Développement d’applications

Historique

HTML 5, CSS3 et JS

HTML

(HyperText Markup Language) :

Il a fait son apparition dès 1991 lors du lancement du Web. Son rôle est de gérer et organiser le contenu.

C'est donc en HTML que vous écrirez ce qui doit être affiché sur une page web : du texte, des liens, des images etc..

- **HTML 1** : Première version créée par Tim Berners-Lee en 1991.
- **HTML 2** : Deuxième version du HTML apparu en 1994 et prend fin en 1996 avec l'apparition du HTML 3.0. C'est cette version qui posera en fait les bases des versions suivantes du HTML. Les règles et le fonctionnement de cette version sont donnés par le W3C (tandis que la première version avait été créée par un seul homme).
- HTML 3 : apparue en 1996, cette nouvelle version du HTML rajoute de nombreuses possibilités au langage comme les tableaux, les applets, les scripts, le positionnement du texte autour des images, etc.
- **HTML 4** : cette version aura été utilisée un long moment durant les années 2000. Elle apparaît pour la première fois en 1998 et propose l'utilisation de frames (qui découpent une page web en plusieurs parties), des tableaux plus complexes, des améliorations sur les formulaires, etc. Mais surtout, cette version permet pour la première fois d'exploiter des feuilles de style, notre fameux CSS !
- HTML 5 : c'est LA dernière version. De plus en plus répandue, elle fait beaucoup parler d'elle car elle apporte de nombreuses améliorations comme la possibilité d'inclure facilement des vidéos, un meilleur agencement du contenu, de nouvelles fonctionnalités pour les formulaires, etc. C'est cette version que nous allons découvrir ensemble.

CSS

(Cascading Style Sheets, aussi appelées Feuilles de style) :

Son rôle est de gérer l'apparence de la page web (agencement, positionnement, décoration, couleurs, taille du texte…).

Ce langage est venu compléter le HTML en 1996.

- **CSS 1** : dès 1996, on dispose de la première version du CSS. Elle pose les bases de ce langage qui permet de présenter sa page web, comme les couleurs, les marges, les polices de caractères, etc.

- **CSS 2** : apparue en 1999 puis complétée par CSS 2.1, cette nouvelle version de CSS rajoute de nombreuses options. On peut désormais utiliser des techniques de positionnement très précises, qui nous permettent d'afficher des éléments où on le souhaite sur la page.
- **CSS 3** : c'est la dernière version, qui apporte des fonctionnalités particulièrement attendues comme les bordures arrondies, les dégradés, les ombres, etc.

W3C et WHATWG

- Le **W3C** (World Wide Web Consortium) est l'organisation qui s'occupe de standardiser le web. Elle réfléchi à l'évolution des standards tels que l'HTML et le CSS. Discute des bonnes pratiques à employer pour écrire son code HTML, ou encore de nouvelles balises qu'il serait intéressant d'ajouter.
- Le **WHATWG** (Web Hypertext Application Technology Working Group) est un second groupe au sein du W3C. Ce groupe est constitué principalement de développeurs des navigateurs tels que Mozilla, Opera ou Apple. L'approche est ici totalement différente puisque ce groupe est beaucoup plus ouvert et surtout a pour objectif d'accélérer la standardisation (ou du moins la mise en place de standards pour les navigateurs).

HTML 5

Nouvelles balises

- Un allégement du code
- certaines balises ont été simplifiées

Code HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <link rel="stylesheet" type="text/css" href="design.css" />
  <script type=”text/javascript” src=script.js"></script>
</head>
<body>
</body>
</html>
```

Code HMTL5

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="utf-8" />
  <link rel="stylesheet" href="design.css" />
  <script src="script.js"></script>
```

```
</head>
```

- **<header>** : Qui indique que l'élément est une en-tête
- **<footer>** : Qui indique que l'élément est un pied-de-page
- **<nav>** : Qui indique un élément de navigation tel qu'un menu
- **<aside>** : Qui correspond à une zone secondaire non liée au contenu principal de la page
- **<article>** : Qui représente une portion de la page qui garde un sens même séparée de l'ensemble de la page (comme un article de blog par exemple)
- **<video>**: Cette balise intègre directement un lecteur vidéo dans la page, avec des boutons Lecture, Pause, une barre de progression, du volume… Un vrai petit Youtube intégré à votre page et natif au navigateur !
- **<audio>**: Cette balise est l'équivalent de la balise video mais pour l'audio. En 3 lignes de code vous avez un lecteur MP3 !
- **<canvas>**: Cette balise est probablement la plus prometteuse de toutes, puisqu'il s'agit d'une surface sur laquelle il est possible de tracer des formes et de les animer. En résumé… C'est dans cette zone que sont réalisées des animations ou des jeux.

