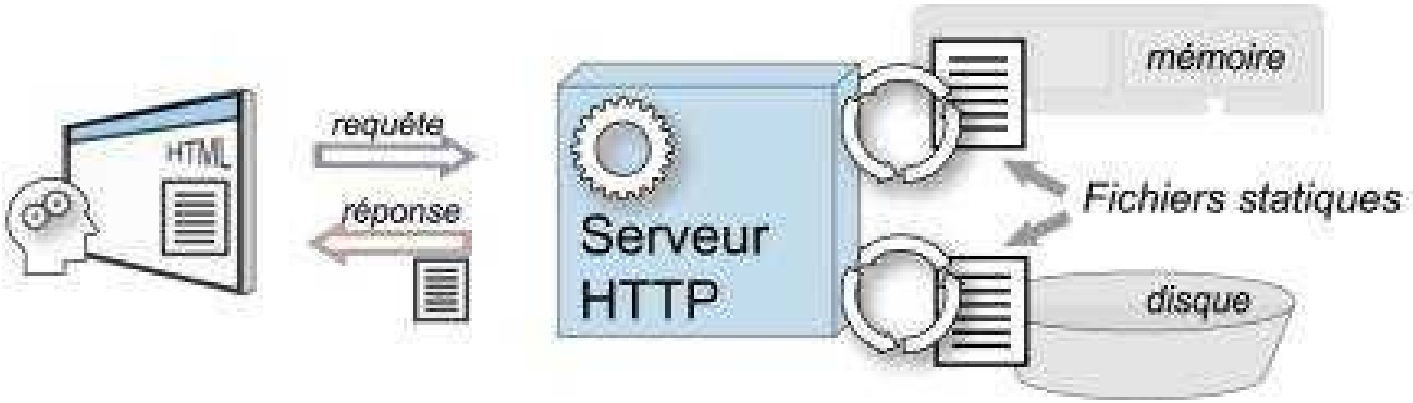
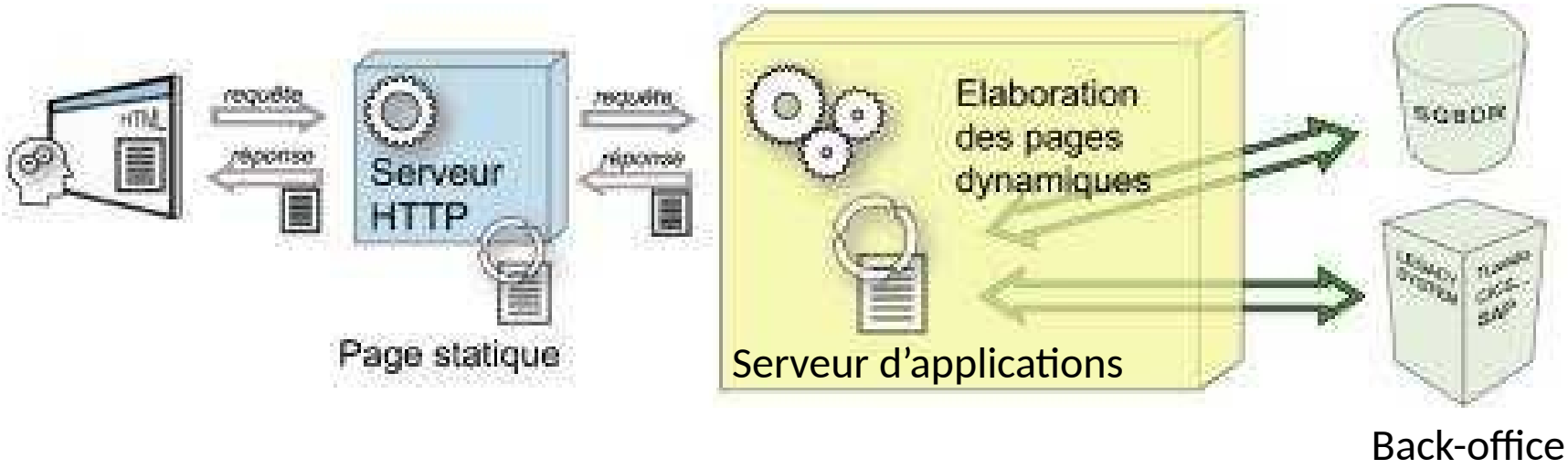
http://www.iquad.in/webapplications.html

