

JavaScript

Histoire

JavaScript (langage de scripting objet) langage de prog de scripts qui peut être utilisé pages web interactives.

Langage créé en 1995 Brendán Eich (appelé à l'époque liveScript) pour le compte de Netscape.

Ensuite l'alliance Netscape-Java → JavaScript. Syntaxe de base intentionnellement similaire à Java et C pour réduire le nombre de nouveaux concepts nécessaires pour apprendre le langage.

ECMA a proposé un standard « ECMAScript » (ISO) : ECMA 262, ISO/CEI 16262

Pendant longtemps, (l'époque jQuery), il s'agissait de l'ES5, la version 5 de l'ECMAScript.

Depuis juin 2015, une nouvelle version est disponible : l'ES6 (Une profonde transformation). La mise à jour du standard JavaScript est désormais annuelle (avec peu de nouveautés).

ECMAScript 2018 pour l'année 2018. Tous les navigateurs modernes supportent l'ES6 depuis un moment, et les frameworks

majeurs (Angular, React, Vue...) utilisent tous cette nouvelle version de JavaScript.

JavaScript est actuellement à la version 1.8.6, est une implémentation du standard ECMA-262.

Sécurité :

onglet navigateur constitue un périmètre séparé dans lequel s'exécute le code (en termes techniques ces périmètres sont des « environnements d'exécution ») signifie → dans la plupart cas, code

chaque onglet est exécuté complètement séparément, code d'un onglet ne peut affecter directement code autre onglet ou autre site.

Ordre d'exécution :

De Haut vers bas → faire attention aux références d'objets.

L'attribut length : let tab = ['pomme', 'banane']; tab.length;

La méthode split :

let myData = 'Manchester,London,Liverpool,Birmingham,Leeds,Carlisle'; let myArray = myData.split(',') ;

La méthode join : let myNewString = myArray.join();

La méthode toString : let dogNames = ['Rocket','Flash','Bella','Slugger']; dogNames.toString(); //

Rocket, Flash, Bella, Slugger

Les méthodes push() et pop() Les méthodes unshift() et shift()

DHTML est un nom générique donné à l'ensemble des techniques utilisées par l'auteur d'une page web pour que celle-ci soit capable de se modifier elle-même en cours de consultation dans le navigateur web. = **HTML+CSS+JS+DOM+AJAX**

Document Object Model (DOM) (est un W3C standard) décrit une interface indépendante de tout langage de programmation et de toute plate-forme, permettant à des programmes informatiques et à des scripts d'accéder ou de mettre à jour le contenu, la structure ou le style de documents. DOM est souvent identifié par une arborescence de la structure d'un document et de ses éléments. P.ex. chaque élément généré à partir du balisage comme, dans le cas de HTML, un paragraphe, un titre ou un bouton de formulaire, y forme un nœud. DOM utilisé pour pouvoir modifier facilement ou accéder au contenu des pages web. A partir arbre DOM donné, il est aussi possible de générer docs dans langage balisage voulu;

Objets HTML et DOM

Window est objet → fenêtre dans laquelle s'affiche une page Web. Location est un sous-objet de Windows. C'est un objet coté-client (donc créé par le navigateur). Location est actuellement une interface intégrée à DOM

L'objet Document défini dans la spécification DOM permet d'accéder au contenu d'une page HTML. Les méthodes les plus utilisées de l'objet sont

getElementsByName et getElementById, open, write... History est un des sous-objets de Windows qui sert d'interface avec l'historique

de navigation conservé par le navigateur. L'interface Node est définie dans la spécification DOM 2, elle permet d'accéder à la structure d'un document HTML vu comme une arborescence d'élément, et de modifier cette structure.

L'interface NodeList → un objet DOM 2, qui permet d'accéder aux éléments d'une page

Web ou fichier XML. C'est élément dynamique, tous changements de structure dans la page modifient le contenu de NodeList.

JavaScript permet de modifier les styles CSS:

On peut accéder à un élément via document.getElementById... modifier la classe d'un élément :

On peut accéder à une classe via l'attribut className de tout élément html...

Toute élément CSS est accessible via JavaScript et DOM selon la règle:

Le nom ne change pas si absence de tiret. Sinon supprimer le tiret et mettre en majuscule la première lettre qui le suit.manipuler les feuilles de style elles-mêmes:

Il s'agit d'activer et désactiver des feuilles de style

Utiliser la propriété disabled pour les éléments <link> et <style>

Accès aux API fetch

Une interface pour récupérer des ressources A un objectif similaire à XMLHttpRequest très utilisé pour l'approche AJAX.

Fonctionnalités plus souple, plus puissante Modèle requête ↔ réponse Définition générique pour les objets de type Request et Response Permet une utilisation étendue de ces objets : service workers, l'API Cache, Autres programmes ou mécanismes qui manipulent ou modifient des requêtes et des réponses

L'objet XMLHttpRequest constitue la clé de voute d'Ajax.

