Javascript

Sébastien GERARD
Thomas BILLIAU
2018

Objectifs

Apprendre à utiliser le langage JavaScript dans vos pages HTML

Présentation du langage

JavaScript = langage de scripting interprété côté client

Créé par Netscape en 1995

Standardisé par ECMAScript

Intérêt : ajouter de l'interactivité utilisateur/pages web

Javascript: utilisations

Javascript va principalement nous servir à :

- Ajouter ou retirer des éléments à une liste ou tableau affiché sur la page
- Ajouter, modifier ou supprimer du texte sur la page
- Changer le style CSS de notre page

Modifier dynamiquement notre document, c'est à dire selon certaines conditions (un clic, un survol, une valeur...)

Intégration du code

```
<body>

    <!-- Le code JavaScript peut être placé dans des balises -->
    <script>
    // Votre code ici
    </script>

    <!-- Le code JavaScript peut être placé dans un fichier externe -->
    <!-- On placera toujours la balise script en dernier -->
    <script src="javascript.js"></script>

</body>
</html>
```

Intégration du code

On privilégiera le script interne si :

- Il y a peu de code.
- Il y a du code spécifique au contexte de la page

On privilégiera l’externalisation de script pour :

- Mutualiser le code et faciliter la maintenance.
- Constituer des bibliothèques de fonctions réutilisables.

La performance (script mis en cache)

Intégration du code

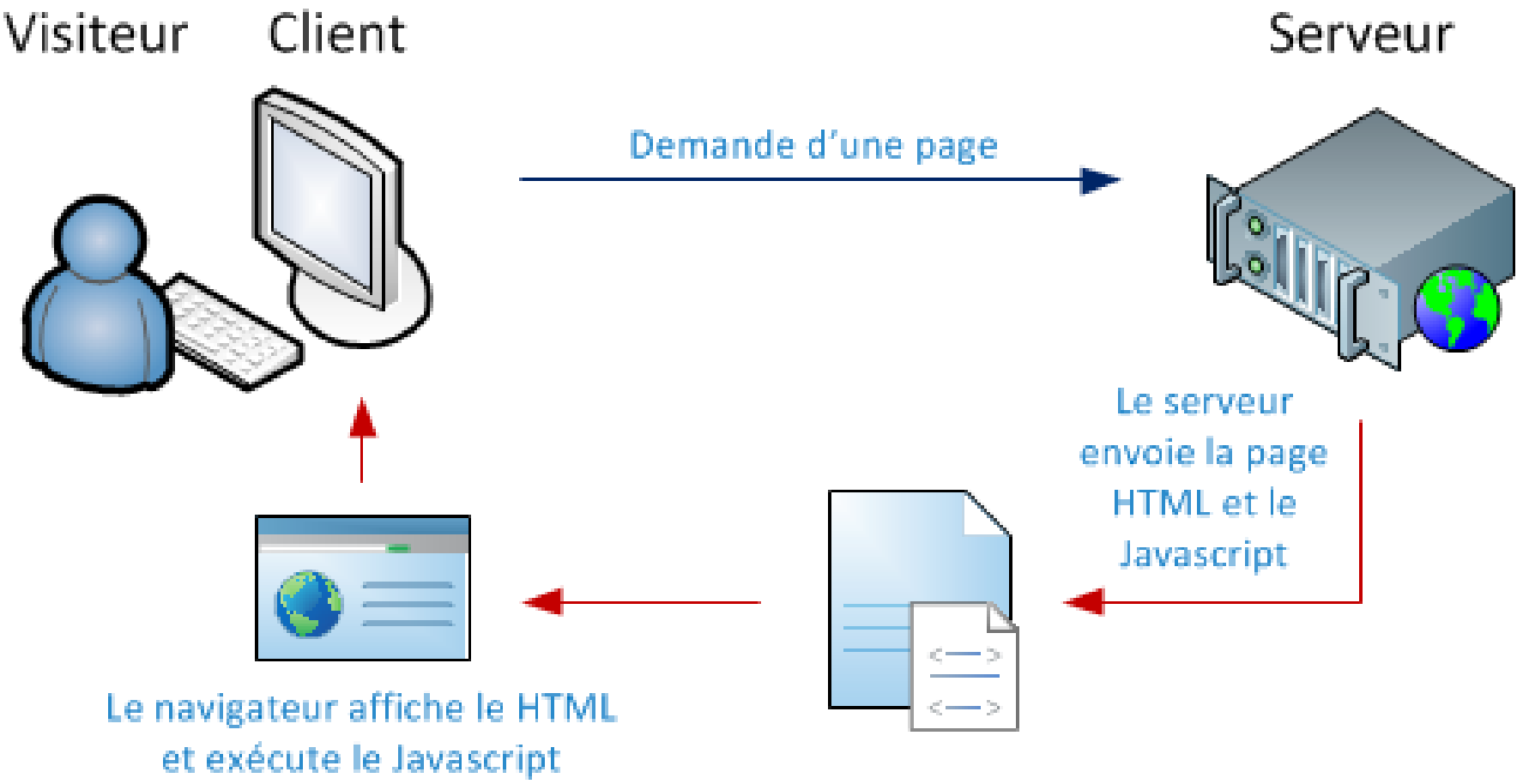
Conseils :

