**Javascript**

Table des matières

[Introduction au javascript 2](#_Toc75260311)

[Les variables 2](#_Toc75260312)

[Les fonctions 2](#_Toc75260313)

[Les structures de code 2](#_Toc75260314)

[Les objets disponible 2](#_Toc75260315)

[Les tableaux 3](#_Toc75260316)

[Le contrôle du DOM par le javascript 3](#_Toc75260317)

[Recuperation d’un <document> 3](#_Toc75260318)

[Lecture/écriture sur <document> 3](#_Toc75260319)

[Recuperation d’un <element> 4](#_Toc75260320)

[Lecture/écriture sur les <element> 4](#_Toc75260321)

[Les objet javascript disponible 4](#_Toc75260322)

[Gestion des evenements 4](#_Toc75260323)

# Introduction au javascript

Le javascript peut à la fois être exécuter en front end, différent Framework existe tel que angular ainsi qu’en back end avec node.js. Ils existent aussi des gestionnaires de dépendance tel que npm ou encore des surcouche tel que typeScript.

Le javascript permet d’utiliser des objets interagissant entre eux, il est impératif (exécute un enchainement d’instruction) et fonctionnel (respecte la notion de fonction au sens mathématique : CAD qu’elle doit toujours renvoyer les mêmes résultats pour les même entré).

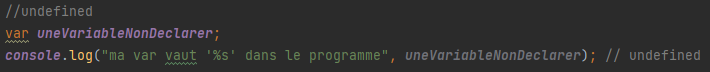
On écrit le chemin de notre js entre les balises <script src= «app.js»></script>

## Les variables

* Il existe des type primitif tel que boolean, null, undefined(jamais initialisé), number, string
* Il existe des objet disponible tel que Date, Math, Array , Object, Function, NaN…

Une image contenant texte

Description générée automatiquementPour déclarer une variable dont le type est définie en fonction de son utilisation :

Il existe une valeur **indefined** quand une variable à été déclarer mais non initialisé

Une image contenant texte, périphérique, mètre

Description générée automatiquementAttention pour tester une égalité strict on utilise === et pour l’inégalité exact !==

## Une image contenant texte Description générée automatiquementLes fonctions

## Une image contenant texte Description générée automatiquementLes structures de code

Une image contenant texte

Description générée automatiquement

## Les objets disponible

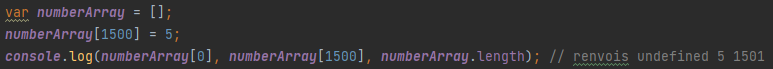
*Une image contenant texte

Description générée automatiquementExemple d’utilisation*

## Les tableaux

Une image contenant texte

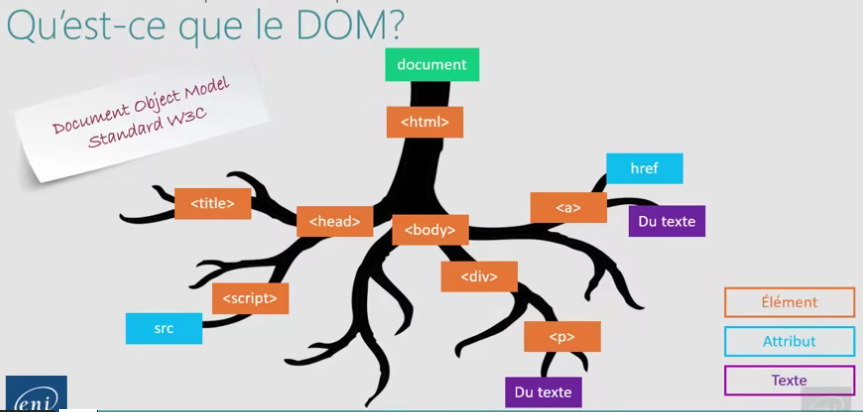
Description générée automatiquement



Une image contenant texte

Description générée automatiquement

# Le contrôle du DOM par le javascript



Le DOM (Document object model) est une représentation arborescente d’une page html. L’arborescence est composée de nœud qui comporte plusieurs éléments, les éléments comporte des attribut et/ou du texte.