Exemple

```
<audio controls="controls">
   <source src="music.mp3" type="audio/mp3" />
   Votre navigateur n'est pas compatible
 </audio>
```

Les lignes de code ci-dessus sont très simple à comprendre.

- La balise <audio> indique au navigateur qu'il devra afficher un contenu audio.
- La deuxième ligne indique l'emplacement du fichier audio. Dans notre exemple, celui-ci se trouve à la racine de notre site web.
- Noter que la balise <source> se ferme seule (comme la balise meta, ou la balise img pour les images par exemple).
- Et enfin, à la dernière et quatrième ligne nous refermons la balise <audio> avec </audio>.

Il est possible de spécifier de toutes nouvelles valeurs dans le champ "type" des balises input, afin d'indiquer le type de contenu du champ. C'est particulièrement utile afin d'effectuer une première validation du contenu avant l'envoi des informations au serveur.

C'est également très important pour la navigation depuis un smartphone, qui affichera un clavier adapté selon le type de contenu.

Par exemple, pour le champ input suivant :

```
<input type="tel«  id=« tel2 » />
```

On aura un clavier sur un iPhone plus adapté

Par exemple, pour le champ input suivant :

```
<input type=« date«  id=« date1 » />
```

On aura un contrôle calendrier sur certains navigateurs et sur les plus anciens un champ texte standard

Amélioration sémantique

Les balises <DIV> remplacées par de nouveaux éléments

<article> <section>

Les éléments <article> et <section>

- L'élément <article> est un fragment indépendant du contenu général. Billet de blog, nouvelle article ou autres types de contenu du texte . Fondamentalement, vous pouvez utiliser cet élément pour le balisage d'un composant destiné à être largement utilisé et distribué.
- L'élément <section> est assez trompeur car il est largement trop employé par les développeurs Web comme une alternative à . Vous devriez savoir que cette balise est étroitement liée à la balise <article> et qu'elle est utilisé pour regrouper un contenu qui diffère d'un autre groupement de contenu sur la page. Généralement, les groupes sont fait par thèmes ou sujet identique.

<header> <footer>

- L'élément <header> a été créé pour une présentation plus sémantique (les systèmes de signes ?) des outils de navigation et des données importantes placées dans l'en-tête de la page Web.
- La balise <footer> est similaire à <header>, elle est utilisé pour créer la structure de pied de page de votre document web. Vous pouvez également utiliser cet élément plusieurs fois sur une seule page pour des blocs différents. Cette balise peut être utilisée, par exemple, pour marquer le droit d'auteur, Conditions d'utilisation et autres, mais aussi pour le marquage de certaines informations sur l'auteur de l'article …

<nav>

La balise <nav>

L'élément <nav> est utilisé pour créer des menus avec des liens vous permettant de naviguer sur les pages du site Web. Par exemple :

- un bloc avec des liens sponsorisés
- un bloc pour les différentes catégories
- Etc.

N'oubliez pas que l'élément <nav> peut également être utilisé plusieurs fois.

Bien souvent cette balise est associée à une liste, ordonnée ou non, de liens : <ul> et <ol>.

<figure> <figcaption>

Les éléments <figure> et <figcaption>

Selon les références du W3C,les éléments <figure> et <figcaption> sont utilisés pour présenter un bloc de contenu avec une légende, qui est généralement référencé comme une seule unité à partir du flux principal du document.

En d'autres termes, vous pouvez l'utiliser pour marquer divers types de supports de contenu comme des illustrations, des photos, des exemples de code et des diagrammes.