On commence par l'importation des fichiers externes avant le codage inline.

Pour les navigateurs sans JavaScript (ou désactivé). Il faut utilisé une balise <noscript> pour prévenir l'utilisateur que le site nécessite l'activation du Javascript.

```
<noscript>
    Javascript est désactivé sur votre navigateur
</noscript>
```

Interprétation



Interprétation du code


```
// Le JavaScript est interprété de façon séquentielle par le navigateur « au fil de l'eau ».


// Une variable ne peut pas être utilisée si elle n'a
// pas d'abord été déclarée.

// Dans le bloc, l'appel de la fonction « test » produit
// une erreur.

test();
var test = function(){
    alert('ok');
}
```

Syntaxe de base

Dawan
Training is our way

```
// SYNTAXE DE BASE

// Chaque ligne de code correspond à une instruction .

// Bonne pratique : séparer les instructions par des ';'
var a = "bonjour";
alert(a);
// Les blocs d'instructions sont placés entre { }
if (a==1)
{
    a++;
    alert(a);
}
```

Commentaires

```
// Commentaire sur une ligne
var commentaire;
/*
Commentaire
sur
plusieurs
lignes
*/
var test;
```

MOTS RÉSERVÉS ECMA 5 ET 6


```
// MOTS RÉSERVÉS ECMA 5 ET 6

// Mots actuellement réservés:

// break            case            catch
// continue         debugger        default
// delete           do              else
// finally          for             function
// if               in              instanceof
// new              return          switch
// this             throw           try
// typeof           var             void
// while            with

// Mots réservés dans le futur:

// class            enum            export
// extends          import          super

// implements       interface       let
// package          private         protected
// public           static          yield
```

Variables

```
// VARIABLES

// Elles doivent être déclarées avant leur
// utilisation.

// Les variables peuvent être déclarées :

// Avec l'instruction var :
var a
// Permet de « localiser » la variable au bloc de
// code courant.

// En leur assignant directement une valeur :
b="Hello";
// L'absence du mot clé var rend la variable globale
// à la page.
```

Variables

```
// Nom de variable sensible à la casse.

// Test est différent de test.

// Le nom des variables doit :

    // Commencer par une lettre, _ ou $.

// Contenir exclusivement des lettres, des chiffres, _ ou $.
```

Variables


```
// TYPAGE

// Typage faible :
    // Pas de typages définitif des variables.
    // Une variable change de type en fonction de la valeur qui lui est affectée
    // Types :
        // Primitif :
        // Number (numérique)
        // Boolean (true ou false)
        // String (chaine de caractère)
        // Référence :
        // Array (tableau)
        // Object {}
```

Variables


```
//Type null
    // Valeur explicite.

// Type undefined
    // Valeur par défaut.
    // variable non déclarée / inexistante
    // Assigné à une variable déclarée, mais sans valeur affectée.
```

