structure

Dans un environnement multi-projet d'application, la structure du projet est flexible.

La seule exigence est un fichier multi-application **ionic.config.json** à la racine du dépôt.

Exemple

```
apps/  
myApp/  
myOtherApp/  
lib/  
ionic.config.json  
package.json
```

Config File

Dans un environnement multi-projet d'application, les applications partagent un seul fichier de configuration

Une application par défaut peut être spécifiée à l'aide de **defaultProject**.

```
{  
  "defaultProject": "myApp",  
  "projects": {  
    "myApp": {  
      "name": "My App",  
      "integrations": {},  
      "type": "angular",  
      "root": "apps/myApp"  
    },  
  },  
}
```

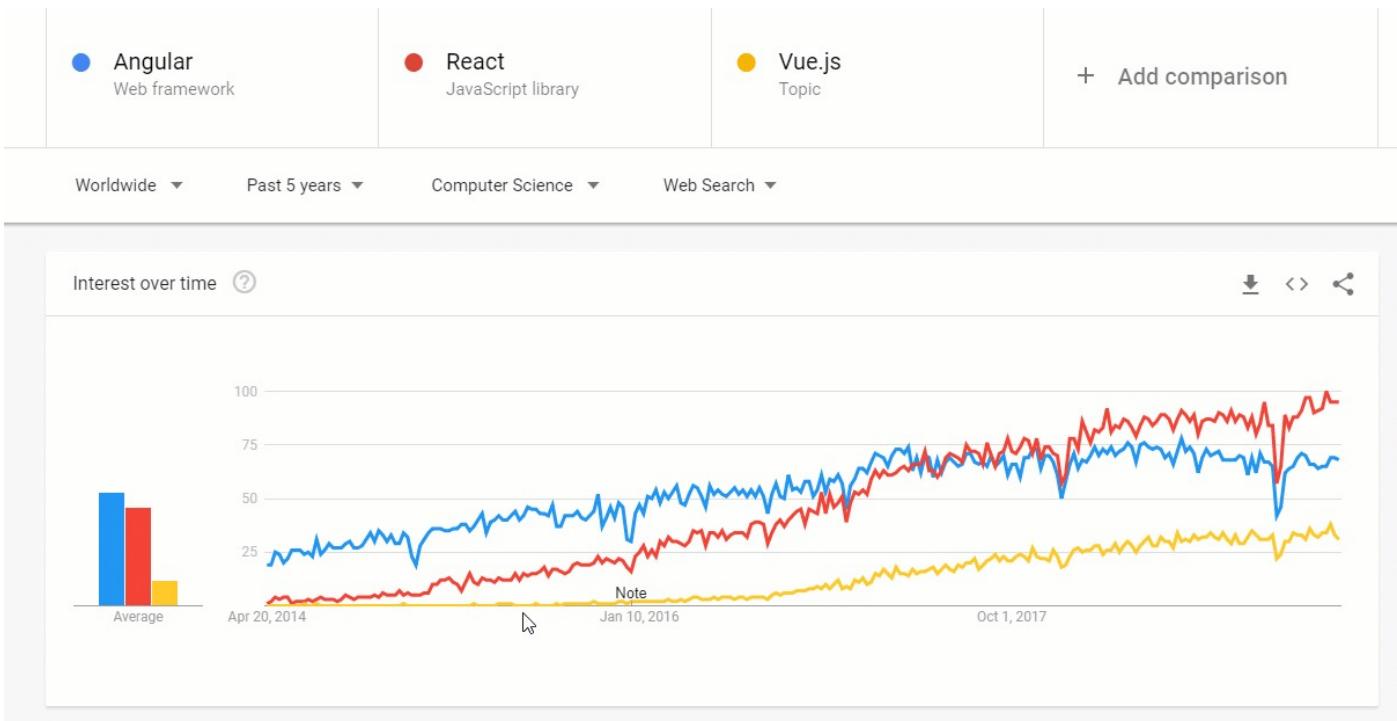
```
"myOtherApp": {  
    "name": "My Other App",  
    "integrations": {},  
    "type": "angular",  
    "root": "apps/myOtherApp"  
}  
}  
}
```

Ajout d'une Application

Si un fichier multi-projet d'application est détecté, le CLI ajoutera la configuration de l'application à la racine au lieu de créer un projet spécifique.

```
cd apps/  
ionic start "My New App" --no-deps
```

Les solutions du marché.



Exécuter la ligne suivante dans le terminal de ligne de commande pour installer le **Ionic CLI** (si besoin).

native-run est utilisé pour exécuter des binaires natifs sur les appareils et les simulateurs/émulateurs, et **cordova-res**, pour générer les icônes et écrans de démarrage:

```
npm install -g ionic native-run cordova-res
```

Angular, le choix par défaut.

Créer une application qui utilise un modèle de démarrage et ajoute un **capacitor** pour la fonctionnalité native:

```
ionic start photo-gallery tabs --type=angular --capacitor
```

PWA

Certains plugins, y compris la Caméra, sont basés sur des fonctionnalités web et des éléments de la bibliothèque PWA.