## Recuperation d’un <document>

* **document.documentElement** ; // recupere l’element html
* **document.head**; // récupère l’élément head
* **document.title** ; // l’élément title
* **document.body** ; // l’élément body
* **document.form** ; // l’ensemble des formulaires

## Lecture/écriture sur <document>

|  |  |
| --- | --- |
| **Lecture**   * **document**.**getElementById()** ; // recupere un element par id * **document**.**querySelector()** ; // avec selecteur css * **document**.**getElementsByTagName()** ; // (div, head etc) * **document**.**querySelector()** ; | **Ecriture**   * **document.write()** ; // va écrire directement dans notre élément * **document.createElement() ;** // créer un élément dont les caractéristique sont fournis par les paramètre * **document.createAttribute()** ; // créer un attribut fournis en paramètre * **document.createTextNode()** ; // créer un nœud de type texte avec les info fournis en param |

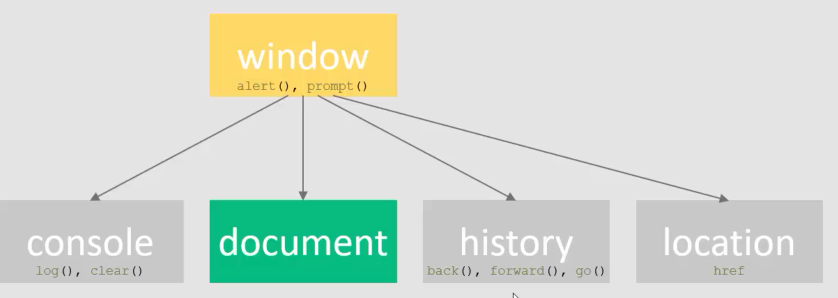
## Recuperation d’un <element>

* **monElement.id ;**
* **monElement.value ;**
* **monElement.attributes ;**
* **monElement.innerText ;**
* **monElement.innerHTML ;**
* **monElement.firstElementChild ;** // récupère le premier élément d’un nœud
* **monElement.lastElementChild;** // récupère le dernier élément d’un nœud
* **monElement.nextElementSibling ;** // allez sur l’élément suivant
* **monElement.previousElementSibling ;** // allez sur l’élément précèdent
* **monElement.parentElement ;** // allez sur l’élément parent

## Lecture/écriture sur les <element>

|  |  |
| --- | --- |
| **Lecture**   * **monElement.getAttribute() ;** * **monElement.hasAttribute() ;** * **monElement.hasAttributes() ;** * **monElement.focus ;** * **monElement.getElementsByTagName()** * **monElement.querrySelector** | **Ecriture**   * **monElement.insertAdjacentElement() ;** * **monElement.setAttribute() ;** * **monElement.appendChild() ;** * **monElement.replaceChild() ;** * **monElement.removeChild() ;** * **monElement.removeAttribute() ;** |

## Les objet javascript disponible



## Gestion des evenements

Une image contenant texte

Description générée automatiquementJavascript permet la gestion des événements suivant les actions de l’utilisateur. On dit qu’on s’abonne au évènement pour les écouter et déclencher un script à l’évènement

Une image contenant texte

Description générée automatiquement

# Développer des librairies en javascript

## Création d’objet et de classe

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquementUn objet en javascript peut être créer sans avoir de classe contrairement à un langage totalement objet tel que la java. Un objet javascript est un ensemble de paramètre et de méthode stocké dans une variable.