Variables


```
//PORTEE DES VARIABLES
// Les variables peuvent être globales ou locales

// Une variable globale :

    // Est accessible et modifiable à n'importe quel endroit du programme

    // Est rattachée à l'objet dit "objet Global"

// Une variable locale :

//      Se déclare, dans un bloc de code donné, en utilisant le mot clé var

//      A une portée limitée au bloc de code dans lequel elle est déclarée

//      Attention : si omission du mot clé var, la variable sera globale, même si elle
//      est déclarée dans un bloc de code!
```

La concaténation

```
// LA CONCATENATION

// Se fait avec le caractère + en javascript
// Ex :
var d = 'Bonjour' ;
var e = 'Lucas ';
var f = d+' '+e ;
// TP :
// Tester le code d'exemple et afficher c via un console.log()

// Reprendre l'exercice précédent, en demandant le prénom de l'utilisateur

// Objectif : afficher un message personnalisé dans la console
```

STRING


```
// TYPE STRING

// String est le type des chaînes de caractères

// Création d'une chaîne :
var uneChaine = "Une chaine de caractère";

// Les propriétés :

    // uneChaine.length : longueur de la chaîne
```

String

```
// Les méthodes :

    // uneChaine.indexOf(chaine) : position d'une chaîne dans uneChaine
    // uneChaine.replace(texte, nouveauTexte) : recherche / remplace
    // uneChaine.toLowerCase() : passe toute la chaîne en minuscule
    // uneChaine.toUpperCase() : passe toute la chaîne en majuscule
    // uneChaine.small() : formate la chaîne avec la balise <small>
    // uneChaine.link(url) : formate la chaîne pour en faire un lien
    // uneChaîne.anchor(name) : formate la chaîne pour en faire une ancre
    // uneChaine.substring(indexDebut, indexFin) : renvoi une portion de la chaîne
    // uneChaine.split(chaineSeparateur) : transforme une chaîne en tableau en coupant aux
    // occurences du séparateur spécifié
```

Array


```
// LES TABLEAUX

// En javascript les tableaux sont de type Array

// Création d'un tableau :
var unTableau = ['Pomme','Poire'];
var unTableau = [];
// Accéder aux éléments d'un tableau

// unTableau[index] : premier élément à index 0
```

Array


```
// Les propriétés :

// unTableau.length : nombre d'éléments

// unTableau.pop() : enlève le dernier élément du tableau

// unTableau.shift() : enlève le premier élément du tableau

// unTableau.join(séparateur) : assemble les éléments du tableau pour
// en faire une chaîne

// unTableau.reverse() : inverse l'ordre des éléments

// unTableau.sort([fonction]) : trie les éléments du tableau

// unTableau.concat(tableau1, ...) : concatène des tableaux

// unTableau.slice(debut, fin) : extrait une tranche de tableau, la valeur de
//  début doit être >= et la valeur de fin est strictement <

// unTableau.push(valeur, ...) : ajoute les éléments à la fin du tableau

// unTableau.unshift(valeur, ...) : ajoute les éléments au début du tableau
```

Array


```
// Exemple de création et parcours d'un tableau
var unTableau = ['Element0', 'Element1','Element3'];

for(var i = 0; i < unTableau.length; i++){
    alert(unTableau[i]);
}

//certains anciens navigateurs parcours également les propriétés
for(i of unTableau){
    alert(unTableau[i]);
}
```

Opérateurs arithmétiques

```
// OPERATEURS ARITHMETIQUES

// Les opérateurs de calcul :
// L'addition : +
// La soustraction : -
// La multiplication : *
// Le division : /
// Le modulo : %
```

Opérateurs logiques

```
// OPERATEURS LOGIQUES

// Les opérateurs binaires
// Et : &
// Ou : |
// Ou exclusif : ^
// Décalage à gauche : <<
// Décalage à droite : >>
// Les opérateurs logiques
// Et : &&
// Ou : ||
// Négation : !
```

Opérateurs


```
// OPERATEUR

// L'opérateur d 'affectation.
// = affecte une valeur à une variable.
// Il est possible de le faire précéder d'un
// opérateur arithmétique :
// += -=
// *= /=
// …
// les opérateurs ++ et -- affectent directement la variable.
```

