

# 05.JavaScript & DOM

13 novembre 2023

## Développement web il3

### JavaScript & DOM

HE-Arc (DGR) 2022

### JavaScript hier

- Page web = HTML (+ CSS + JavaScript)
- Exécuté par le browser (client)
- Interprété, faiblement typé, OO
- Historiquement
  - Depuis Netscape 2 (1995, Brendan Eich)
  - Petites applications exécutées par le navigateur
  - DHTML : rollovers, validation de formulaires, ...

### JavaScript aujourd'hui

- Page web = HTML + CSS + **JavaScript**
- Compilation JIT
- HTML5, AJAX, bookmarklets
- One Page Apps
- Implémentations hors-browser
  - Node.js, Spidermonkey, Rhino
  - script d'app (Qt, Notepad++, ...)

- Langage cible de compilateurs : emscripten<sup>1</sup>, WebAssembly<sup>2</sup>
- Embarqué : Espruino<sup>3</sup>, robotique : Node Bots<sup>4</sup>, CylonJS<sup>5</sup>
- Applications Desktop : Electron<sup>6</sup>, sciter<sup>7</sup>

## \*Script

- ECMAScript : Norme depuis 1997
  - Juin 2023 : ECMA-262 14th edition<sup>8</sup>
  - Support<sup>9</sup> des différentes implémentations
  - Conversions avec BabelJS<sup>10</sup>
- JavaScript : implémentation Firefox (réf. MDN)
- Variantes (à transpiler) :
  - Typescript<sup>11</sup> : variante fortement typée, avec des classes (MS)
  - Coffeescript<sup>12</sup>
    - \* sucre syntaxique
    - \* compilé -> js

## JavaScript

- Différentes implémentations<sup>13</sup> : navigateur, srv, apps, ...
- Permissif : du mauvais code est peu maintenable
  - Design Patterns<sup>14</sup>
  - Bonnes pratiques<sup>15</sup>
- Interface pour scripter le navigateur
  - Accès et modification du contenu via DOM

---

<sup>1</sup><https://emscripten.org/>

<sup>2</sup><http://webassembly.org/>

<sup>3</sup><http://www.espruino.com/>

<sup>4</sup><https://nodebots.io/>

<sup>5</sup><https://cylonjs.com/>

<sup>6</sup><https://electronjs.org/>

<sup>7</sup><https://sciter.com/>

<sup>8</sup><https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>

<sup>9</sup><https://compat-table.github.io/compat-table/es2016plus/>

<sup>10</sup><https://babeljs.io/>

<sup>11</sup><https://www.typescriptlang.org/>

<sup>12</sup><http://coffeescript.org/>

<sup>13</sup>[https://en.wikipedia.org/wiki/List\\_of\\_ECMA\\_Script\\_engines](https://en.wikipedia.org/wiki/List_of_ECMA_Script_engines)

<sup>14</sup><https://addyosmani.com/resources/essentialjsdesignpatterns/book/>

<sup>15</sup><http://jstherightway.org/>

- Bookmarklets<sup>16</sup>, exemples<sup>17</sup>
- Requêtes HTTP (Fetch API, Xml Http Request)
- Développement d'applications complètes, parfois offline
- Langage de script généraliste (paquets npm)

## Caractéristiques du langage

- Orienté Objet par prototype
- Syntaxe proche de C, Java
- Faiblement typé :
  - Pas de déclaration, type déterminé par la dernière affectation
  - Risque : typo => nouvelle variable. Utiliser const et let
- Types :
  - Primitifs : Boolean Null Undefined Number String Symbol
  - Objets : Object Function
- Particularités
  - Prototypes<sup>18</sup>
  - Fermetures<sup>19</sup>
  - Promesses<sup>20</sup> (MDN<sup>21</sup>, Google<sup>22</sup>)

## Fonctions

- Pas de type de retour
- Possibilité de retourner ou non une valeur
- Sans retour, valeur spéciale : undefined
- Pas de surcharge (la dernière définie prime)
- function est un type
- Fonctions imbriquées, anonymes
- Fonctions globales :

---

<sup>16</sup><http://www.howtogeek.com/125846/the-most-useful-bookmarklets-to-enhance-your-browsing-experience/>