Évolution des architectures d'Application Web

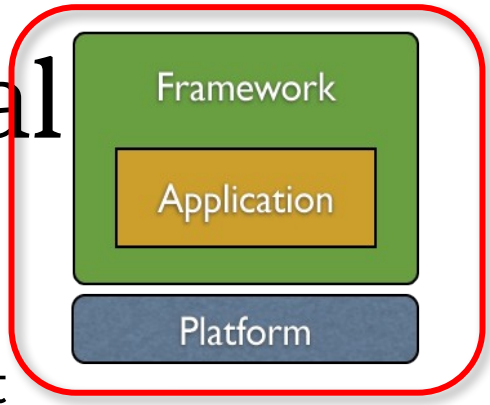
Architecture Web statique

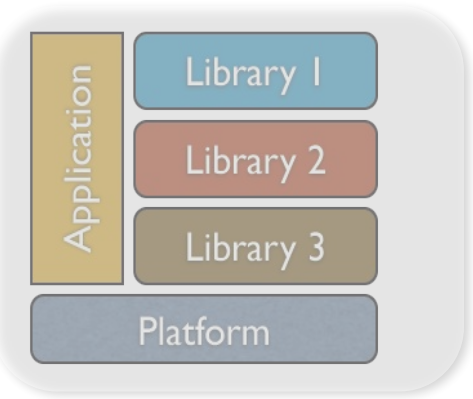


Architecture Web dynamique

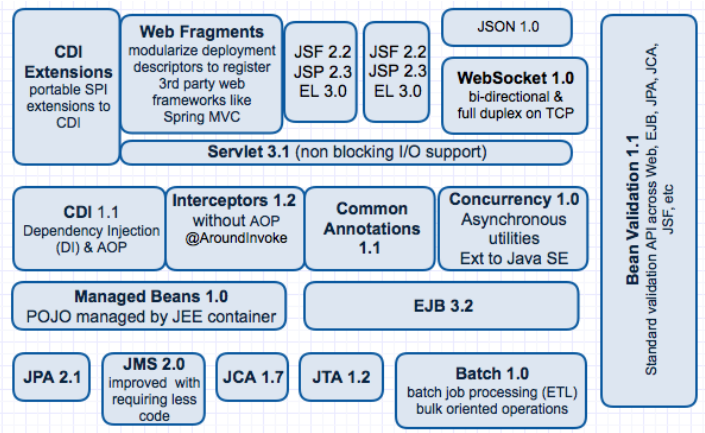


Cadre architectural



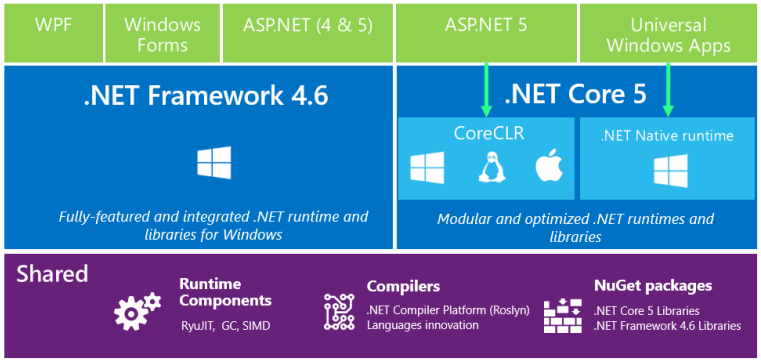


- Logiciel d'infrastructure offrant un contexte d'exécution isolé aux composants applicatifs



- La pile JEE (1995)
  - Multi plateforme
  - Mono langage (Java)

- La pile .NET (2002)
  - Mono plateforme (Windows)
  - Multi langage (VB, C#, C++ etc.)



