MEAN

- Stack de développement javascript composée :
  - de 2 frameworks
    - ExpressJS : MVC côté serveur tournant sur NodeJS.
    - AngularJS : MVVM coté navigateur
  - d'une base de données orientée documents.
    - MongoDB

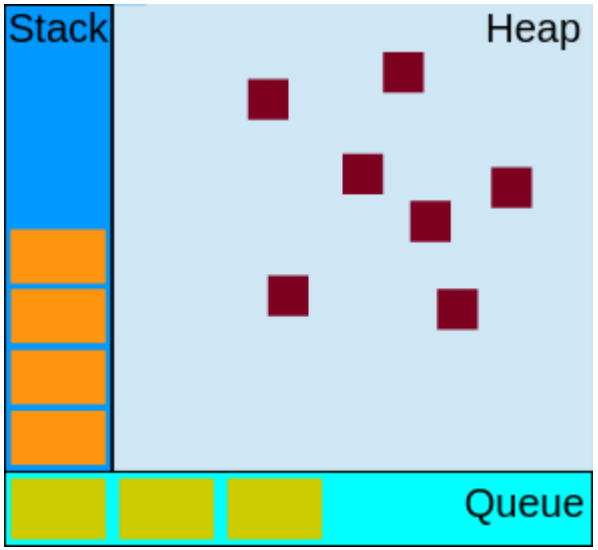
Javascript : Rappel - 1

Langage

- interprété
- orienté objet (héritage prototypal)
- fonctionnel
- faiblement typé
- Mono thread
- **Asynchrone, non bloquant** : une opération asynchrone (typiquement lorsque elle-ci consiste à travailler sur des données qui se trouvent en dehors de l'environnement d'exécution: dans des fichiers, dans une base de données, récupérées depuis un serveur distant etc.), rend immédiatement la main après qu'elle ait été déclenchée. La fonction appelante poursuit alors sont exécution sans tenir compte du résultat de l'opération.

Javascript : Rappel - 2

https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop



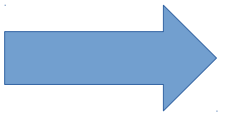
- La Stack représente la pile des appels de fonction en cours. La fonction qui se trouve au sommet de la pile est la fonction en cours d'exécution.
- La Queue représente la file des fonctions de rappel en attente d'exécution.
- La Heap (ou tas) représente l'espace mémoire ou sont stoquées les varaibles locales aux fonctions. En javascript, **chaque appel de fonction** créé un scope (on parle également de **closure**)**.** Lorsque la fonction rend la main, le scope qui lui a été associé est détruit, sauf lorsque ses variables sont référencées par des fonctions en attente d'execution.

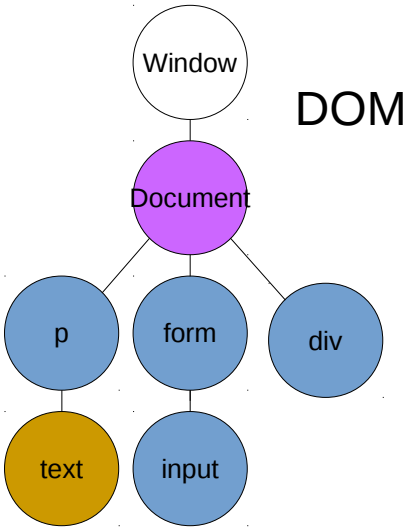
La Stack croit à mesure que les appels de fonctions s'imbriquent et se vide lorsque les fonctions appelées rendent successivement la main. Lorsque la Stack est vide (la première fonction appelante de la série rend la main), la fonction qui se trouve en tête de file est extraite pour être empilée dans la Stack. Et le cycle recommence.