<sup>17</sup><http://www.hongkiat.com/blog/100-useful-bookmarklets-for-better-productivity-ultimate-list/>

<sup>18</sup>[https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Le\\_mod%C3%A8le\\_objet\\_JavaScript\\_en\\_d%C3%A9tail](https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Le_mod%C3%A8le_objet_JavaScript_en_d%C3%A9tail)

<sup>19</sup>[http://www.w3schools.com/js/js\\_function\\_closures.asp](http://www.w3schools.com/js/js_function_closures.asp)

<sup>20</sup><https://www.promisejs.org/>

<sup>21</sup>[https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global\\_Objects/Promise](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Promise)

<sup>22</sup><https://web.dev/articles/promises?hl=fr>

`escape()`, `unescape()`, `isFinite()`, `isNaN()`,  
`parseFloat()`, `parseInt()`, `Number()`, `String()`,  
`eval()`, ...

## JavaScript dans la page web

- Éléments `<script>` exécutés dans l'ordre de la page
- Conseillé de les placer en fin de page<sup>23</sup>
- Événements (`onclick`, `onerror`, `onsubmit`, ...)
  - Embarqués dans les balises (`onXXX`)

```
<div id="intro" onclick="change();" />
```

Utiliser DOM

```
<script type="text/javascript">
```

```
    document.getElementById("intro").onclick = change;
```

```
</script>
```

- Conseillé d'inclure le code (attribut `src`)

```
<script type="text/javascript" src="script02.js"></script>
```

`language="JavaScript"` est déprécié et type vaut par défaut `text/javascript`.

The type attribute gives the language of the script or format of the data. [...] The default, which is used if the attribute is absent, is "text/javascript".

HTML5: `script`<sup>24</sup>

---

<sup>23</sup><https://stackoverflow.com/questions/1638670/javascript-at-bottom-top-of-web-page>

<sup>24</sup><https://www.w3.org/TR/html5/scripting-1.html#the-script-element>

## Unobstrusive JS<sup>25</sup>

- Séparation JS...

```
document.addEventListener("DOMContentLoaded", function() {  
    document.getElementById('date').addEventListener("change", validateDate);  
});
```

- ...et HTML

```
<input type="text" name="date" id="date" />
```

- Dégradation élégante
  - Alternatives pour un browser ne supportant pas JS
- Accessibilité
  - Les fonctionnalités restent accessibles en cas d'erreur
- Utilisabilité
  - Le script doit faire gagner du temps, pas distraire

It is an incredibly popular mistake to use load where DOMContentLoaded would be much more appropriate, so be cautious.

MDN: DOMContentLoaded<sup>26</sup>

## Node.js<sup>27</sup> / Deno<sup>28</sup>

- Node.js : une implémentation hors navigateur
  - environnement d'exécution + bibliothèques
  - event driven, non-blocking IO -> scalable
  - V8 engine
  - scripts exécutables sans navigateur
  - npm<sup>29</sup> : gestionnaire de paquets
  - gulp : make js
- Exemples<sup>30</sup> d'applications

---

<sup>25</sup>[https://en.wikipedia.org/wiki/Unobtrusive\\_JavaScript](https://en.wikipedia.org/wiki/Unobtrusive_JavaScript)

<sup>26</sup><https://developer.mozilla.org/en/docs/Web/Events/DOMContentLoaded>

<sup>27</sup><https://nodejs.org>

<sup>28</sup>[https://www.reddit.com/r/node/comments/nx9qqr/deno\\_vs\\_nodejs\\_a\\_comparison\\_you\\_need\\_to\\_know/](https://www.reddit.com/r/node/comments/nx9qqr/deno_vs_nodejs_a_comparison_you_need_to_know/)

<sup>29</sup><https://www.npmjs.com>

<sup>30</sup><https://colorlib.com/wp/npm-packages-node-js/>

- gulp, grunt, bower, yarn
- browserify
- serveur http
- express, cordova, forever, dev, pm2, karma, sails, phantomjs
- Tuto<sup>31</sup>, Playground<sup>32</sup>

## DOM

- Document Object Model
- Représentation arborescente de la page
- Accessible depuis objet JS document
- Possibilité d'accéder au contenu de la page :
  - Lecture
  - Modification
  - Ajout
- JS peut donc modifier le contenu d'une page