Opérateurs de comparaison


```
// OPERATEURS DE COMPARAISON

// Opérateurs de comparaison :
    // a == b Égalité en valeur après conversion.
    // a != b Différence de valeur après conversion.
    // a < b  Inférieur.
    // a > b  Supérieur.
    // a <= b  Inférieur ou égal.
    // a >= b  Supérieur ou égal.
```

Précédence des opérateurs : link

Tests de types

```
// TEST DE TYPES

// Faites attention au typage faible
// Il peut y avoir une conversion implicite de types différents lors des opérations.
// Vous pouvez tester le type d'une variable avec :
// typeof();
var a = 32;
console.log('Type de ma variable a:');
console.log(typeof(a));
```

Tests de types

```
// TESTS DE TYPES

// Pour une égalité stricte on utilisera une comparaison par type et par valeur :

    // a === b Égalité en type et en valeur.

    // a !== b Différence de type ou de valeur.

// NaN : Not a Number
// La propriété globale NaN est une valeur
// utilisée pour représenter une quantité qui
// n'est pas un nombre
```

Fonctions

Ensemble d’instructions réalisant une certaine tâche
Peuvent prendre 0 ou plusieurs paramètres
Peuvent renvoyer un résultat
Permettent la création de code réutilisable
Permettent de structurer le code et gagner en clarté

```
function nomFonction(parametres){
        //return valeur;
}
```

Appel : nomFonction(parametres);
Appel dans un événement :

```
document.addEventListener("click", function(){
document.getElementById("demo").innerHTML = "Hello World";
});
```

Fonctions


```
// FONCTIONS

// Déclarée via le mot clé function

// Une fonction comporte :

    // Un nom
    // Zéro ou plusieurs arguments entre parenthèses
    // Le corps de la fonction entre accolades { }

// Une fonction peut :

    // Retourner un résultat de traitement (tous types de
    // données) grâce à l'instruction return

    // Le JavaScript parse d'abord les fonctions avant les variables
```

Objets

Les objets sont des sortes de variables (information stockée en zone mémoire) qui vont contenir diverses informations et que nous pouvons utiliser de manière particulière.

```
var personne = {
    prenom : ‘Ida’,
    nom : ‘hub’,
}
```

On pourra alors accéder directement aux valeurs des différentes propriétés de notre objet (ex : **personne.prenom**)

Les objets vont également pouvoir contenir des fonctions, on les appellera alors des méthodes dont on pourra se servir (ex : **personne.fonction()**)

Intérêt du javascript

Les navigateurs web nous offrent une API qui nous permet d'accéder à différentes méthodes et valeurs qui vont nous être utiles dans l'élaboration de nos programmes et de nos documents web

Ex :

```
console.log('1 utilisateur se trouve dans la console') ;
console.warn('Attention je suis dans la console') ;
console.info('Salut, je suis dans la console') ;
```

NB: N'hésitez pas à faire des console.log(); afin
De vérifier que le résultat est bien celui que vous cherchez

Récupérer une valeur et l’utilis (atelier)

Au clic sur un bouton

Utiliser prompt(‘Question : ‘) ; pour afficher une fenêtre de dialogue.

L’appel de prompt devra se faire lors de la création d’une variable.

Afficher la valeur de la variable dans la console.

Algorithmie

Si (il fait trop chaud ET il ne pleut pas) Alors
    Ouvrir la fenêtre
Sinon
    Fermer la fenêtre
FinSi

Si (il ne fait pas trop chaud OU il pleut) Alors
    Fermer la fenêtre
Sinon
    Ouvrir la fenêtre
 FinSi

Algorithmie

```
Si (il fait trop chaud) Alors
    Si (il ne pleut pas) Alors
        Ouvrir la fenêtre
    Sinon
        Fermer la fenêtre
    FinSi
Sinon
    Fermer la fenêtre
 FinSi
```

Structures de contrôle

Structures conditionnelles

- if
- switch

Structures itératives

- while
- do ... while
- for
- for ... in
- for … of
- Array.prototype.forEach()

Conditions

```
if(condition) {
        instructions;
 }
 else if(autre condition) {
        instructions;
 }
 else {
        instructions;
 }
```

```
switch(expression) {
        case cas1:
                instructions;
                break;
        case cas2:
                instructions;
                break;
        default:
                instructions;
}
```

