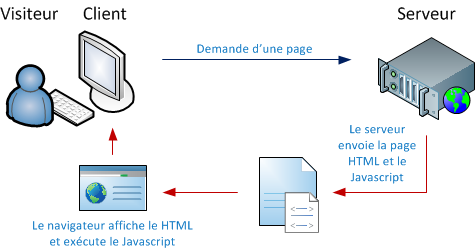
|  |  |  |
| --- | --- | --- |
| BTS SN1 | Développement Web | Lycée La Providence |
| Année 2017 | JavaScript | Amiens |
| Julien Langlacé | Cours S03CO7SN1 | Séquence S03SN1 |



Le JavaScript est un langage (orienté Object comme c++, Php 7, java) c’est un langage interprété par un client web ( ex navigateur internet ). Les fichiers JavaScript même s’ils sont stockés sur un serveur web, ne sont pas interprétés par le serveur. (Le code est donc envoyé directement via des requêtes http).



Un langage de programmation orienté objet est un langage qui contient des éléments, appelés **objets**, et que ces différents objets possèdent des caractéristiques spécifiques ainsi que des manières différentes de les utiliser. Le langage fournit des objets de base comme des images, des dates, des chaînes de caractères… mais il est également possible de créer soi-même des objets.

Si le JavaScript a été conçu pour être utilisé conjointement avec le HTML, mais il est aussi régulièrement utilisé pour réaliser des extensions pour différents programmes (firefox , chrome).

Le JavaScript est donc exécuté durant l’affichage d’un flux HTML. Il permet :

-Modifier dynamiquement le contenue d’une page en modifiant les balises <html>

-Interagir avec l’utilisateur en lançant des évènements (exemple souris qui bouge , clique etc)

Le JavaScript possède une syntaxe d’écriture proche de C et PHP.

- ; à la fin d’instruction, utilisation des boucles for foreach if else switch case etc…

-les variables ne sont pas typé comme en C mais doivent être déclaré contrairement à PHP

-les commentaires sont C ou PHP avec l’utilsation // ou /\* …… \*/

-Il peux être débugué dans Firebug

-il n’y a pas d’écho comme en php mais on utilise la fonction alert() , alert("coucou") ;

**I) Installation de JavaScript ;)**

Comme Javascript (JS) est un langage interprété, il n’y a pas besoin de le compiler. Il n’y a pas d’installation à faire. Par contre il faut respecter certain principe pour que tout fonctionne correctement.

1. **Ecriture du Code**

Le code doit être écrit de façon claire (bien indenté) et commenté avec des noms de variables corrects :

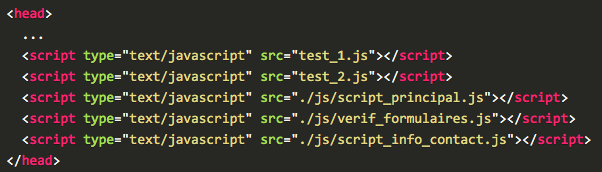
//exemple de nom de variable

MaVariableQuiSertAFaireQuelqueChose.

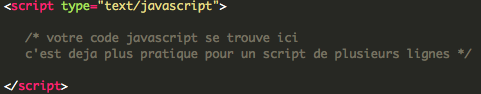
1. **Ranger son bout de code (son script)**

A la manière du CSS. On peut écrire ses scripts dans un fichier à part et en faire un lien dans le code HTML ou directement écrire entre 2 balise <script type="text/javascript"> </script>

Solution 1 : Un lien vers un fichier.js ou le code est ecrit



Solution 2 : Ouverture d’une balise script



Solution 3 : Injecté directement dans un évènement HTML ( ici onclick )



**II) Structure d’un code JS**

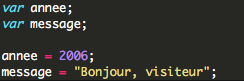
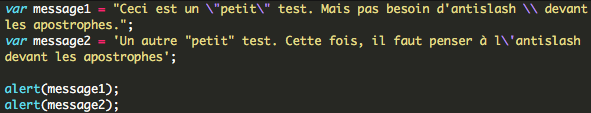
**A) les variables**

Comme dans chaque bon langage, nous avons besoin de travailler avec des variables.