## DOM

```
<html>
<head>
  <title>My title</title>
</head>
<body>
  <h1>A heading</h1>
  <a href="#">Link text</a>
</body>
</html>
```

## L'objet Document

- Trouver ou modifier des éléments
- Méthodes de Document

---

<sup>31</sup><https://www.tutorialspoint.com/nodejs/index.htm>

<sup>32</sup><https://runkit.com>



Figure 1: DOM tree

```
querySelector(), querySelectorAll(),  
getElementById(), getElementsByTagName(), getElementByClass(),  
createElement(), createTextNode()
```

- Méthodes de Node (appel depuis nœud parent)

```
insertBefore(child), appendChild(child),  
removeChild(child), replaceChild(new,old)
```

## Ajouter un noeud

```
function addNode() {  
    var inText = document.getElementById("textArea").value;  
    var newText = document.createTextNode(inText);  
  
    var newGraf = document.createElement("p");  
    newGraf.appendChild(newText);  
  
    var docBody = document.getElementsByTagName("body")[0];  
    docBody.appendChild(newGraf);  
}
```

- Création du nouveau nœud :
  - newText contient le texte à ajouter
  - newGraf est un élément p qui contient le texte
- Ajout du nœud comme une feuille de body :
  - Sélection du parent (le premier noeud body)
  - Ajout du nouveau nœud depuis son parent

## Supprimer un nœud

```
function delNode() {  
    var allGraf = document.getElementsByTagName("p");  
  
    if (allGraf.length > 1) {  
        var lastGraf = allGraf.item (allGraf.length-1);  
        lastGraf.parentNode.removeChild(lastGraf);  
    }  
    else {
```



```

        console.error("Nothing to remove!");
    }
}

```

- Sélection du nœud à supprimer :
  - allGrafS contient tous les éléments p
  - lastGraf contient le dernier du tableau allGrafS
- Suppression :
  - Suppression du nœud sélectionné depuis son parent<sup>33</sup>

## Insérer un nœud

```

function insertNode() {
    var newText = document.createTextNode("New Text");
    var newGraf = document.createElement("p");
    newGraf.appendChild(newText);

    var divMod = document.getElementsByTagName("div")[0];
    var allGrafS = divMod.getElementsByTagName("p");
    var oldGraf = allGrafS.item(0);           // position

    divMod.insertBefore(newGraf, oldGraf);
}

```

- Création du nouveau nœud :
  - allGrafS contient tous les éléments p
  - lastGraf contient le dernier du tableau allGrafS
- Insertion :
  - Recherche du parent
  - Recherche du frère gauche
  - Insertion depuis le parent

## Avec jQuery

- Création et ajout :

---

<sup>33</sup><https://developer.mozilla.org/en-US/docs/Web/API/Node/parentNode>

```
var noeud = $('<p>Nouveau texte</p>'); // create node
$("body").append(noeud); // après le dernier fils
```

- Sélection et Suppression :

```
var noeud = $("p"); // select node(s)
noeud.remove();
```

## Références

- Une réintroduction à JavaScript<sup>34</sup>
- How does it feel to learn JS in 2016<sup>35</sup>
- Référence MDN<sup>36</sup>
- Tutoriels The Modern JS Tuto<sup>37</sup> w3schools<sup>38</sup>
- Outils de développement Chrome et Firefox (F12, Ctrl+Shift I)
- Visualisation du DOM<sup>39</sup>
- Outils web
  - JSFiddle<sup>40</sup>
  - JSLint<sup>41</sup>

## Sources

---

<sup>34</sup>[https://developer.mozilla.org/fr/docs/Web/JavaScript/Une\\_r%C3%A9introduction\\_%C3%A0\\_JavaScript](https://developer.mozilla.org/fr/docs/Web/JavaScript/Une_r%C3%A9introduction_%C3%A0_JavaScript)

<sup>35</sup><https://hackernoon.com/how-it-feels-to-learn-javascript-in-2016-d3a717dd577f>

<sup>36</sup><https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference>

<sup>37</sup><https://javascript.info/>

<sup>38</sup><http://www.w3schools.com/js/>

<sup>39</sup><http://bioub.github.io/dom-visualizer/>

<sup>40</sup><https://jsfiddle.net/>

<sup>41</sup><http://www.jshint.com/>