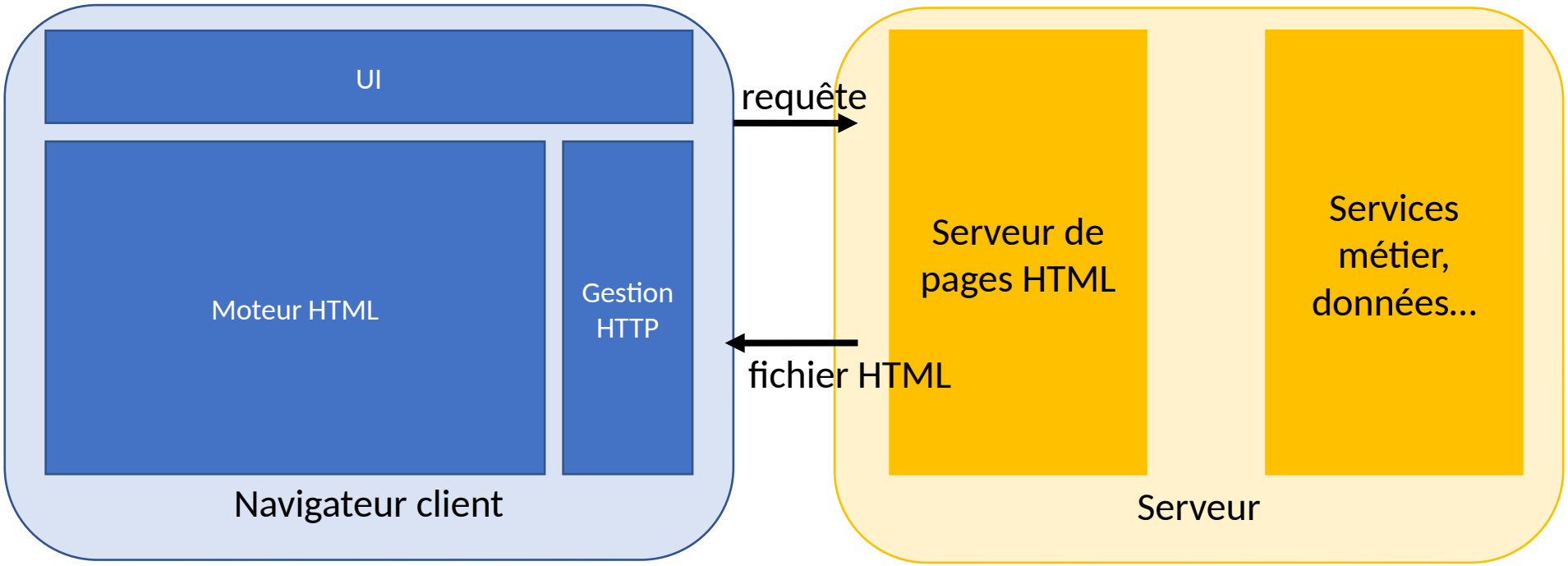
Bénéfices

- Évite l'utilisation de code « technique » dans le code applicatif / distinction entre la partie développement (métier) et la partie technique (administration)
- Principaux services fournis aux applications
  - accès aux BD
  - connexion aux applications d'entreprise
  - gestion des transactions, des sessions, des reprises sur panne
  - sécurité
  - passage à l'échelle
  - gestion par console d'administration unifiée
  - …
- Support du déploiement de composants

→ Pas important vers du code applicatif plus réutilisable !

L'application Web classique des années 90

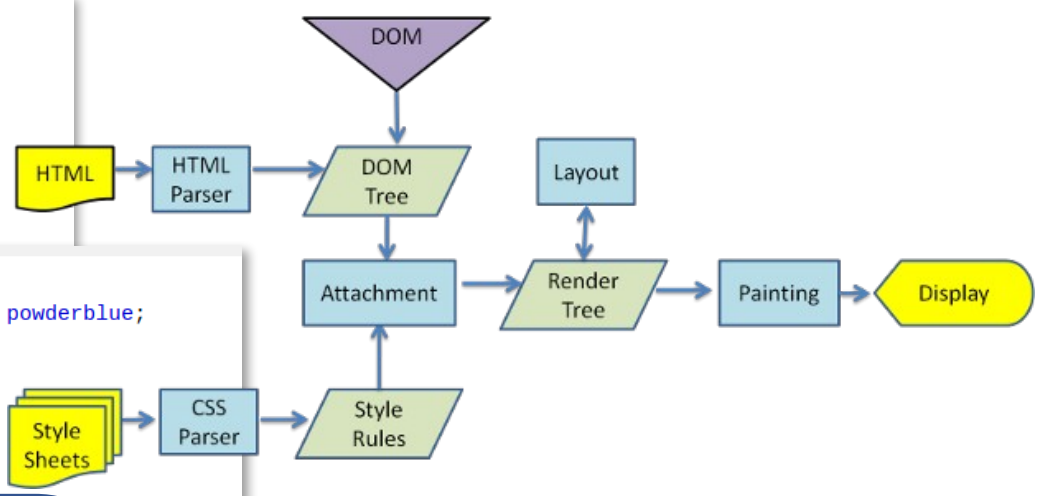
- Un client léger (navigateur rudimentaire)
- Un protocole simple et sans état
- Exécution essentiellement sur le serveur