En PHP on utilise le $ exemple **$Mavariable = 1**; en c/c++ on type la variable exemple **int MonNombre ;** pour déclarer un nombre **char MonCaractère ;** pour déclarer un caractère.

En Javascript on utilise le mot clé : **var** suivit du nom de la variable il n’y a pas de typage particulier sauf si on veux un tableau il faudra ecrire var MonTableau = new Array();

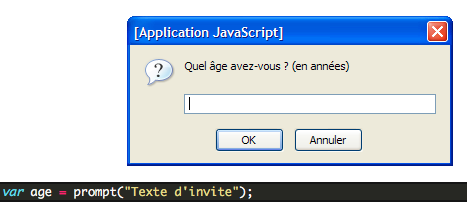
Exemple var **MavariableNombre = 0 ; var MonTableauDeChiffre[] ;**

**B) les opérations sur les variables**

|  |
| --- |
| resultat = resultat + X; // on ajoute X à la variable resultat  resultat += X; // on augmente la valeur de resultat de X  variable++; //Incrémentation de 1  variable--;//décrémentation de 1  variable += X; //On ajoute X à variable  variable -= X; //On retire X à la variable  variable \*= X; //On multiplie X à variable  variable /= X; //On divise variable avec X  variable %= X; //On fait la division euclidienne avec X  variable += 2; //On incrémente variable de 2 |

1. **L’interaction utilisateur**



Pour récupérer une information direct d’un utilisateur on utilise la fonction **prompt(message) ;**

**On peut aussi récupérer des données de l’utilisateur via les formulaires etc…**

1. **Structure de base d’un code JS**

Dans un premier temps le code est situé entre les deux balises script (cela permet au navigateur de savoir quel langage il doit interprété.

<script type="text/javascript">

</scritp>

A l’intérieur des ses balises on déclare nos variables en premier et on fait son algo.

A la fin on peut y écrire des fonctions que l’on créer et utilise.

<script type="text/javascript">

// Déclaration des variables

var MaVariable1 = 0 ;

var MaVariable2 = 0 ;

//Algo de traitement

MaVariable1 = 1 ;

MaVariable2 = 2 ;

MaVariable1 = MaVariable1 + MaVariable2 ;

Alert(MaVariable1) ;

</scritp>

**III) Les fonctions en JavaScript**

**A) Déclaration d’une fonction**

Comme en PHP et C/C++ on peut créer des fonctions pour réutiliser un bout de code.

Pour cela il faut la déclarer avec un nom, y mettre des paramètres d’entrée entre parenthèse.

Les fonctions peuvent retourner ou non une variable (voir fonction en C et PHP).

En JavaScript on utilise le mot clé : **function** pour déclarer une fonction.

Exemple : **Function NomDeMaFonction (paramètre1,paramètre2,etc…){**

**//code de la fonction**

var UneVariable = 2 ;

//on utilise le mot clé return si on veut renvoyer un résultat

return UneVariable ;;;

**}**

**B) Appel d’une fonction**

Pour appeler une fonction on l’appel juste avec son nom() et éventuellement les paramètres attendus. Si la fonction retourne une variable il faudra la placer dans une variable

Exemple :

<script language="javascript">

//appel de **DitBonjour** ()

***DitBonjour() ;***

//appel de **AjouteMoiUn** ()

var MonNombre = 2 ;

***MonNombre = AjouteMoiUn(MonNombre) ;***

//fonction 1 qui affiche une AlerteBox

function **DitBonjour**(){

alert("Bonnnnjourrrr") ;

}

//fonction 2 qui prend en entrée une variable , lui ajoute 1 et la retourne

function **AjouteMoiUn**(Mavariable){

Mavariable++ ;

Return Mavariable ;

}

</scritp>

**IV) Les Objets en javascript**

On n’expliquera pas comment programmer Objet ici ni comment cela fonctionne. Pourtant on va avoir besoin d’utiliser des objets en javascript pour pouvoir faire des choses intéressantes.