C'est une dépendance séparée, afin de l'installer:

```
npm install @ionic/pwa-elements  
Next, import @ionic/pwa-elements by editing src/main.ts.
```

```
import { defineCustomElements } from '@ionic/pwa-elements/loader';  
  
// Call the element loader after the platform has been bootstrapped  
defineCustomElements(window);
```

```
ionic serve
```

Intégration avec React.

Pour créer un nouveau projet, exécutez la commande suivante:

```
ionic start myApp blank --type=react  
cd myApp
```

À partir d'ici, la commande globale permettra la création d'un projet React avec toutes les dépendances.

```
ionic serve
```

Intégration avec Vue.

Actuellement en version bêta.

[@ionic/vue](#) combine la base de l'outillage et les API qui sont adaptées à VueJS.

Préparer le “build” et le déploiement

\$ ionic build ou ionic cordova build <platform> [options]

Exemples :

```
$ ionic cordova build ios
$ ionic cordova build ios --prod --release
$ ionic cordova build ios --device --prod --release --
--developmentTeam="ABCD" --codeSignIdentity="iPhone
Developer" --provisioningProfile="UUID"
$ ionic cordova build android
$ ionic cordova build android --prod --release -- --
--keystore=filename.keystore --alias=myalias
$ ionic cordova build android --prod --release -- --
--minSdkVersion=21
$ ionic cordova build android --prod --release -- --
--gradleArg=-PcdvBuildMultipleApks=true
```

Personnalisation ciblée de la plateforme (IOS/Android).

Le service **Platform** peut être utilisé pour obtenir des informations sur votre appareil actuel.

Vous pouvez obtenir toutes les plates-formes associées avec l'appareil à l'aide de plates-formes de la méthode, y compris si l'application est en cours de visualisation à partir d'une tablette, si c'est sur un appareil mobile ou un navigateur, et la plate-forme (iOS, Android, etc).

Vous pouvez également obtenir de l'orientation de l'appareil, si elle utilise de droite à gauche (sens de la traduction, et beaucoup plus. Avec cette information, vous pouvez entièrement personnaliser votre application pour s'adapter à n'importe quel appareil.

```
import { Platform } from 'ionic-angular';

@Component({...})
export MyPage {
  constructor(public platform: Platform) {

  }
}
```

Automatiser la création des icônes et écrans de démarrage.

`$ ionic cordova resources`

Ionic peut générer automatiquement des icônes et écrans de démarrage à partir de la source des images (.png, .psd ou .l'ia) pour vos plates-formes.

La source de l'image pour les icônes, idéalement, devrait être d'au moins **1024×1024px** et situés à des **ressources/icon.png**.

La source de l'image pour les écrans de démarrage, idéalement, devrait être au moins de **2 732×2732px** et situé à **ressources/splash.png**. Si vous avez utilisé **ionic start**, il devrait déjà être présent par défaut.

```
ionic cordova resources
```

Présentation des services de la “Ionic Platform”.

- [Ionic Deploy](#)

Permet à vous et votre équipe pour envoyer des modifications du code en direct directement à vos utilisateurs, lorsque vous êtes prêt, sans attendre en ligne au Magasin d'applications de révision et d'approbation.

- [Ionic Package](#)

Créer des applications natives dans le cloud avec Ionic Paquet pour obtenir à partir du code à l'app store avec de la plate-forme de dépendances ou d'complexité étapes de génération.

- [Ionic Monitor](#)

Ionic Monitor est la seule solution qui identifie les erreurs de la couche web, avec des sourcemaps tout le chemin vers le fichier d'enregistrement.

“Build” service de compilation.

Il est nécessaire de créer un compte **Ionic Pro**.

Ionic Pro Package rend facile de construire des binaires pour iOS et Android dans le cloud. Parfait pour l'automatisation des binaires et pour les développeurs, à l'aide de Windows qui veulent construire des applications iOS.

Prise en main

Pour commencer, vous aurez besoin de [télécharger les certificats de profils](#) pour iOS et/ou Android activer le Paquet pour construire votre application dans le cloud.

Cliquez sur l'onglet Code de votre Application, vous devriez être à vos constructions liste. Cliquez sur le bouton de Package pour la construction, vous souhaitez obtenir un binaire Natif de.

Dans le Package GUI, vous serez en mesure de choisir le type de construction que vous souhaitez exécuter, et le Cert que vous souhaitez utiliser.

Cycle de déploiement continu.

La fonctionnalité **Ionic Pro's Live Deploy** permet de mettre à jour l'INTERFACE utilisateur et la logique métier de votre application à distance, en temps réel.

Pousser HTML, JS, CSS et mises à jour directement à vos utilisateurs sans passer par l'app store pour corriger rapidement les bugs et de les expédier de nouvelles fonctionnalités.

Cela vous permet:

- Mise à jour de votre demande sur demande
- Éviter l'app store d'édition pour de nombreux correctifs
- Obtenir de nouvelles fonctionnalités et corrections de bugs à vos utilisateurs rapidement
- Test A/B, des changements avec certains utilisateurs avant de lancer à tout le monde



Merci d'avoir participer à cette formation.