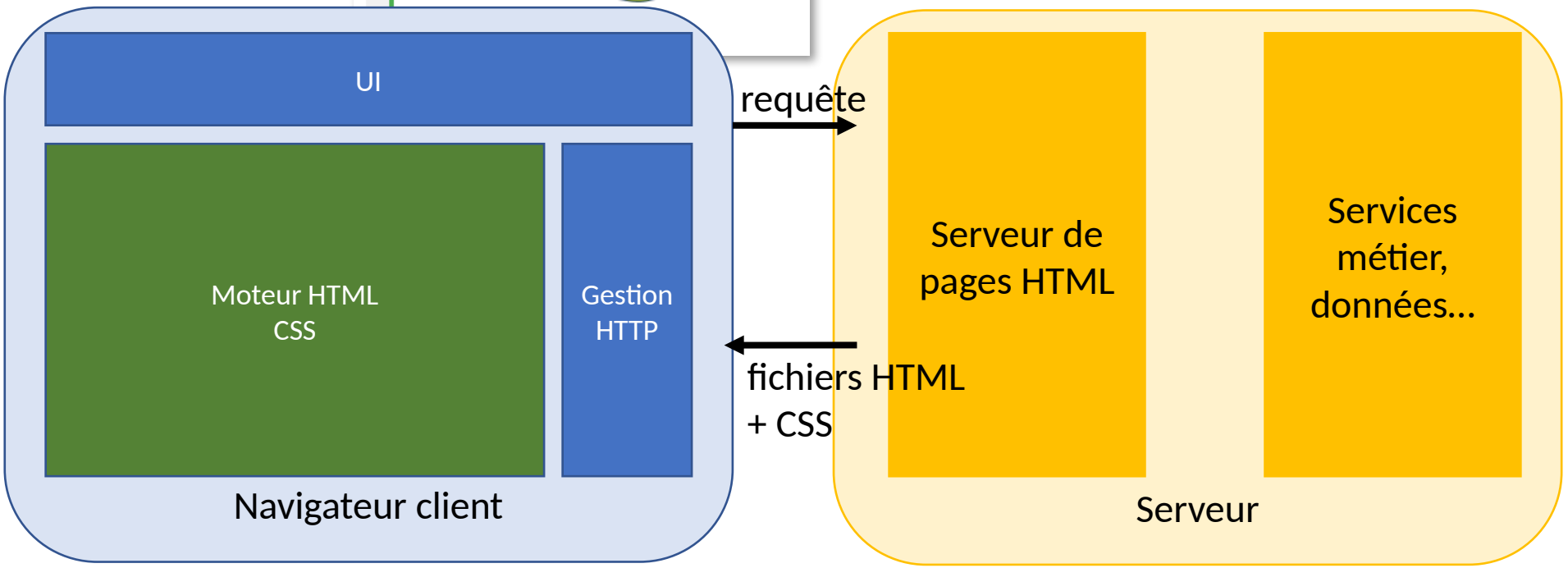


Mi-90's : une partie de la présentation est transférée au client

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```css
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
```



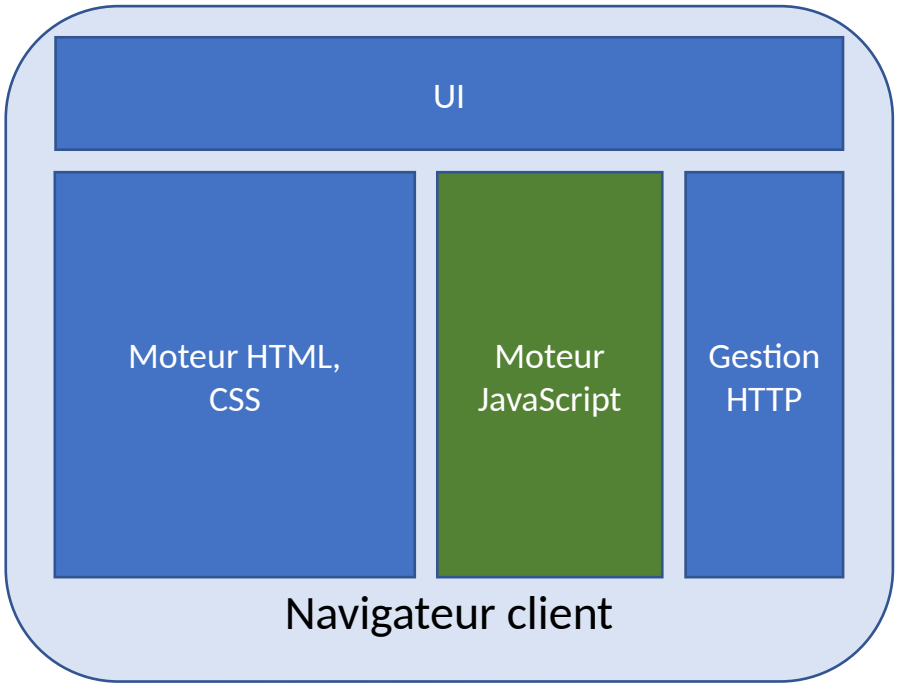


1996-97 : scripts interprétés par le client

- Tag <script> dans le HTML
- Interprété par le navigateur

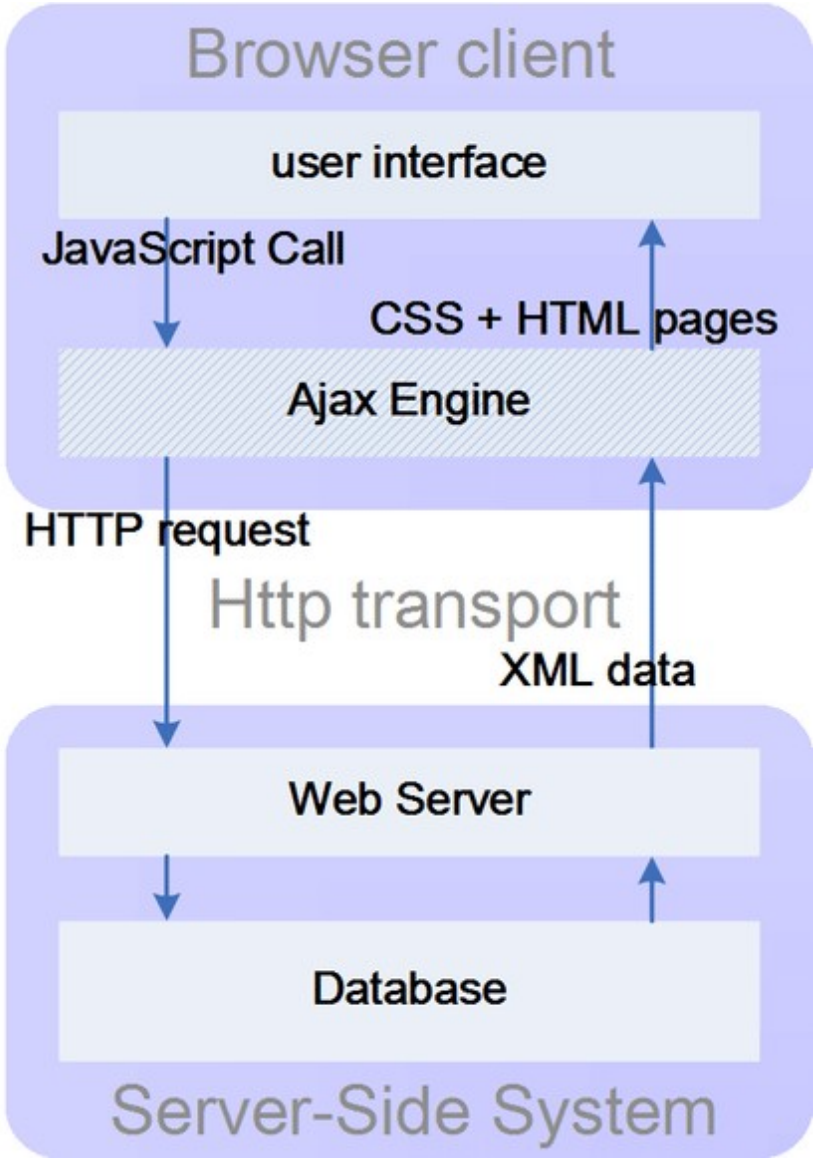
→ Dynamic HTML (DHTML)