La balise <figure> définit un contenu autonome, comme des illustrations, des diagrammes, des photos, des listes de codes, etc

La balise <figcaption> définit une légende pour un élément <figure>.

exemple

```
<figure>
  <img src="bx1.jpeg" alt="Bordeaux 1" style="width:50%">
  <figcaption>Université de Bordeaux</figcaption>
</figure>
```

<aside>

L'élément <aside> est utilisé pour un contenu secondaire qui n'est pas emboîtée dans aucun élément du document (<article> ...).

L'exemple le plus approprié pour utiliser l'élément <aside> est la fameuse Sidebar ou colonne latéral.

Nous pouvons utiliser cet élément avec l'éléments <nav>, les bannières publicitaires ou tout simplement pour le contenu qui doit être placés séparément du contenu principal.

Nouveaux éléments de formulaire

<datalist>

```
<input list="navigateur" />
<datalist id="navigateur">
        <option value="Internet Explorer">
        <option value="Firefox">
        <option value="Chrome">
        <option value="Opera">
        <option value="Safari">
 </datalist>
```

L'élément <datalist> spécifie une liste pré-définis d'options pour un élément <input>.

L'élément <datalist> est utilisé pour fournir une « saisie semi-automatique » sur les éléments <input>. Utilisé l'attribut id de la balise <input> et l'attribut list de la balise <datalist>pour les relier ensemble.

<keygen>

```
<form action="demo_keygen.php" method="get">
   Username: <input type="text" name="nom_utilisateur" />
   Encryption: <keygen name="security" />
   <input type="submit" />
</form>
```

Le but de l'élément <keygen> est de fournir un moyen sécurisé pour authentifier les utilisateurs.

La balise <keygen> spécifie un champ générateur de clés dans un formulaire.

Lorsque le formulaire est soumis, deux clés sont générées, l'une publique et l'autre privée.

<output>

```
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">0
   <input type="range" name="a" value="50" />100
   +<input type="number" name="b" value="50" />
   =<output name="x" for="a b"></output>
 </form>
```

L'élément <output> représente le résultat d'un calcul (réalisé par un script par exemple).

Hors ligne

Votre site web HTML5 accessible Hors-ligne grâce au cache – Fichier manifest

A quoi sert le cache d'un navigateur ?

- Naviguer sur un site ou une page web sans connexion internet (après l'avoir déjà visité ou téléchargé et stocké en mémoire dans le cache)
- Optimiser la vitesse de chargement des pages puisque les fichiers sont présents en local
- Réduire la charge du serveur, il ne transmet que les fichiers qui ont changé depuis la dernière visite

cache manifest

```
CACHE MANIFEST

# version 0.1
```

```
 index.html
style.css
 script.js
```

Les lignes de code ci-dessus sont suffisamment claire pour ne pas être détaillée. A Noter : Il est possible d'ajouter d'autres options à notre fichier manifest :

- CACHE, : liste les fichiers à mettre en cache;
- NETWORK : liste les fichiers qui nécessitent OBLIGATOIREMENT une connexion internet;
- FALLBACK : liste les fichiers qui, s'ils ne sont pas accessibles en ligne, renvoient vers d'autres fichiers

Relier le manifest à votre site

Définissez le type MIME du manifest par exemple grâce au fichier .htaccess

Avec un fichier htaccess. Il faut déclarer le MIME-type du fichier manifest :

```
AddType text/cache-manifest manifest
```

Cela servira à tous les fichiers avec une extension manifest. Ils auront pour ont pour MIME-type : text/cache-manifest.

Relier le manifest à votre site avec la balise <html>

Pour utiliser le fichier manifest du cache, il suffit d'ajouter une propriété à l'élément <html> :

```
<!DOCTYPE html>
<html lang="fr" manifest="site.manifest">
              <meta charset='utf-8'>
```

Rafraichir le cache

Il est important de noter que, même lorsque l'utilisateur est en ligne, le navigateur demandera seulement au serveur les nouveaux contenus dans trois cas:

- L'utilisateur efface la mémoire cache (et évidemment la suppression de votre contenu).
- Si des modifications ont été apportées aux fichiers manifest.
- Le cache est mis à jour via le JavaScript

Alors, pour forcer tous les utilisateurs a recharger leur cache, vous pouvez changer quelque chose dans le fichier manifest. La plupart du temps, vous aurez probablement juste a changer un commentaire, et ce sera suffisant.