Un peu comme l’objet PDO en php pour les bases de données ;)

En javascript et contrairement à php pour utiliser une fonction (méthode) d’un objet on utilise le .point plutot que la fleche ->

// on execute la méthode mafonction() de l’objet qui porte le nom Monobjet

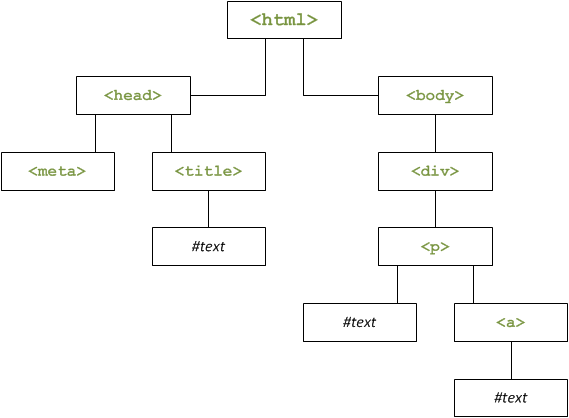
**Monobject.mafonction() ;**

**A) les Objets importants en HTML**

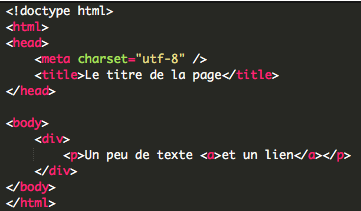
Pour JS, tous les éléments HTML sont des objets (images, liens, etc.) : on va donc pouvoir s'en servir pour en connaître ou en modifier les caractéristiques (comme l'adresse de l'image ou ses dimensions).

Le premier objet HTML par défaut en javascript est **Window** qui correspond à la fenêtre de votre navigateur. Il est implicite pas besoin de l’appeler pour l’utiliser. Window.alert("hello") ; peut s’écrire alert("hello") ; . L'objet **document** est un sous-objet de **window**, c’est l'un des plus utilisés. Et pour cause, il représente la page Web et plus précisément la balise **<html>.** C'est grâce à cet élément-là que nous allons pouvoir accéder aux éléments HTML et les modifier.

**B) les DOM (Document Object Model)**

 Le DOM est une API(**Application Programming Interface**) qui s'utilise avec les documents XML et HTML, et qui va nous permettre, via le JavaScript, d'accéder au code XML et/ou HTML d'un document. C'est grâce au DOM que nous allons pouvoir modifier des éléments HTML (afficher ou masquer un <div> par exemple),

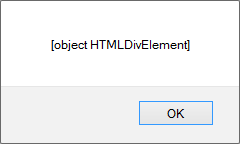
**En javaScript, on parlera d'élément HTML et non de balise**

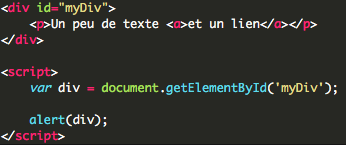
**** l'élément <html> contient deux éléments, appelés **enfants** : <head> et <body>. Pour ces deux enfants, <html> est l'élément **parent**. Chaque élément est appelé **nœud** (node en anglais). L'élément <head> contient lui aussi deux enfants : <meta> et <title>. <meta> ne contient pas d'enfant tandis que <title> en contient un, qui s’appelle #text. Comme son nom l'indique, #text est un élément qui contient du texte.

**C) Repérer et sélectionner les objets HTML**

L'accès aux éléments HTML via le DOM est assez simple. L'objet document possède trois méthodes principales : getElementById(), getElementsByTagName() et getElementsByName().

Exemple : ici on utilise la méthode getElementByID de l’objet document ( document représente la page web ) on récupère ainsi l’objet HTML div de son vrai nom : (HTMLDivElement). On ne peut pas faire d’alert dessus car la fonction **alert()** ne sait pas afficher des objets mais juste des strings.