```
<!DOCTYPE html>
<html>
<body>

<h1>My First JavaScript</h1>

<p>JavaScript can change the content of an HTML element:</p>

<button type="button" onclick="myFunction()">Click Me!</button>

<p id="demo">This is a demonstration.</p>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Hello JavaScript!";
}
</script>
```



2005 : patron « AJAX » : Vue et Contrôleur déportés sur le client

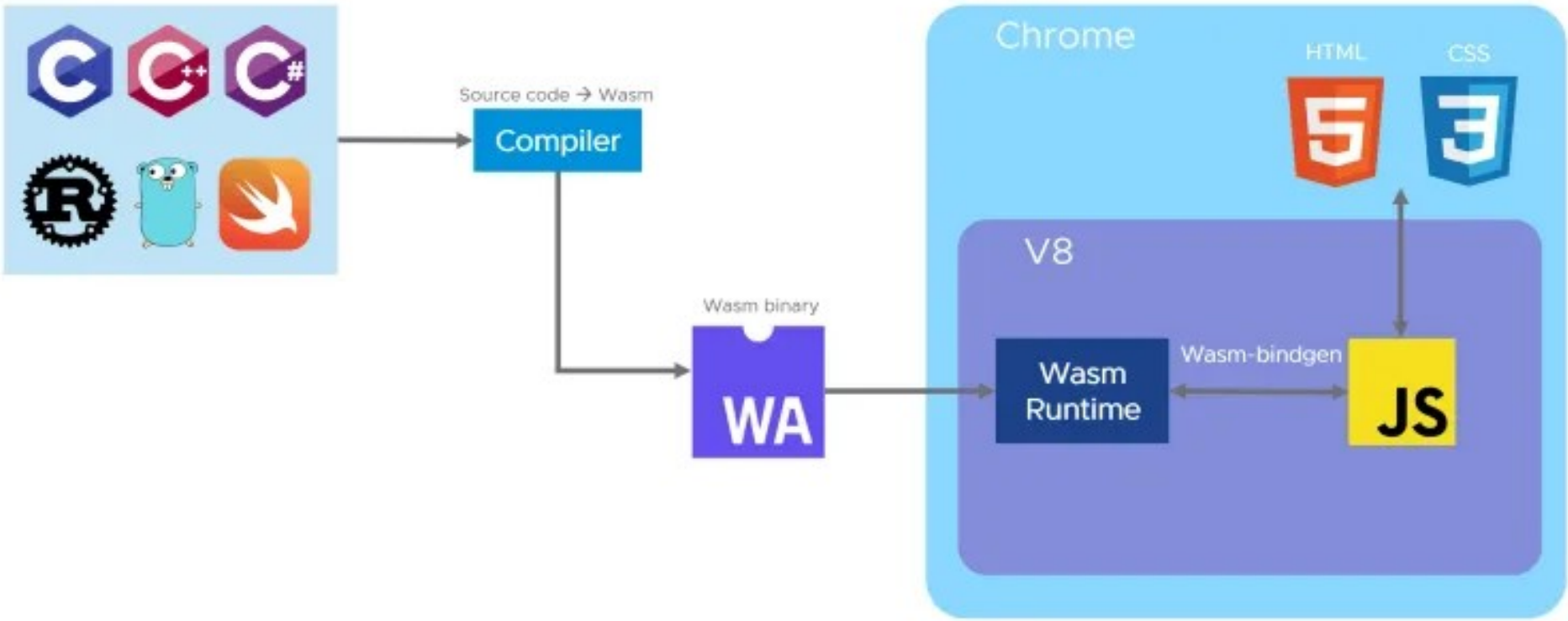
- *Asynchronous JavaScript + XML*
- Lorsqu'une action engendre la récupération de nouvelles données
  - La requête est faite au serveur en arrière-plan
  - A la réception, seule la portion de page impactée est rechargée
- Atouts
  - Moins de trafic et de charge serveur
  - Application sans effet de page
  - Possibilité d'applications très riches (par agrégation de services, traitements...)



https://www.researchgate.net/figure/The-comparison-between-a-classic-web-application-model-and-an-Ajax-web-application-model_fig3_241415122

2015 : Wasm (WebAssembly)

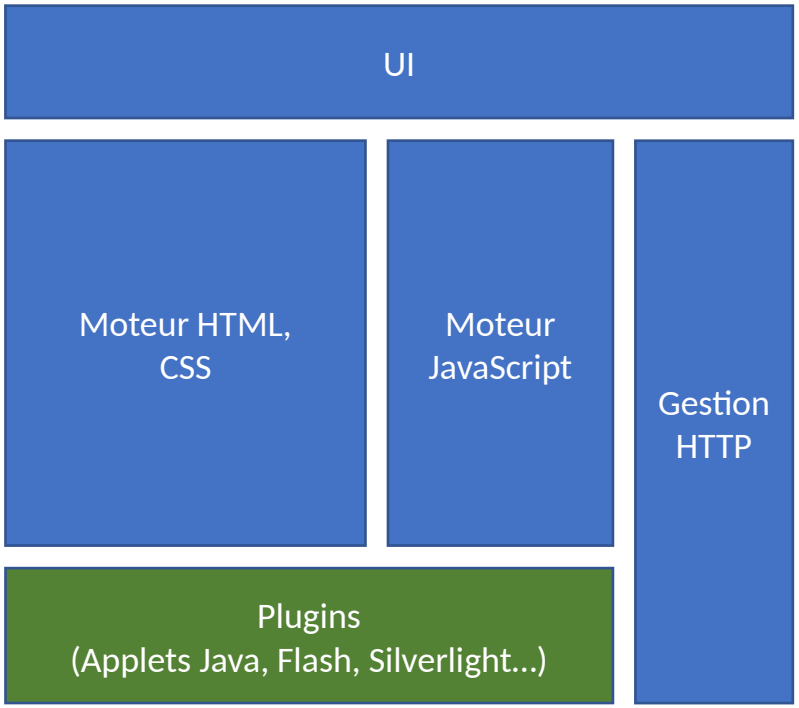
- Binaire exécutable dans les navigateurs
- Alternative ou complément à JavaScript



https://wasmlabs.dev/articles/docker-without-containers/

2000 : Intégration d'applets

- balise ou script référençant une petite application compilée
- exécutée par le navigateur dans un espace réservé