Javascript

Historique

- Version 0 : JavaScript a été créé en 1995 par Brendan Eich.
- Version 1.0 : Première release intégrée à Mosaic en Mars 1996.

- Version 1.2 : Il a été standardisé sous le nom d'ECMAScript en juin 1997 par Ecma International dans le standard ECMA-262.
- Version 1.3 : Octobre 1998. JavaScript n'est depuis qu'une implémentation d'ECMAScript, celle mise en œuvre par la fondation Mozilla. L'implémentation d'ECMAScript par Microsoft (dans Internet Explorer jusqu'à sa version 9) se nomme JScript, tandis que celle d'Adobe Systems se nomme ActionScript.
- ECMAScript 8 : c'est la dernière version publiée en Juin 2017.

Définition

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs.

C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés.

En outre, les fonctions sont des objets de première classe. Le langage supporte le paradigme objet, impératif et fonctionnel.

Javascript - Client

Le code Javascript qui se trouve dans des pages Web, sert généralement à dire comment la page Web doit réagir. Si la page Web contient du code Javascript, le navigateur lit le code Javascript et suit les instructions du code.

Généralement le code Javascript dans une page Web sert à :

- Faire bouger, apparaitre ou disparaitre des éléments de la page (un titre, un menu, un paragraphe, une image…).
- Mettre à jour des éléments de la page sans recharger la page (changer le texte, recalculer un nombre, etc).
- Demander au serveur un nouveau bout de page et l’insérer dans la page en cours, sans la recharger.
- Attendre que l’utilisateur face quelque chose (cliquer, taper au clavier, bouger la souris…) et réagir (faire une des opérations ci-dessus suite à cette action).

Le code Javascript sert donc à donner du dynamisme à la page. Sans lui, la page ressemble à une page de livre, un peu animée (grâce à un autre langage appelé le CSS), mais qui ne change pas beaucoup.

Javascript - Client classique

Pour intégrer du code Javascript dans une page HTML il faut l’inclure entre les balises <script> </script>

```
Les commentaires javascripts sont matérialisé soit par
```

```
// pour commenter une ligne
soit par
/* …. */
lorsque l'on veut commenter plusieurs lignes
```

Les déclarations locales se font à l'aide du mot clef var

Il est possible d'utiliser des bibliothèques externes à l'aide de la syntaxe ci-dessous :

```
<script type="text/javascript" src="monscript.js"></script>
```

Il est possible de manipuler les éléments d'une page HTML en utilisant les objets implicites du modèle DOM :

- Document
- Forms
- Elements
- .....

Pour créer des objets on peut utiliser la syntaxe ci-dessous :

```
var Bouteille= function(cepage,château,annee) {
this.cepage = cepage;
    this.chateau = chateau;
    this.annee = annee;
  }
var bouteille = new Bouteille(« cabernet », « saint julien », « 1987 »  );
```

Pour debugger on peut utiliser soit le debugger du navigateur soit la console :

```
console.log(« erreur »);
```

Javascript - Client moderne

Introduite avec ECMAScript 2015 la notation class permet de simplifier l'écriture du code Javascript.

```
class Bouteille {
  constructor(cepage, chateau, annee) {
    this.cepage = cepage;
    this.chateau = chateau;
    this.annee = annee;
  }
}
```

Il est ensuite possible d'ajouter des méthodes dans la classe comme dans tous les langages Objet.

Pour instancier un objet on utilise le mot-clef new

```
const bouteille = new Bouteille(« sauvignon », « saint leon »,   « 2019 »);
```

Il est aussi possible de faire de l'héritage à l'aide du mot-clef extends

```
class Magnum extends Bouteille {
……
}
```

En plus de la programmation objet standard, Javascript manipule nativement le format de données JSON à l'aide de l'objet JSON.

```
var obj = JSON.parse(<chaine representant l'objet>);
```

Il est donc ainsi facile de créér un objet javascript à partir de sa description JSON.

