Qu’est-ce que le JavaScript ?

Sommaire

Les repères de lecture


Retour au chapitre


Définition


Objectif(s)


Espace Élèves


Vidéo / Audio


Point important / À retenir


Remarque(s)


Pour aller plus loin


Normes et lois


Quiz

Introduction

JavaScript est un langage web interprété, gratuit et open source (code source disponible), il permet d'interagir avec l'internaute sans avoir à recharger la page web sur laquelle il se trouve. Le code JavaScript est exécuté par le navigateur internet que vous utilisez (Firefox, Chrome, Microsoft Edge, Opéra), et téléchargé avec les éléments HTML composant la page web affichée.

JavaScript (JS) est un langage de programmation principalement utilisé dans les pages web interactives.

Nous allons rapidement aborder l'utilisation initiale du JavaScript à ces applications actuelles.

I. Petite histoire du JavaScript

A. Vérification de formulaire

Initialement, le JavaScript était uniquement cantonné au développement de petites facilités complémentaires (généralement la vérification de formulaire). À cette époque (celle que l'on nomme Web 1.0), tout se faisait côté serveur (analyse des requêtes transmises par l'internaute, stockage et lecture des informations, composition des pages HTML, et envoi vers le navigateur).

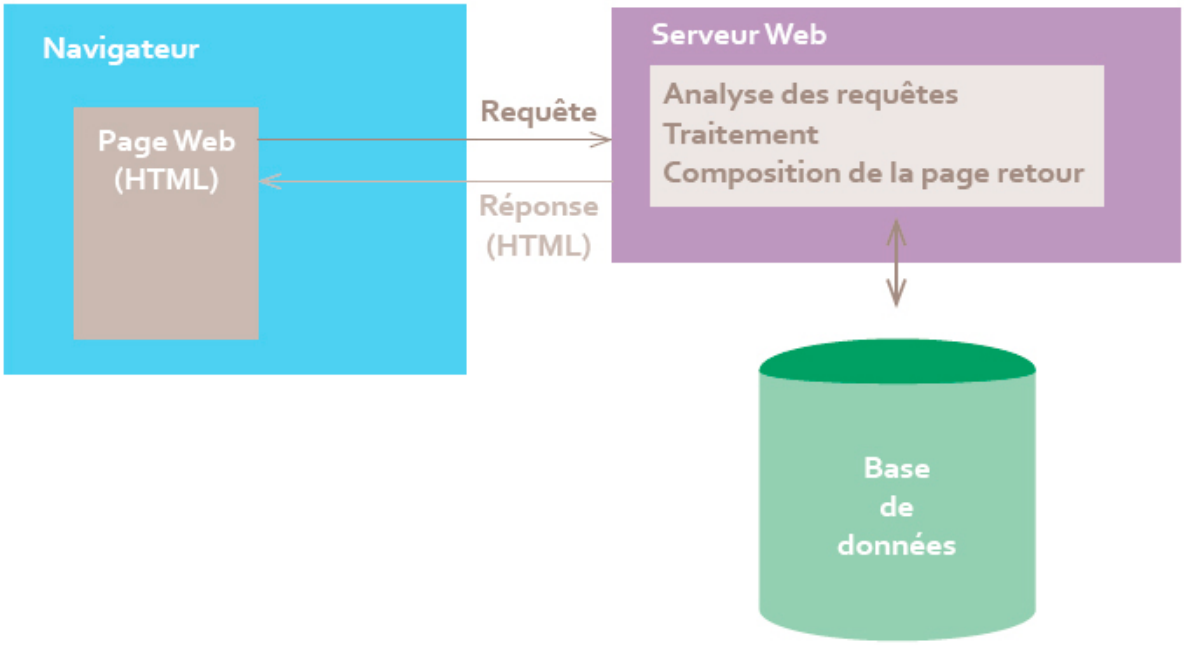

Fig. 1

B. Affichage dynamique

Les applications web devenant de plus en plus ambitieuses et complexes, il est vite apparu nécessaire d'éviter au maximum les rechargements de pages à chaque interaction de l'internaute. Pour ce faire, il fallait que le navigateur puisse en temps réels analyser lui même, les actions de l'internaute et modifier à la volée la page affichée.

JavaScript contenait toutes les fonctions nécessaires pour réaliser ces tâches :

- fonctions permettant de modifier à la volée le HTML (donc la page web affichée) ;
- fonctions d'interaction avec le serveur en tâche de fonds (Ajax) ;
- capacité à traiter les évènements :
  - ☐ provoqués par l'internaute (touche pressée, changement de champs de saisie, clic…) ;
  - ☐ planifiés (exécution de fonctions périodiques) ;
  - ☐ initialisés par le navigateur (chargement ou déchargement d'une page web…).