```
<html>
  <head>
    <script type="text/javascript">
       swfobject.embedSWF("myContent.swf", "myContent", "300
    </script>
  </head>
  <body>
    <div id="myContent"> <p>Alternative content</p> </div>
    <object width="400" height="50" data="bookmark.swf"></ob
    <embed width="400" height="50" src="bookmark.swf">
    <applet code="Bubbles.class" width="350" height="350">
          Java applet that draws animated bubbles.
    </applet>
    <object width="300" height="300"
          data="data:application/x-silverlight-2,"
          type="application/x-silverlight-2" >
         <param name="source" value="SilverlightApplication1.xap
    </object>
  </body>
```

2010-15 : déclin des applets et apparition d'HTML 5

- Flash/Silverlight : problèmes de sécurité et performance, prise en charge abandonnée
- HTML5 : fonctionnalités pour des applications plus riches et interactives :
  - Contenus multimedia
  - Formulaires
  - Graphisme
  - Stockage de données local
  - Web Workers (scripts JavaScript qui s'exécutent en arrière-plan, en parallèle avec le thread principal de l'application avec lequel ils communiquent par messages)
  - Mobile : API de périphériques (Géolocalisation, caméra et micro, autres capteurs…)

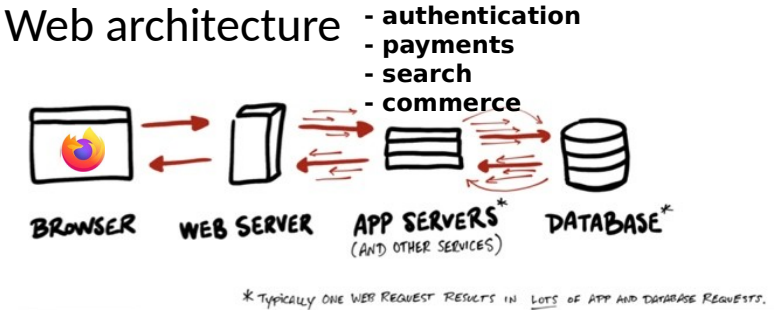
2015 : JAMstack

Transfère la charge du serveur vers le client autant que possible

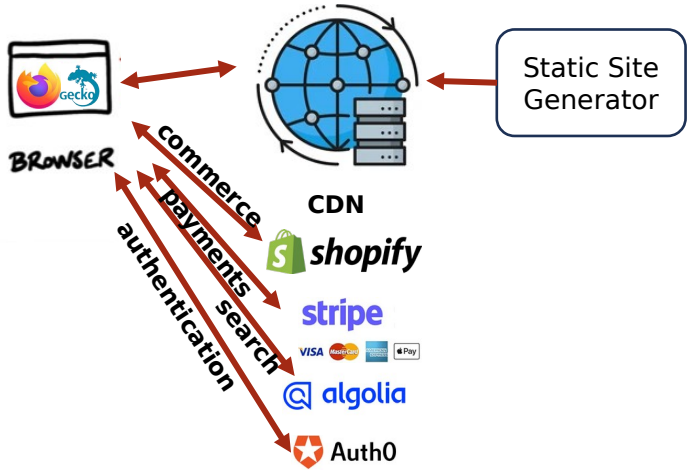
Sites pré-rendus en pages statiques et données récupérées via des APIs

- **JavaScript** pour que le navigateur prenne en charge l'aspect et le comportement d'un site
- **APIs** pour que le client requête directement les données
- **Markup** statique chargé dans un CDN

Traditional Web architecture



Jamstack

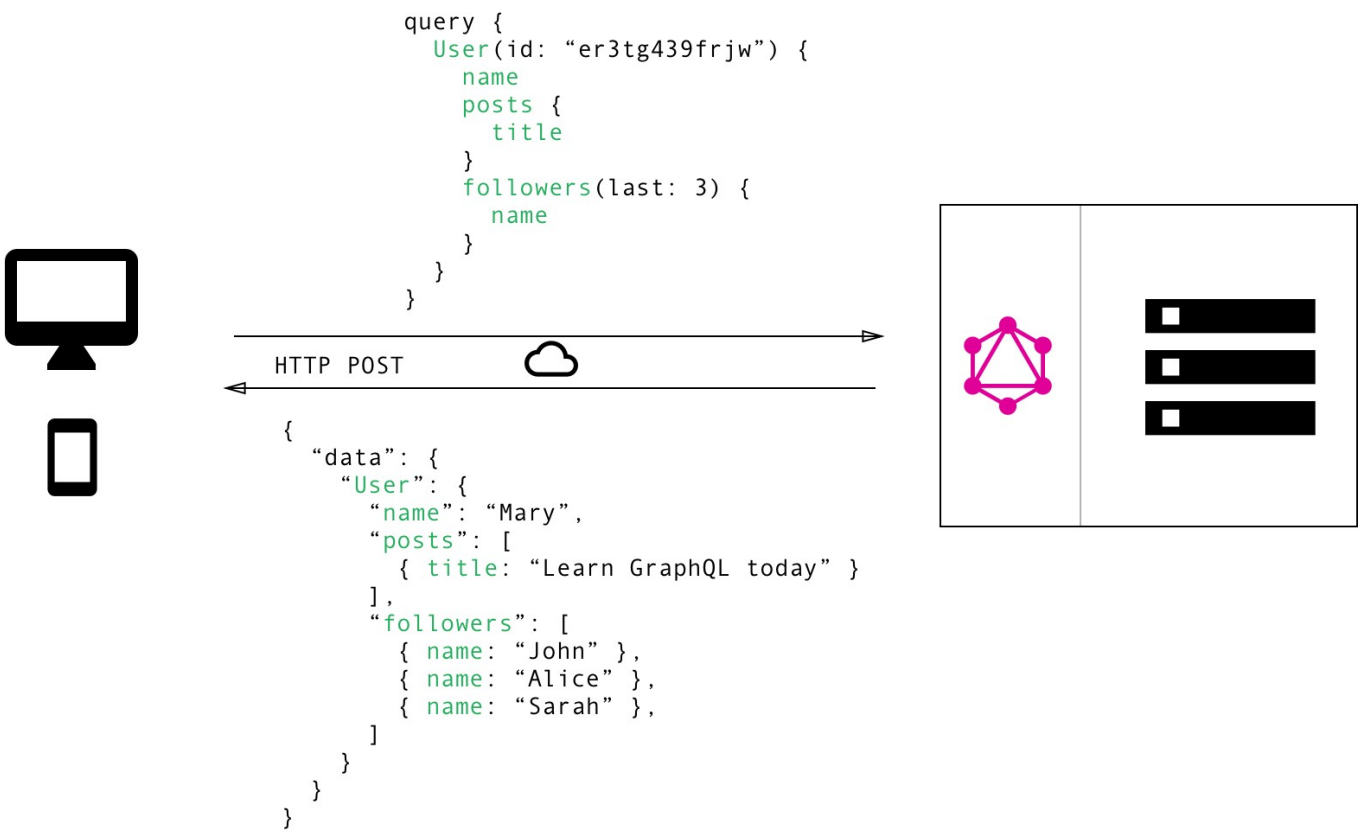


Jamstack

- Avantages :
  - Chargement rapide
  - Passage à l'échelle plus simple et moins cher