Il est aussi possible de modifier l'arbre DOM directement pour cela on peut utiliser :

```
document.getElementById() : pour récupérer un objet par son id
document.getElementsByTagName() : pour récupérer un objet par son nom
document.createElement(<type d'élément>) : pour créer un élément DOM celui-
ci devra ensuite être ajouté à
```

l'arbre DOM à l'aide de la méthode appendChild

```
document.createTextNode
```

pour créer un nœud texte à afficher

Pour une complète référence vous pouvez consulter la documentation de référence :

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference

Javascript - Client subtilités

Le code Javascript est lu et exécuté sur le navigateur Web, donc sur l'ordinateur de l'internaute.

C'est ce qu'on appelle du code "côté client". Et il ne peut pas interagir directement avec le code "côté serveur" (celui qui a produit la page Web); et donc on ne peut pas communiquer directement entre le Javascript côté client et PHP côté serveur par exemple.

Comme l'internaute a le contrôle de sa machine, il peut choisir de désactiver le support de Javascript sur son navigateur. Dans ce cas, ce dernier ignorera le code Javascript et fera comme si il n'était pas là. Il verra la page, mais tous les éléments qui fonctionnent avec Javascript ne marcheront pas.

hors d'un navigateur Web

Avec l'amélioration des performances de Javascript, le langage a été de plus en plus utilisé en dehors du navigateur Web. On le retrouve aujourd'hui un peu partout :

- Sur les serveurs, l'exemple le plus célèbre étant NodeJS, un outil qui permet de générer les pages Web avant de les envoyer au navigateur.

- Sur les interfaces des ordinateurs, il permet d'afficher des fenêtres et des boutons (Scripting QT, Gnome Shell).
- Sur les téléphones, pour le moment sur FirefoxOS et PhoneGap, il permet d'écrire des applications.

Javascript moderne

Aujourd'hui le langage Javascript est massivement utilisé sur tous les sites Web grand public. Des outils, tel que jQuery, on été développés pour faciliter son utilisation et les navigateurs sont devenus beaucoup plus efficaces dans son traitement.

La popularité grandissante dans les années 2000 d'un outil appelé AJAX, qui permet à Javascript de mettre à jour une page sans la recharger, a propulsé le langage sur le devant de la scène. L'utilisation d'AJAX rendant, si il elle est bien faite, la consultation du site plus rapide et fluide, on l'a rapidement retrouvé sur tous les sites les plus fréquentés.

Javascript permet aujourd'hui, avec le support d'autres technologies (Flash, HTML5, canvas, CSS3, WebGL...), de faire des choses très évoluées comme de la 3D, de la manipulation d'images, de sons et de videos. Bientôt, avec des innovations comme WebRTC, Javascript permettra de faire du Peer-to-peer et de la vision conférence à l'intérieur du navigateur Web.

Local Storage

Le Local Storage est une manière élégante de stocker dans le navigateur des informations facilement. Par exemple, pour écrire puis lire une valeur dans le Local Storage il suffit d'écrire :

```
localStorage.setItem("name", « ouattara");
    alert(localStorage.getItem("name"));
```

La variable sera toujours disponible si l'utilisateur ferme puis ré-ouvre son navigateur.

L'utilisation du Local Storage est proche de celle des cookies, mais contrairement aux cookies, ces informations ne sont jamais communiquées au serveur.

Elles sont ainsi particulièrement adaptées aux applications offline.

Géolocalisation

Il ne s'agit pas strictement d'une spécification de l'HTML5, mais elle y est souvent associée.

Il est possible grâce à l'API de géolocalisation d'accéder aux coordonnées de l'utilisateur si celui-ci a accepté de partager sa position via le bandeau s'affichant en haut de page :

On peut alors très facilement disposer d'informations telles que :

- La latitude, la longitude, et l'altitude de l'utilisateur
- Son orientation par rapport au Nord
- La vitesse à laquelle il se déplace

Drag and Drop

Il est également possible d'effectuer des "glisser-déposer" dans une page web.

Par exemple, si l'on souhaite pouvoir déplacer un élément d'une liste, il suffit de déclarer cet élément avec l'attribut "draggable" à true :

```
<li draggable="true">Element de ma liste</li>
```

Définir une zone potentielle de destination :

```
<div ondrop="drop(this, event)"
```

Puis ensuite effectuer le traitement Javascript correspondant:

```
function drop(target, e) {
  // Traitement
}
```

Notons que l'utilisation de bibliothèques telles que jQuery est préférable à l'injection de code Javascript dans l'HTML.