**On peut aussi utiliser** querySelector() et querySelectorAll() en y glissant en paramètre la sélection d’une class css. (Nouveauté 2017 compatible IE8 et + )

Exemple : //sélectionne les balises de type <span> contenues dans les classes .item elles-mêmes

//contenues dans un élément dont l'identifiant est #menu.

var query = document.querySelector('#menu .item span'),

queryAll = document.querySelectorAll('#menu .item span');

A votre niveau vous ne connaissez pas toute la puissance de programmer en objet donc il faudra retenir toutes les méthodes et propriétés de chaque objet. Sauf si vous comprenez la notion d’héritage :



L’objet HTMLDivElement est un objet de type HTMLElement lui meme objet de type Element lui meme objet de type Node. Donc HTMLDivElement est un node il a en hérite donc ses propriétés et ses méthodes ;).

Donc si l’objet Node possède une propriété name, alors l’objet HTMLDivElement aura cette même propriété.

L’inverse n’est pas vrai. Si HTMLDivElement possède une propriété TailleDiv, l’objet node ne l’aura pas forcement.

D

**D) Editer les objets HTML**

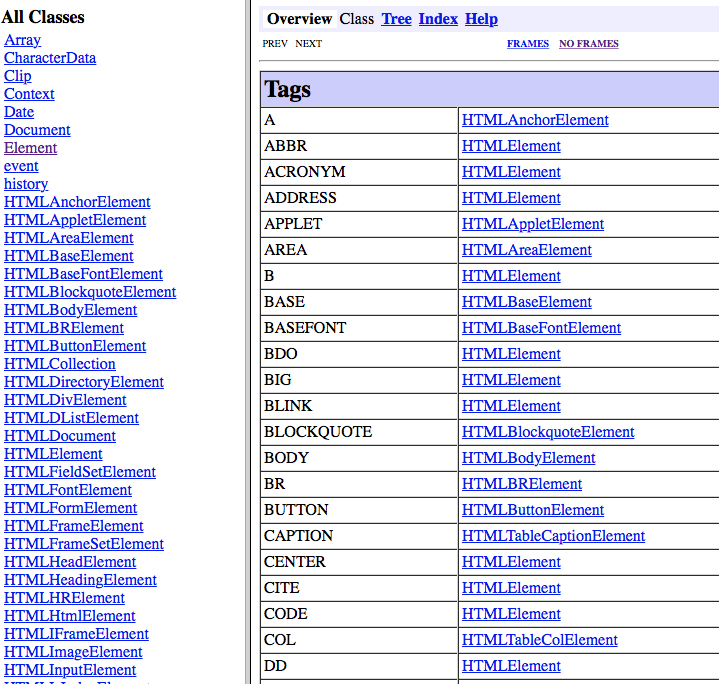
Une fois que l’on récupère un objet dans son script JS après l’avoir récupéré par exemple avec la méthode getElementById : var MonObjetHtml = document.getElementById('IdDeLaBalise');

On va pouvoir le manipuler (MonObjetHtml) en modifiant ses propriétés (ici les attributs d’une balise HTML) ou en appelant ses méthodes (fonctionnalités de l’objet en question)

Exemple : l'attribut href d'un <a> par exemple peut être modifié.

Il n’est pas possible de connaître toutes les propriétés et méthodes de chaque objet HTML existant. Il faut utiliser la documentation du DOM. Il existe pour cela google recherche DOM HTML exemple de site :

<http://www.krook.org/jsdom/>



1. **Tuto Langlacé pour modifier les objets HTML**

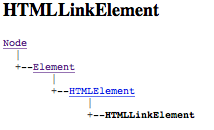
**Exemple nous souhaitons modifier dynamiquement la source d’un lien hypertext**