DOM - Rappel

HTML (texte)

```
<!Doctype>
	<head>
	</head>
	<body>
		<p>Hello</p>
		<form>
			<input type='input'>
		</form>
		<div></div>
		<script>
			document.querySelector('form').submit();
		</script>
	</body>
</html>
```

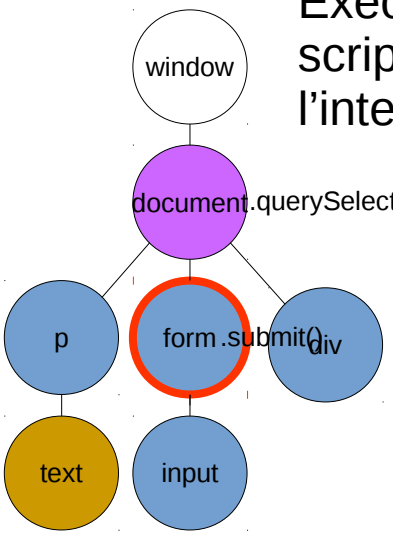




DOM (mémoire)

Le **moteur de rendu** du navigateur construit le DOM à partir de la page HTML et fait appel à l'interpréteur javascript qui exécute alors le code javascript. Le **moteur de rendu** expose à l'interpréteur javascript l'arbre DOM qu'il a construit.

Note : Le script de l'exemple précédent est « inliné », c'est à dire que son contenu apparaît intégralement dans le contenu de la page HTML. Un tel script s'exécute de manière synchrone, c'est à dire qu'il commence à s'exécuter lorsque le parseur HTML du moteur de rendu rencontre la balise <script>, et n'a accès qu'aux éléments du DOM qui l'ont précédé. Il en va autrement du chargement des scripts externes (référencés avec l'attribut src).