Web Sockets

Il s'agit là d'un nouveau protocole ambitieux de communication avec le serveur. Un navigateur ne peut habituellement qu'effectuer des requêtes au serveur puis recevoir sa réponse.

C'est une communication unidirectionnelle (dite par canal simplex). Les Web Sockets apportent la communication bi-directionnelle (dite full-duplex) entre le client et le serveur.

Pour prendre un exemple un peu plus concret, demandez-vous comment vous réaliseriez une page dont le contenu devrait toujours être à jour (comme un chat par exemple) ?

Frameworks

Avec la multiplication des développements Javascript, particulièrement portée par l'avénement des applications Web, de nombreux framework ont vu le jour.

- Angular
- EmberJS
- NodeJS
- React
- ExtJS
- Jquery
- AngularJS
- ....

React

React (aussi appelé React.js ou ReactJS) est une bibliothèque JavaScript libre développée par Facebook depuis 2011. Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page (ou portion) HTML à chaque changement d'état.

React est une bibliothèque qui ne gère que l'interface de l'application, considéré comme la vue dans le modèle MVC.

React est une bibliothèque permettant de développer des composants autonomes à faire interagir pour développer des applications.

Ceux-ci sont réutilisables et peuvent être utilisés dans des applications développées à l'aide d'autres frameworks.

React utilise le principe: « les données descendent, l'état remonte ». Donc lorsque l'état d'un composant évolue il remonte l'information à son parent et ainsi de suite jusqu'à arriver au composant « père » de tous les composants intéressés par cette évolution.

React ne manipule pas directement le DOM mais utilise un DOM Virtuel de la page à manipuler.

C'est le « développeur » qui décide quand mettre en cohérence ce DOM Virtuel avec le DOM réel.

Cette approche améliore les performances mais permet théoriquement d'utiliser React en dehors de HTML (à condition d'utiliser un modèle document).

Afin d'éviter de surcharger le code source des pages avec des createElement, React peut être utilisé avec l'extension JSX de Javascript

Angular

Angular (communément appelé "Angular 2+" ou "Angular v2 et plus")2,3 est un cadriciel (framework) coté client open source basé sur TypeScript dirigée par l'équipe du projet Angular à Google et par une communauté de particuliers et de sociétés. Angular est une réécriture complète de AngularJS, cadriciel construit par la même équipe.

Celle-ci permet de structurer son code avec à la manière de XML.

Angular est une réécriture complète d'AngularJS pour le rendre plus modulaire.

Angular n'a pas de notion de "portée" ou de contrôleurs, mais utilise une hiérarchie de composants comme principale caractéristique architecturale.

L'utilisation de TypeScript est induite par Angular.

Typescript est un sur-ensemble de ECMAScript6 et par extension de Javascript (ECMAScript5).

NodeJS

La majorité des frameworks utilise des outils tournant sous nodeJS.

NodeJS est une plate-forme d'exécution Javascript serveur (i.e hors d'un navigateur).

Cette plate-forme modulaire et extensible se rapproche d'un serveur d'application dans sa philosophie.

Parmi les services « généraux » fourni on peut notre un serveur http ou un gestionnaire de paquet.

Installation de NodeJS

Le moyen le plus simple d'installer NodeJS est de récupérer l'archive et de la dézipper.

Pour pouvoir aussi utiliser le gestionnaire de paquet npm, il vous faudra également installer git.

Pour plus de facilité il est recommandé d'ajouter les répertoires d'installation au PATH.

Installation d'Angular

Une fois l'environnement NodeJS d'installé il suffit d'utiliser le gestionnaire de paquet npm pour installer Angular.

```
    npm install –g @angular/cli
```

Cela installe l'ensemble des dépendances nécessaires à Angular dans votre environnement Node.

Afin de prendre en main la solution se reporter à l'excellente documentation en ligne :

```
https://angular.io/guide/quickstart
```

Installation de ReactJS

Comme Angular React est un ensemble de composant s'exécutant sous Node.

Ainsi pour créer son application il suffit d'utiliser :

```
npx create-react-app
```

Pour lancer le serveur de test il suffit d'aller dans le répertoire créé et de faire

```
npm start
```