L'utilisation massive de ces possibilités fut une contribution essentielle à l'essor de ce qu'on nomme le Web 2.0, et à l'apparition de systèmes « temps réel » comme le sont les réseaux sociaux. Et si vous essayez d'imaginer un Facebook où il vous faudrait recharger (on dit aussi « rafraîchir ») sans cesse votre affichage pour avoir les mises à jour des commentaires, des photos ou autres informations, vous conviendrez vite que ce site web n'aurait pas pu avoir le succès qu'il connaît aujourd'hui.

C. Développement d'application

Aujourd'hui, il semble que l'évolution du Web (on parle du Web 3.0), s'oriente vers le développement d'applications web semblables à celles que l'on peut installer sur son ordinateur. Pour y parvenir, le JavaScript a été complété par des fonctions très puissantes (glisser/déposer, traitement d'image, capacité de stockage sur le navigateur, exécution « offline ») que l'on nomme HTML5.

Parmi les possibilités marquantes, on peut noter qu'il existe désormais des applications dites entièrement « navigateur » (qui n'utilisent pas de serveur) ; que le JavaScript peut s'exécuter sur le serveur, et que la Fondation Mozilla qui développe le navigateur Firefox vient juste de « releaser » un système de gestion de mobile dont les applications s'écrivent nativement en JavaScript. Le JavaScript a donc de beaux jours devant lui !

Voici deux exemples d'application du Javascript :

Fig.2 Le site Twitter

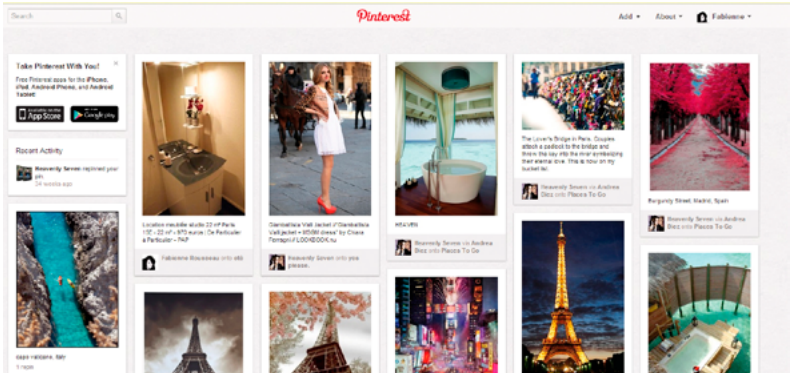

Fig.3 Le site Pinterest

II. Problème de compatibilité

Bien que normalisé par la fondation W3C, le JavaScript (mais aussi le HTML et le CSS dans une moindre mesure) a fait les frais de la guerre des navigateurs. Ainsi, on ne trouve parfois pas les mêmes fonctions ou les mêmes résultats de fonctions selon le navigateur ou, même la version de navigateur utilisée. Cela peut rendre le travail du développeur très laborieux lorsqu'il s'agit d'écrire du JavaScript compatible quel que soit le navigateur utilisé par l'internaute.

Donc, initialement conçu pour le développement de petites applications, le JavaScript s'est avéré un peu démuni lorsqu'il vint à s'agir de développer de grosses applications. Il fallait pouvoir développer vite et efficace.

III. Apparition de librairies de fonction : jQuery

Heureusement, le langage permettait au développeur de factoriser le code en lui rendant possible le développement de ses propres fonctions. Mais un saut de productivité a vraiment été franchi avec l'apparition de bibliothèques de fonctions prêtes à l'emploi, compatibles multi-navigateurs.

Leur utilisation permet au développeur de se concentrer sur la logique « métier » de l'application – celle directement bénéfique pour l'utilisateur – plutôt que de « patauger » dans les couches de bas niveau et dans les méandres de la compatibilité. La plus connue de ses bibliothèques est **jQuery**, qui est presque un langage en soi, et qui plus est évolutif, car il s'enrichit sans cesse de fonctions et de plugins mis à disposition par la communauté des développeurs de JavaScript ou par les éditeurs de grands sites.

A. Performance

Bien qu'étant un langage interprété et contrairement à certaines idées reçues, le JavaScript est un langage très rapide. La puissance des ordinateurs (pc et mobiles) est largement suffisante pour que son utilisation soit quasiment transparente pour l'internaute, si l'application a été un tant soit peu bien conçue.

Par ailleurs, son utilisation permet de déporter une partie importante de la logique applicative vers le navigateur et de soulager ainsi d'autant les serveurs (Puissance CPU, mémoire) et le réseau (diminution du nombre de requêtes lourdes). Les délais de traitement des requêtes et de transit dans le réseau sont donc réduits ; les temps de réponses d'autant, ce qui est tout à l'avantage de l'expérience utilisateur.

B. Ne pas confondre Java et JavaScript

Java et JavaScript sont deux langages différents qui n'ont rien à voir entre eux, bien qu'ils puissent interagir ensemble. L'un, s'exécute sur le navigateur (JavaScript), l'autre sur le serveur (Java).

Le JavaScript est un langage indépendant ! Cela signifie qu'il ne se soucie pas du langage utilisé pour le développement de l'application côté serveur (Java, PHP, Ruby on Rails, Ruby, C, ASP, Perl, voire JavaScript).