Onload : (on lui affecte une fonction avec les instructions à exécuter lorsque la requête est reçue);

Onreadystatechange : (on lui affecte une fonction avec les instructions de traitement de la réponse);

readyState : l'état d'avancement de la requête de 0 (non initialisée) à 4 (complétée), **responseText / responseXML** : la réponse du serveur à la requête en texte (i.e. objet reconnu par JavaScript) ou formatée en un objet XML. **Status** : état de la réponse HTTP du serveur. (200 OK) **open(method, url, bool)** : (si mode asynchrone bool vaut true et sinon false) **send(data)** : Envoie la requête HTTP au serveur (quand il n'y a pas de données la valeur est null)

GET - attention à l'écriture de l'argument url déclarer une variable pour récupérer la valeur du champ cible du formulaire et ensuite utiliser la fonction javascript escape : var var1 =

document.forms[index1].elements[index2].value; var url = "http://adresse-web-serveur.nom-procedure?var1=" + escape(var1);

POST : Utiliser la propriété setRequestHeader de l'objet XMLHttpRequest req.setRequestHeader('Content-Type',

'application/x-www-form-urlencoded') Ensuite, préparer la donnée (même démarche que pour url ci-dessus) à envoyer au serveur : var data= "id=" + escape(username) + "&password=" + escape(password); req.send(data);

HTTP/BORDEL

Entête de requete client

Accept = type MIME visualisable par l'agent

Accept-Encoding = méthodes de codage acceptées compress, x-gzip, x-zip

Accept-Charset = jeu de caractères préféré du client

Accept-Language = liste de languesfr, en, ...

Authorization = Identification du browser auprès du serveur

Cookie = cookie retourné

Content-Length = Longueur du corps de la requête

From = adresse email de l'utilisateur rarement envoyé pour conserver l'anonymat de l'utilisateur

Host = spécifie la machine et le port du serveur un serveur peut héberger plusieurs serveurs

If-Modified-Since = condition de retrait la page n'est transférée que si elle a été modifiée depuis la date précisée. Utilisé par les caches

Referer = URL d'origine page contenant l'ancre à partir de laquelle

a été trouvé l'URL.

User-Agent = donner des informations sur le client, comme le nom et la version du navigateur, du système d'exploitation.

<style type="text/css">

@import url(styles/affichage.css) media;

</style>

Le code JavaScript :

Peut changer le contenu d'un élément HTML, la valeur d'un attribut

HTML

document.getElementById("demo").innerHTML = "Hello

JavaScript";

document.getElementById("image").src = "newPic.png";

document.getElementById("demo").style.fontSize = "35px";

Est ajouté dans HTML via balise script dans body ou head.

Plusieurs scripts

peuvent être ajoutés. Permet de définir des fonctions :

function myFunction() {

document.getElementById("demo").innerHTML

= "Paragraph

changed;"}

Peut être appelé lorsque un événement se produit:

<button type="button" onclick="myFunction()">Try it</button>

Variables :

avec Let : Portée : bloc ({...}) //// Double déclaration : non ⇒

SyntaxError

Utilisation de la variable avant sa déclaration : non ⇒ ReferenceError

Variables déclarées avec Var // Portée : Globale (à l'extérieur) ou

fonction // Double déclaration : oui // Utilisation de la variable avant

sa déclaration (voir Javascript Hoisting) : oui

Hoisting est (pour de nombreux développeurs) un comportement

inconnu ou négligé de JavaScript ⇒ bugs au niveau des

programmes

Pour éviter ces bugs il est recommandé de déclarer toutes les

variables au début de chaque portée

JavaScript et la sécurité : JavaScript ne peut pas

accéder aux fichiers stockés sur le site du

client; JavaScript ne peut pas accéder aux pages

html venant d'autres sites. JavaScript n'autorise

pas l'installation de programmes exécutables sur

le site client;

Limites de JavaScript Compatibilité navigateurs

et plateformes; Code JavaScript non visible par

les moteurs de référencement;

Les noeuds DOM :

Les liens hiérarchiques entre noeuds définissent structure du

document et sont représentés dans l'arbre DOM. Il y a trois types de

noeuds DOM dans une arborescence :

Les noeuds « éléments ». L'étiquette associée au noeud donne le nom

de l'élément;

Les noeuds « text ». Le texte contenu dans un élément apparaît

comme un nœud texte dans l'arbre DOM.

Les noeuds « attribut ». Les attributs d'un élément html sont

représentés comme des noeuds attributs dans l'arbre DOM.

Méthodes de Document. Element createElement(String) Crée

élément, balise, dont nom est donné, et retourne objet Element.

Element getElementById(String) Retourne élément dont on donne

l'identificateur.... void open() Crée nouveau document. void close()

Ferme document courant. void write(String) Ecrit dans le

document. Warning: Use of the document.write() method is

strongly discouraged. void writeln(String) Ecrit chaîne en paramètre

et ajoute saut à la ligne.

Warning: Use of the document.writeln() method is strongly

discouraged.

Attributs de Document (Attributs en lecture seule, définis lors

de la création de la page.)

DocumentType **doctype** (Le doctype définit en tête de page.)

DOMImplementation **implementation**

Element **documentElement** objet qui représente une balise et

est doté de méthodes.

String **title**,

String **URL**,

HTMLElement **body**,

HTMLCollection **links**,

HTMLCollection **forms**,

String **cookie**

événements page et fenêtre

onabort - si y a une interruption dans

le chargement onerror - en cas d'erreur

pendant le chargement de la page

onload - après la fin du chargement de

la page onbeforeunload ; onunload ;

onresize ; **événements souris** onclick -

sur un simple clic ondblclick - sur un

double clic onmousedown - lorsque

que le bouton de la souris est enfoncé,

sans forcément le relâcher

onmousemove - onmouseout -

onmouseover - onmouseup;

événements clavier onkeydown -

lorsqu'une touche est enfoncée

onkeypress - lorsqu'une touche est

pressée et relâchée onkeyup -

lorsqu'une touche est relâchée