Boucles

while : exécute des instructions tant qu'une expression booléenne est vraie

```
while(expression) {
	instructions;
}
```

do ... while : (semblable à while), les instructions sont exécutées au moins une fois

```
do {
	instructions;
} while(expression);
```

Boucles

for : les instructions sont répétées un nombre de fois (ce nombre est connu)

```
for(var i = min; i < max; i++) {
	instructions;
}
```

for..of : Itération sur les propriétés d'un objet ou sur les éléments d'un tableau

```
for(var i of array) {
	instructions;
}
```

Exécution du code

Le code Javascript sera exécuté lors d'un événement (chargement de la page, clic sur un bouton...) via un écouteur ou un attribut html.

Ex : <button onclick= « alert('click') » ;

DOM 0 ne se pratique plus privilégié addEventListener

Évènements

Déclenchent un traitement en fonction des actions de l’utilisateur (click, mouvement de souris, etc.)

Traités par des gestionnaires d’événements (event Handler)

Utilisation :

Évènements

```
// Évènements

// addEventListener :
document.addEventListener("click", function(){
    document.getElementById("demo").innerHTML = "Hello World";
});
```

link

Principaux évènements

Événement	Déclenchement
load	Au chargement de la page HTML
unload	A la fermeture de la page HTML
mouseOver	Au passage de la souris
mouseOut	A la sortie de la souris
focus	Au focus de l'élément
blur	A la perte du focus
change	Au changement de la valeur
click	Au click de la souris
submit	A l'envoi du formulaire

Liens vers la liste des évènements MDN

TP formulaire

Valider un formulaire

- Ciblez un champ au clique avec un texte pour signalez que le champ est sélectionné ou désélectionné
- Vérifiez que le champ nom, l’email et l’age soit conforme à la désélection

Vérifiez que les champs soit conforme à la validation

TP galerie photo

Réalisez une galerie photo

- Créez une rangée avec quelques images

Au clique sur l’une des images créez un affichage au dessus plus grand

Window


```
// Window

// Un objet de type Window est un objet qui contient des informations
// sur une fenêtre du navigateur

// Une variable window de type Window est créée automatiquement à
// l'ouverture du navigateur

// Elle contient des informations sur la fenêtre courante
```

Window

```
// Principales propriétés :

// window.closed : Spécifie si une fenêtre est fermée
// window.defaultStatus : L'affichage par défaut dans la barre de statut
// du navigateur
// window.navigator : Contient des informations sur le navigateur utilisé
// window est le contexte par défaut (this), on peut donc omettre son nom
// quand on accède à ses attributs (et méthodes)
```

Window


```
// Les méthodes :

// window.focus() : passe la fenêtre au premier plan
// window.blur() : passe la fenêtre au second plan
// window.setTimeout() : évalue une chaîne de caractères ou une fonction
// après un certain laps de temps
// window.print() : imprime le contenu de la fenêtre
// window.moveBy() : déplace la fenêtre d'une certaine distance
// window.moveTo() : déplace la fenêtre vers un point spécifié
// window.close() : ferme la fenêtre
// window.resizeBy() : redimensionne la fenêtre d'un certain rapport
// window.resizeTo() : redimensionne la fenêtre
// window.open(url, nom, attributs) : ouvre une nouvelle fenêtre (popup)
```

Document


```
// Document

// Un objet de type Document est un objet qui contient des informations
// sur le document chargé par la fenêtre du navigateur

// Une variable document de type Document est créée automatiquement lors du chargement
// d'un document par le navigateur

// document est rattaché à l'objet window :
window.document;
```

Document


```
// Les principales propriétés :

    // document.applets : retourne la collection d'applets java présente dans le document
    // document.domain : indique le nom de domain du serveur ayant apporté le document
    // document.forms : retourne la collection de formulaires présents dans le document
    // document.images : retourne la collection d'images du document
    // document.links : retourne la collection de liens du document
    // document.referrer : indique l'adresse de la page précédente
    // document.title : indique le titre du document

// Les méthodes :

//      document.close() : ferme le document en écriture
//      document.open() : ouvre le document en écriture
//      document.write() : écrit dans le document
//      document.writeln() : écrit dans le document et effectue un retour à la ligne
```