  - Les failles de sécurité du front ne menacent pas le back
- Inconvénients :
  - Mise à jour moins conviviale pour les éditeurs de contenus
  - Dépendant des systèmes tiers
  - Mal adapté aux sites demandant des fonctions complexes et dynamiques

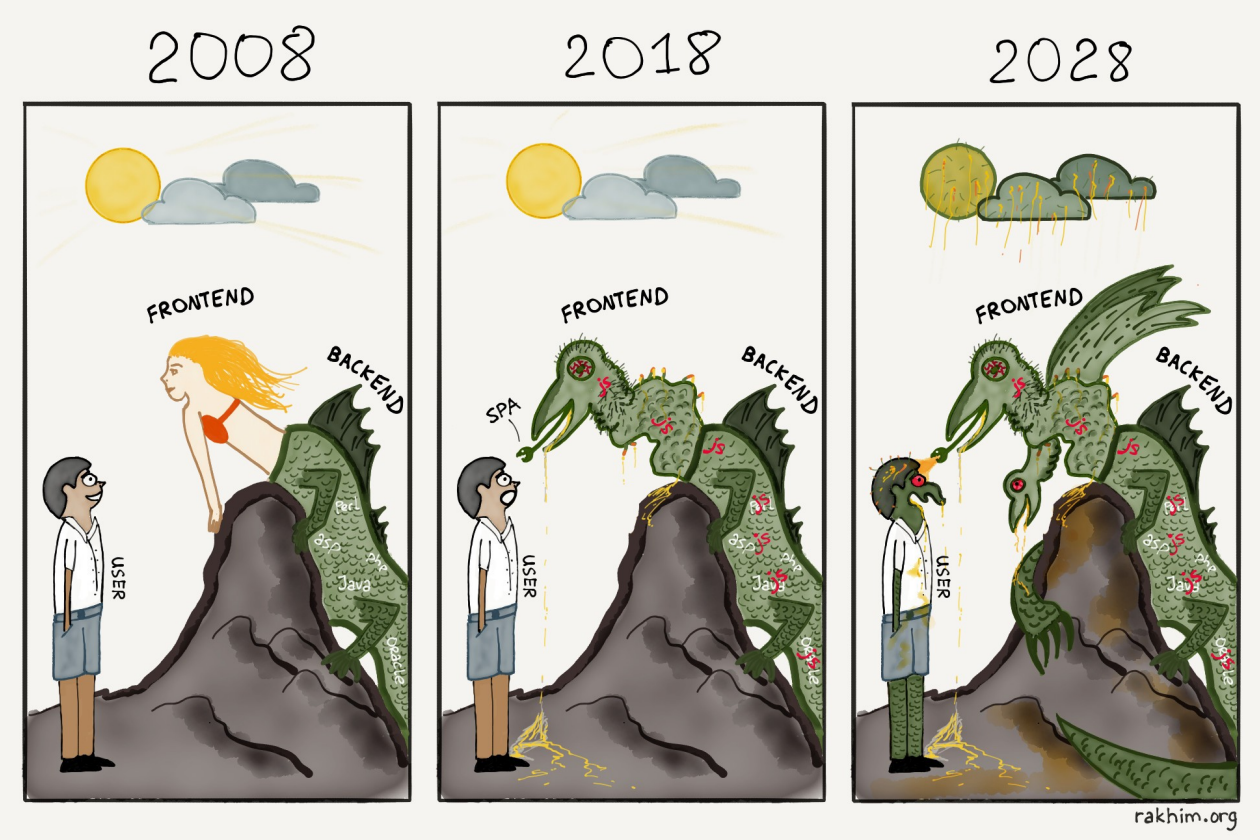
2015 : GraphQL

- Au lieu de points de terminaison spécifiques le back expose un schéma de données
- Le client requête uniquement les données dont il a besoin



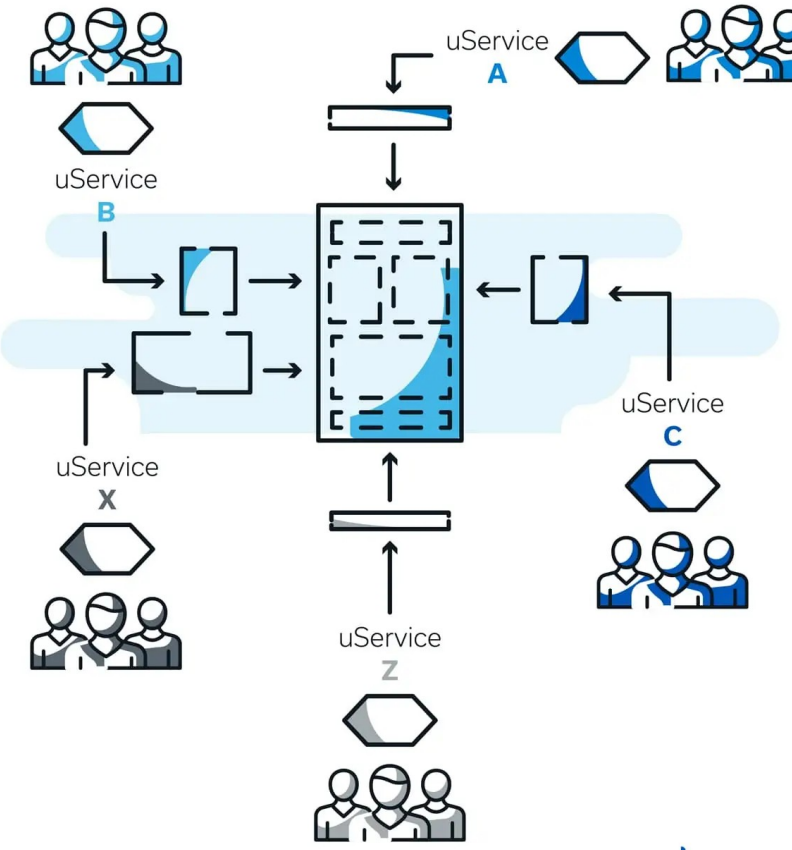
Conclusion :
vers des architectures “sans tête”

- Couplage lâche du back qui expose des APIs et du front qui récupère et présente les données



2015 : micro frontends

- Page web assemblage de micro composants
- Chaque équipe développe son micro composant de la BD à l'UI
- Nécessite coordination pour
  - Homogénéité visuelle
  - Authentification partagée



https://bluesoft.com/micro-frontends-the-missing-piece-of-the-puzzle-in-feature-teams/