**<a id="MonLien" href="http://www.google.com">Go Google ?</a>**

1. **Identifier le type d’objet à modifier.**

**Google nous indique de les balises <a> sont des objets de type :** HTMLLinkElement (merci google)

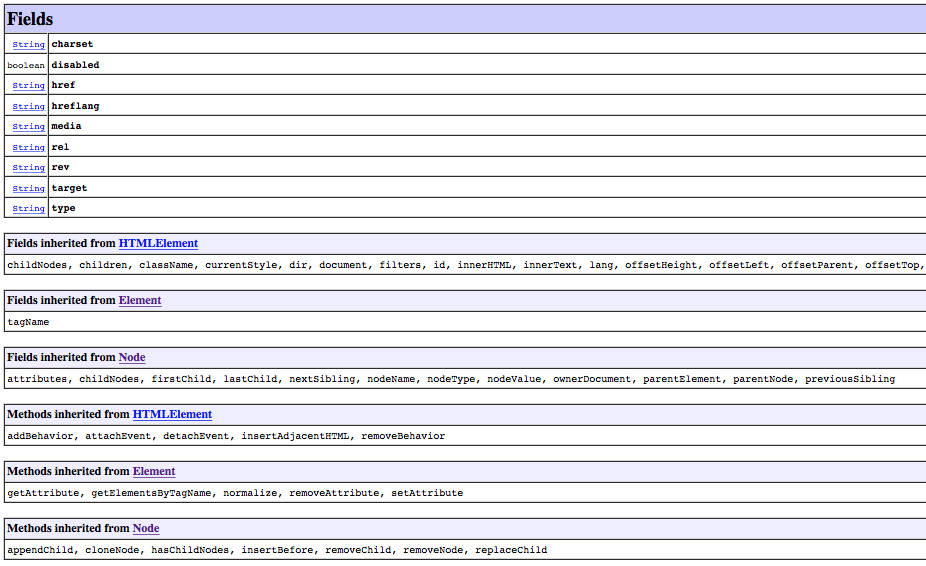
**2) Parcourir la documentation de l’objet en question sur le DOM**

****

La documentation nous indique que **HTMLLinkElement** est un objet de type **HTMLElement** lui meme de type **Element** lui meme de type **Node.**

**3) Repérer les Propriétés et les Méthodes de l’objet en question**

**(ici HTMLLinkElement)**

****

**"Fields" sont les propriétés, et "Méthods" sont les fonctions de l’objet que l’on peut utiliser. Remarquez que certaines "Methods" sont héritées des objets parents "Node" , "Element"…**

**A votre niveau vous ne le savez peut être pas encore mais certaines propriétées sont privées et ne peuvent être modifiées directement. Il faut passer par une méthode de l’objet pour la modifier.**

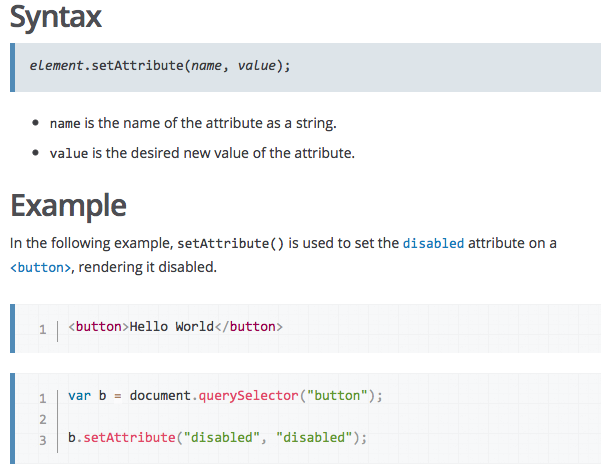
**La propriété qui nous intéresse à modifier ici est href**

**<a id="MonLien" href="http://www.google.com">Go Google ?</a>**

**Il faut rechercher la méthode que permet de modifier ce champs. (ici setAttribute)**

**Google recherche ou ce magnifique site complet Firefox :**

[**https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement**](https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement)

****

**4) Ecrire le script JavaScript**

**<a id="MonLien" href="http://www.google.com">Go Google ?</a>**

<script type="text/javascript">

// On recherche notre objet <a> qui a iD = **MonLien**

var VariableObjetLien = document.getElementById(‘**MonLien’**);

// On édite l'attribut « href »

VariableObjetLien.setAttribute('href', 'http://www.facebook.com') ;

</script>

**V) Les Evènements en JavaScript**

**A) Définition d’un évènement**

Les événements permettent de déclencher une fonction selon qu'une action s'est produite ou non. Un événement peu être rattaché à un élément de votre page Web (par exemple, une balise<div>) pour faire en sorte de déclencher un code JavaScript lorsque l'utilisateur fera une action sur l'élément en question.