Navigator


```
//NAVIGATOR

// Un objet de type Navigator est un objet qui contient des informations sur le navigateur

// Une variable navigator, de type Navigator, est créée automatiquement à l'ouverture du
// navigateur

// Les propriétés :

    // appCodeName : le nom du navigateur
    // appMinorVersion : le type de version mineure
    // appName : le nom complet du navigateur
    // appVersion : la version du navigateur
    // cookieEnabled : indique si le navigateur autorise les cookies
    // language : contient le code langue du navigateur
    // onLine : indique si le navigateur est connecté à internet
    // platform : contient le nom du système d'exploitation
```

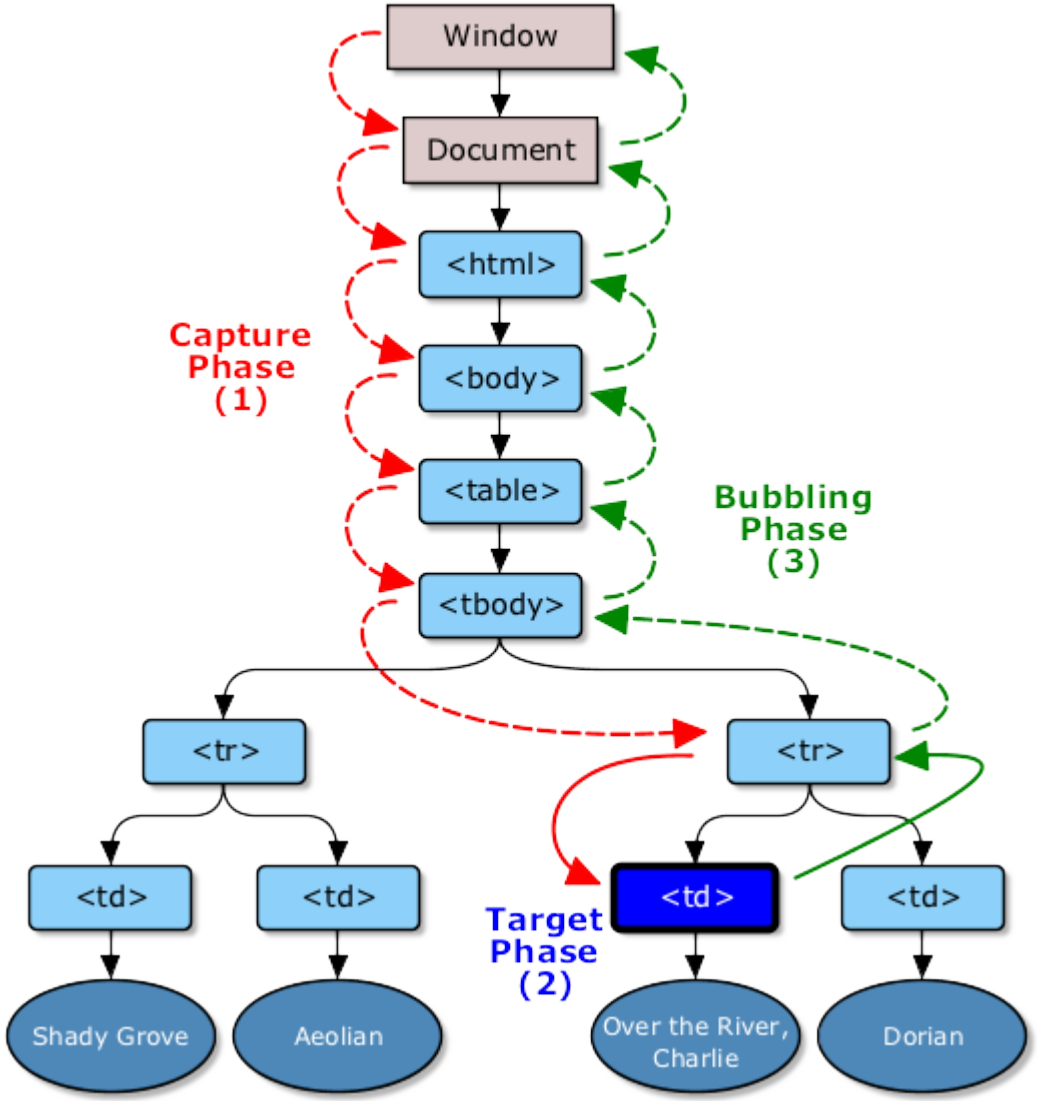
Navigator


```
// LES COOKIES

// Permettent de stocker de l'information sur le poste de l'internaute

// Sous forme clé = valeur

// Les valeurs ne doivent pas contenir d'espace ni de caractères spéciaux

// Un cookie est associé à un domaine donné. Seul celui-ci peut lire un cookie
// qui lui est associé

// Les cookies sont gérés via la propriété cookie de l'objet document
```