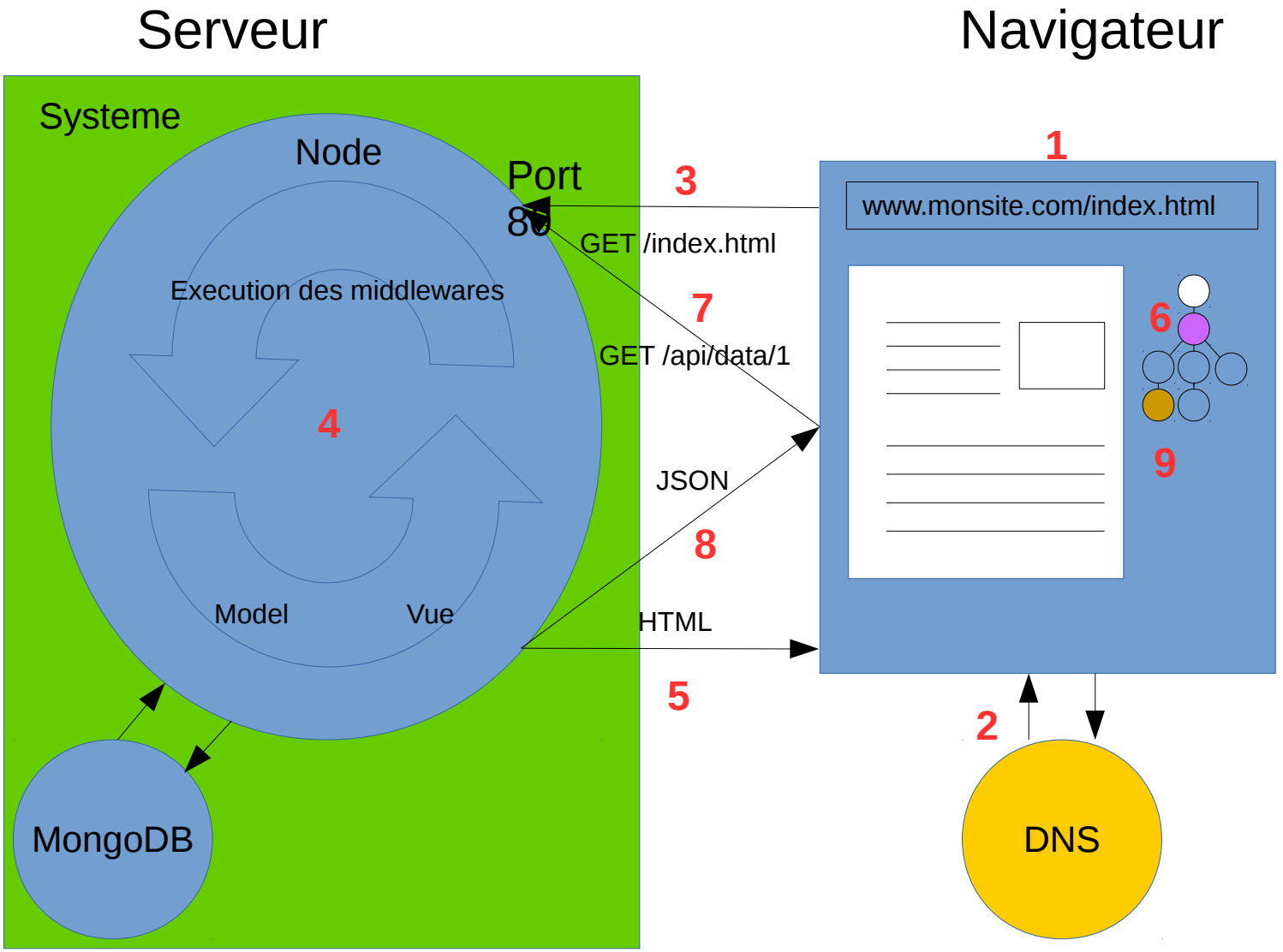
Exécution du script par l'interpréteur

Node

Le **navigateur** est doté d'un **interpréteur** JavaScript et d'un **moteur de rendu** qui lui fourni le DOM (représentation en mémoire de la page HTML sous la forme d'un arbre).

Coté serveur, cette faculté à exécuter du code javascript est apportée par **NodeJS**. Il est composé d'un **interpréteur** Javascript et d'un ensemble de **modules** qui lui permettent d'accéder aux ressources système (socket TCP, système de fichier, information OS etc ...).

Stack MEAN : scénario - 1



Stack MEAN : scénario - 2

- **1** saisie de l'URL dans la barre d'adresse
- **2** requête DNS de résolution de nom : traduit www.site.com en adresse IP
- **3** requête HTTP en direction du serveur dont l'adresse IP vient d'être récupérée
- **4** Exécution de l'application basée sur Expresse. Expresse utilise le module 'http' de node qui permet d'instancier un serveur web.

  On distingue 2 temps:
  - Étape d'initialisation : exécuté qu'une seule fois au lancement
    - Paramétrage d'expresse
    - Déclaration des middlewares
    - Déclaration des routes
  - Étape de fonctionnement : A chaque requête reçue
    - Execution à la chaîne des middlewares enregistrés lors de l'étape d'initialisation suivant l'ordre dans lequel ils ont été déclarés.
    - Routage
    - Exécution du contrôleur associé à la page demandée. Peut éventuellement s'appuyer sur une BDD.
    - Execution de la vue chargée de formater les données en HTML

  Cette architecture est communément appelée MVC (pour Model Vue Controler)

Stack MEAN : scénario - 3

- **5** retour de la page formatée en HTML au navigateur
- **6** construction du DOM et chargement d'AngularJS

  AngulaJS est un framework classé parmi les frameworks MVVM (Model Vue Vue Model). Cet acronyme met en évidence l'interaction et le couplage fort qu'il existe entre la couche Model et la Vue. Une modification du model (données) se répercute instantanément sur la vue (DOM) et inversement.

  Bien souvent, les applications Front-end vont inscrire les données persistantes ou vont aller chercher les données à afficher auprès d'un WEB service accédé via une API REST. Par conséquent, une partie de la configuration des routes et des contrôleurs d'express concernera cette API.
- **7** requête en direction de l'API Rest
- **8** retour JSON
- **9** mise à jour du DOM par Angular