### Les classes

Une image contenant texte

Description générée automatiquement*Déclaration classique d’une classe en javascript*

Une image contenant texte

Description générée automatiquement

*Et avec le mot clé class (exactement la même chose, c’est ce qu’ont appel du « sucre syntaxique »)*

### Héritage

*Une image contenant texte

Description générée automatiquement*Une image contenant texte

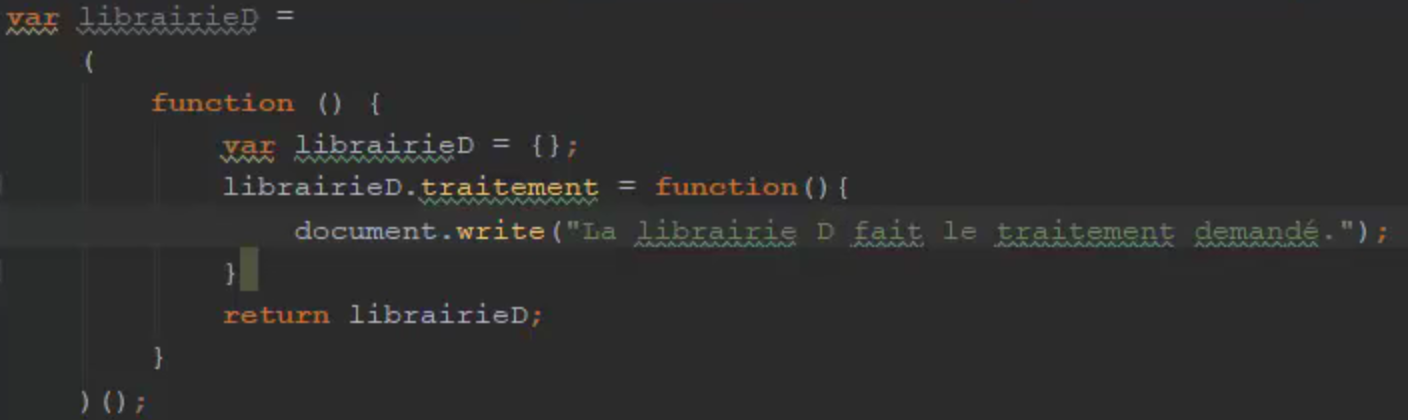
Description générée automatiquement

*Avec le mot clé extends*

## Conflit entre librairies

**

Si plusieurs librairies utilisent des nom de méthodes identique ce sera celle qui sera renseigner en dernier dans le HTML qui l’emporte… Pour gérer les soucis de conflits entre librairies on va utiliser les fonctions anonymes ou l’encapsulation de fonction dans un objet qui portera le nom de la librairie.

**

Et pour exploiter cette librairie (ici dans le fichier html), ce qui limitera les conflits sur les noms de méthode :

Une image contenant texte

Description générée automatiquement

## La spécialisation des méthodes et les closures

On ne peut pas utiliser la surcharge de méthode en javascript, on utilisera l’encapsulation pour le faire : il faudra déclarer plusieurs variantes, mais ce n’est pas dutout pratique (maintenance, taille de code etc)

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquementOn peut utiliser une closure (une fonction qui retourne elle-même une fonction) pour optimiser notre substitution de surcharche.

Une image contenant texte

Description générée automatiquement

## Le stockage local

Permet de stocké des information en local sur le navigateur qui seront reutilisable : le localStorage (pas de date d’expiration) et le SessionStorage (durée de vie limité a la session)

Une image contenant texte

Description générée automatiquement

## Les traitements asynchrones

### callback

Les traitements asynchrones permettent d’avoir une application réactive (ex on commence à taper un mot et il y as une proposition d’auto complétion), de gérer les traitements longs (en ne bloquant pas l’applications).

Pour utiliser ces traitements on va utiliser les callbacks ou/et les promesses.

Un callback est la possibilité de passer une fonction en paramètre d’une autre fonction qui sera appeler le moment venu. Ici le setTimeOut est juste pour simuler le temps d’un traitement.

Une image contenant texte

Description générée automatiquement

### Promesse

Les promesses sont des alternatives au callback qui permettent de limiter l’interdépendance entre l’appelant et l’appeler. Le programme ci-dessous créer la même finalité que le script pour les callbacks.

On doit retourner une **Promise** qui renverra soit **reject** ou **resolve** qui seront capturer dans un **then catch** quand le traitement sera terminé. Le then est déclencher en cas de succès et catch en cas de reject