DOM


```
// LE DOM

// DOM : Document Object Model

// Le DOM est une API pour manipuler les documents HTML et XML :

    // définit la structure du document : hiérarchie d'objets.

    // définit des objets et types d'objets spécifiques.

    // définit la manière dont on accède aux éléments du document et comment on peut agir
    // dynamiquement sur eux.

// Le W3C propose une normalisation du DOM
```

Interprétation du code



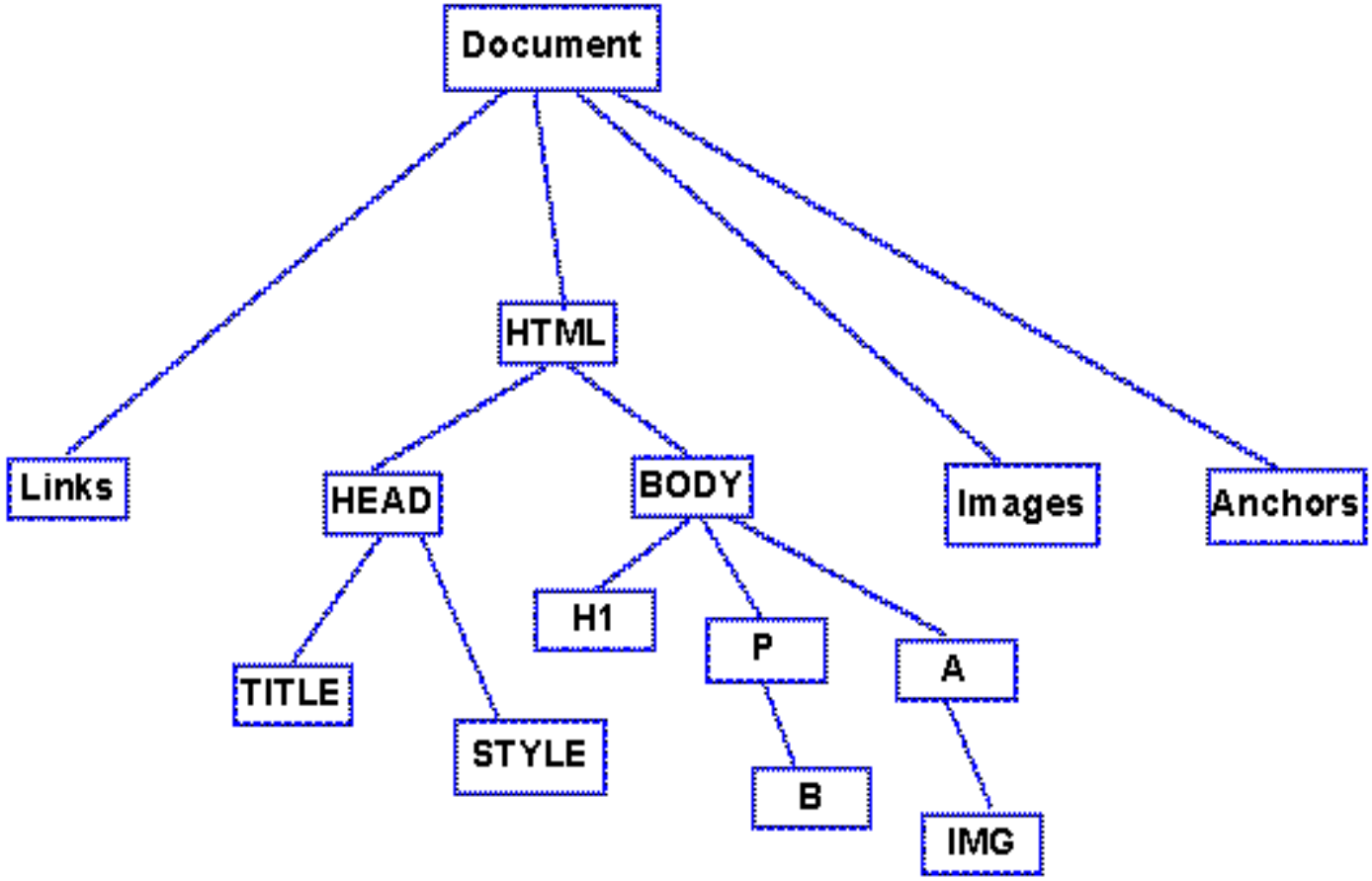
Document interface


```
// INTERFACE DOCUMENT

// Propriétés et méthodes :

    // documentElement : élément racine du document
    // createElement(tag) : crée un élément, renvoie le nœud créé
    // createTextNode(data) : crée un nœud texte, renvoie le nœud créé
    // getElementsByTagName(tagName) : renvoie la liste de tous les éléments tagName
    // getElementById(ID) : renvoie le nœud dont l'attribut id vaut ID
    // getElementsByClassName(class) : renvoie les noeuds dont l'attribut class vaut class
    // childNodes : liste des nœuds enfant
    // parentNode : le nœud parent
    // nextSibling / previousSibling : le prochain /précedent nœud
    // firstChild / lastChild : premier /dernier nœud enfant
    // nodeName : nom du nœud
    // nodeValue : valeur du nœud
    // appendChild(node) : ajoute un nœud enfant en dernière position
    // removeChild(node) : retire un nœud enfant
    // insertBefore(new, ref) : ajoute un nœud juste avant le nœud de référence
    // replaceChild(new, old) : remplace l'ancien nœud par le nouveau
    // cloneNode(bool) : retourne un clone du nœud
    //     true pour une copie incluant les fils
    // hasAttributes() : indique si le nœud a des attributs
    // hasChildNodes() : indique si le nœud a des fils
```