Exemple d’évènement

|  |  |
| --- | --- |
| onclick | Cliquer (appuyer puis relâcher) sur l'élément |
| ondblclick | Double-cliquer sur l'élément |
| onmouseover | Faire entrer le curseur sur l'élément |
| onmouseout | Faire sortir le curseur de l'élément |
| onmousedown | Appuyer (sans relâcher) sur le bouton gauche de la souris sur l'élément |
| onmouseup | Relâcher le bouton gauche de la souris sur l'élément |
| onmousemove | Faire déplacer le curseur sur l'élément |
| onkeydown | Appuyer (sans relâcher) sur une touche de clavier sur l'élément |
| onkeyup | Relâcher une touche de clavier sur l'élément |
| onkeypress | Frapper (appuyer puis relâcher) une touche de clavier sur l'élément |
| onfocus | « Cibler » l'élément |
| onblur | Annuler le « ciblage » de l'élément |
| onchange | Changer la valeur d'un élément spécifique aux formulaires (input,checkbox, etc.) |

Attention certains évènements sont bloqués par les navigateurs.

Il existe aussi des évènements spéciaux pour navigateur mobile et tablette.

Pour ajouter un événement sur un élément HTML il faut faire comme ceci :

**<span onclick="maFonctionJS(monArugument1,monArgument2,etc…);">**

**Contenu de la balise html**

**</span>**

-On rajoute l’événement ici **onclick** suivit de =" "

-On place dans les " " le nom d’une fonction JavaScript que vous avez créé.

**B) Les évènements dans le DOM Version 1**

Il est possible d’ajouter un événement à un élèment du DOM dans un code javascript. Pour cela il suffit de rechercher l’objet HTML en question dans sont code et de lui ajouter un événement en passant par ses méthodes.

Exemple :

<!-- élèment div dans votre code html -->

<div id="clickme">Cliquez-moi !</div>

<!—-début du script JS -->

<script language="javascript">

//recherche de l’element via sont id

var elementDiv = document.getElementById('clickme');

//on ajoute à l’évenement onclick de l’elementDiv une fonction

elementDiv.onclick = function() {

alert("Vous m'avez cliqué !");

};

</script>

**Pour supprimer un événement sur un element on lui affect la fonction sans code**

element.onclick = function() {};

**C) Les évènements dans le DOM Version 2**

A la différence de la version 1 le DOM-2 permet de gérer plusieurs événements sur un même élément. Pour cela on ne touche plus à la propriété de l’élément mais on appelle une méthode à la place.

DOM1 : elementDiv.onclick = function() {alert("Vous m'avez cliqué !");};

DOMZ : elementDiv.addEventListener('click', function() {alert("Vous m'avez cliqué !");});

Grâce à la méthode addEventListener( ‘nomEvenement’,’fonction attribuée’) il est possible d’ajouter plusieurs événement pour un même élément html.

La méthode removeEventListener(‘nomEvenement’, maFunction); permet de retirer une fonction attribuée à l’élènement en question

**C) L’Objet Event de javascript**

L’objet Event nous permet de récupérer une multitude d'informations sur l'événement actuellement déclenché. Par exemple : les touches actuellement enfoncées, les coordonnées du curseur, l'élément qui a déclenché l'événement…

Cet objet est particulier car il est accessible que lorsqu'un événement est déclenché. Son accès ne peut se faire que dans une fonction exécutée par un événement, cela se fait de la manière suivante

DOM 1 :

// L'argument « e » va récupérer une référence vers l'objet « Event »

element.onclick = function(e) {

// Ceci affiche le type de l'événement (click, mouseover, etc.)

alert(e.type);

};

DOM 2:

// L'argument « e » va récupérer une référence vers l'objet « Event »

element.addEventListener('click', function(e) {

// Ceci affiche le type de l'événement (click, mouseover, etc.)

alert(e.type);

};

**Les fonctionnalités de l'objetEvent**

**type** est donc une propriété de **l’objet event** il en existe d’autre :

- target. permet de récupérer une référence vers l'élément dont l'événement a été déclenché

- clientX pour la position horizontale de la souris

- clientY pour la position verticale de la souris`

- keyup et keydown pour récupérer une touche du clavier

document.addEventListener('keydown', function(e) {down.innerHTML =e.keyCode;});