IV. Comment développe-t-on en JavaScript ?

Comme tout langage parlé ou écrit, le langage JavaScript possède un vocabulaire, une grammaire et une ponctuation. C'est donc en vous enseignant ces éléments que nous allons vous familiariser à ce langage et vous inculquer les bases essentielles pour devenir un bon développeur JavaScript.

Tout apprentissage nécessite une pratique, nous les complèterons par des exercices qui vous permettront de tester et cimenter vos connaissances.

Tout ceci représente déjà une masse de connaissances non négligeables, ainsi les fonctions évoluées (comme l'utilisation du HTML5 et de JQuery) seront abordées dans un autre manuel.

A. Pré-requis

Ce que vous devez déjà savoir : connaissance basique du HTML, et du CSS.

Le JavaScript trouve son plus grand intérêt dans la manipulation du HTML et du CSS qui permettent la modification de l'affichage présenté à l'utilisateur. Nombre de ses fonctions sont spécifiquement dédiées à cette tâche. Il vous faut donc au moins quelques connaissances de bases de ces deux normes.

B. Chargement du JavaScript par le navigateur

Le JavaScript est chargé par le navigateur lors du chargement des pages HTML à l'aide de balises Script. Les balises Script permettent soit l'intégration directe du code dans le HTML, soit le chargement de fichiers de code JavaScript.

C. Code JavaScript directement inséré dans le HTML

L'insertion s'effectue à l'aide de la balise HTML « SCRIPT » :

```
<script type="text/javascript"> Votre code JavaScript </script>
```

Fig. 4

Il est conseillé de mettre le code en commentaire (HTMLement parlant) afin d'éviter que les navigateurs ne supportant pas le JavaScript – ce qui est assez marginal – n'affichent pas le code JavaScript dans le fil de la page affichée par l'internaute.

Le code JavaScript est inséré directement dans le code HTML ou dans des fichiers d'extension « js ». Vous utilisez donc le même éditeur de texte que celui que vous pourriez utiliser pour le HTML ou le CSS. Parmi les plus couramment utilisés et les plus faciles à prendre en main, nous pouvons vous conseiller :

- Notepad ++ ;
- TextEdit (l'éditeur standard de Mac OS X) ou TextMate ;
- SublimeText ou Atom ;
- Nano ou Vim (Linux).

V. Votre premier script JavaScript

1. À l'aide de l'éditeur que vous aurez choisi, créez un fichier. html contenant le texte suivant :

```
<html>
<body>
<script type="text/javascript">
document.write("Bonjour à tous!")
</script>
</body>
</html>
```

Fig. 5

2. Sauvegardez votre fichier.

3. Double-cliquez sur l'icône représentant votre fichier pour l'ouvrir dans votre navigateur par défaut (ou ouvrez-le à l'aide du menu « fichier » de votre navigateur si celui-ci est déjà lancé).

Résultat attendu
Le code ci-dessus doit afficher le résultat suivant :

Bonjour à tous !

Fig. 6

A. Fichier JavaScript « appelé » par le HTML

Il vous faudra pour ce faire créer au moins deux fichiers :

– un fichier contenant le HTML ;
– un fichier contenant votre code JavaScript. Ce dernier est inséré dans le HTML lors du chargement du fichier HTML à l'aide de la balise SCRIPT et de l'attribut SRC :

```
<html>
<head>
<script type="text/javascript" src="nom-de-fichier.js"></script>
</head>
<body>
...
</body>
</html>
```

Fig. 7

B. Quand s'exécute le code JavaScript ?

Le navigateur lit séquentiellement le HTML de haut en bas. Le code JavaScript est exécuté par le navigateur après lecture de la balise SCRIPT et du téléchargement du fichier JavaScript spécifié par l'attribut SRC (s'il existe).

C. Quelle est la meilleure manière d'inclure le code JavaScript ?

On préférera séparer le code JavaScript du HTML. On privilégiera donc l'inclusion de fichier JavaScript à l'aide de la balise HTML SCRIPT et de son attribut SRC, vous pourrez ainsi rendre votre code bien lisible et réutilisable en le répartissant dans autant de fichiers que vous le jugerez nécessaire.

En général, on place les balises script dans le header du HTML :

```
<html>
<head>
<script ...>
Code JavaScript
</script>
</head>
<body>
</body>
</html>
```

Fig. 8

Cela permet de vous assurer que tous les éléments HTML (boutons, liens…) et les fonctions JavaScript sont chargées avant que le navigateur ne rendent possibles les interactions de l'internaute avec les éléments de la page que vous lui présentez.

Si vous placez votre ou vos inclusions(s) dans le corps du HTML (entre les balises < body > et < /body > il y a un risque que l'internaute puisse interagir (cliquer sur un bouton par exemple) sans que le code JavaScript qui gère cette interaction ne soit prêt à traiter cet évènement (le navigateur n'ayant pas encore téléchargé le code JavaScript correspondant).