Document interface


```
// Un Element est un nœud représentant une balise

// Propriétés/méthodes principales :

// tagName : nom de la balise
// getAttribute(name) : renvoie la valeur de l'attribut
// setAttribute(name,value) : définit la valeur de l'attribut name
// getElementsByTagName(tagName) : renvoie la liste de tous les éléments
//     tagName qui sont fils de l'élément
// removeAttribute(name) : retire un attribut
// querySelector('cibleCSS'): renvoi la cible css
// querySelectorAll('cibleCSS'): renvoi les cibles CSS
```

DOM


Document
HTML
Links
HEAD
BODY
Images
Anchors
H1
P
A
TITLE
STYLE
B
IMG

Accéder aux éléments du DOM

Pour accéder aux éléments du DOM on peut cibler différents éléments, voici quelques exemples :

Tous les h2 de la page:
```
var elem = document.querySelectorAll(‘h2’) ;
```

Tous les éléments d’une class:
```
var elem.class = document.querySelectorAll(‘.class’) ;
```

Tous les id de la page:
```
var elem.identifiant = document.querySelectorAll(‘#id’) ;
```

Manipulation du document

Pour ajouter ou remplacer du contenu dans une page HTML

```
let contents = myElement.innerHTML;
```

Pour du texte on utilisera :

```
element.textContent = "ceci est un simple exemple de texte";
```

Accès aux attributs

Création d'un élément

```
var e = document.createElement('p');
```

Modifier un élément

```
e.style.textAlign = 'center';
```

Ajouter l'élément au parent

```
parent.appendChild(e);
```

Ajouter l'élément avant

```
element.parentNode.insertBefore(new_element,this);
```

Supprimer un élément

```
var e = document.getElementById('deleteMe');
e.parentNode.removeChild(e);
```

TP menu

Réalisez un menu dynamique

Modifiez votre menu pour que l’affichage des sous catégories se fassent avec du JavaScript

Informations diverses :

MDN Javascript

JavaScript.com

Différence entre let et var