05.JavaScript & DOM

27 avril 2021

Développement web dlm3

JavaScript & DOM

HE-Arc (DGR) 2019

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[1], WebAssembly[2]
- Embarqué : Espruino[3], robotique : Node Bots[4], CylonJS[5]
- Applications Desktop : Electron[6], [7]

*Script

- ECMAScript : Norme depuis 1997
  - Juin 2020 : ECMA-262 11th edition / 2019[8]
  - Support[9] des différentes implémentations
  - Conversions avec BabelJS[10]
- JavaScript : implémentation Firefox (réf. MDN)
- Variantes (à transpiler) :
  - Typescript[11] : variante fortement typée, avec des classes (MS)
  - Coffescript[12]
    * sucre syntaxique
    * compilé -> js

JavaScript

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

[1] https://github.com/kripken/emscripten/wiki
[2] http://webassembly.org/
[3] http://www.espruino.com/
[4] https://nodebots.io/
[5] https://cylonjs.com/
[6] https://electronjs.org/
[7] https://sciter.com/
[8] https://www.ecma-international.org/publications/standards/Ecma-262.htm
[9] http://kangax.github.io/compat-table/es2016plus/
[10] https://babeljs.io/
[11] https://www.typescriptlang.org/
[12] http://coffeescript.org/
[13] https://en.wikipedia.org/wiki/List_of_ECMAScript_engines
[14] https://addyosmani.com/resources/essentialjsdesignpatterns/book/
[15] http://jstherightway.org/

  - Bookmarklets[16], exemples[17]
  - 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 `var` et `let`
- Types :
  - Primitifs : `Boolean Null Undefined Number String Symbol`
  - Objets : `Object Function`
- Particularités
  - Prototypes[18]
  - Fermetures[19]
  - Promesses[20] (MDN[21], Google[22])

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 :

[16] http://www.howtogeek.com/125846/the-most-useful-bookmarklets-to-enhance-your-browsing-experience/
[17] http://www.hongkiat.com/blog/100-useful-bookmarklets-for-better-productivity-ultimate-list/
[18] https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Le_mod%C3%A8le_objet_JavaScript_en_d%C3%A9tails
[19] http://www.w3schools.com/js/js_function_closures.asp
[20] https://www.promisejs.org/
[21] https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Promise
[22] https://developers.google.com/web/fundamentals/getting-started/primers/promises

```
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[23]
- Evénements (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[24]

[23] https://developer.yahoo.com/performance/rules.html#js_bottom=
[24] https://www.w3.org/TR/html5/scripting-1.html#the-script-element

Unobstrusive JS[25]

- 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[26]

Node.js[27]

- Node.js : une implémentation hors navigateur
  - environnement d'exécution + bibliothèques
  - event driven, non-blocking IO -> scalable
  - V8 engine
  - scripts exécutables sans navigateur
  - npm[28] : gestionnaire de paquets
  - gulp : make js
- Exemples[29] d'applications

[25] https://en.wikipedia.org/wiki/Unobtrusive_JavaScript
[26] https://developer.mozilla.org/en/docs/Web/Events/DOMContentLoaded
[27] https://nodejs.org
[28] https://www.npmjs.com
[29] https://colorlib.com/wp/npm-packages-node-js/

  - gulp, grunt, bower, yarn
  - browserify
  - serveur http
  - express, cordova, forever, dev, pm2, karma, sails, phantomjs

- Tuto[30], Playground[31]

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`

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

[30] https://www.tutorialspoint.com/nodejs/index.htm
[31] https://runkit.com

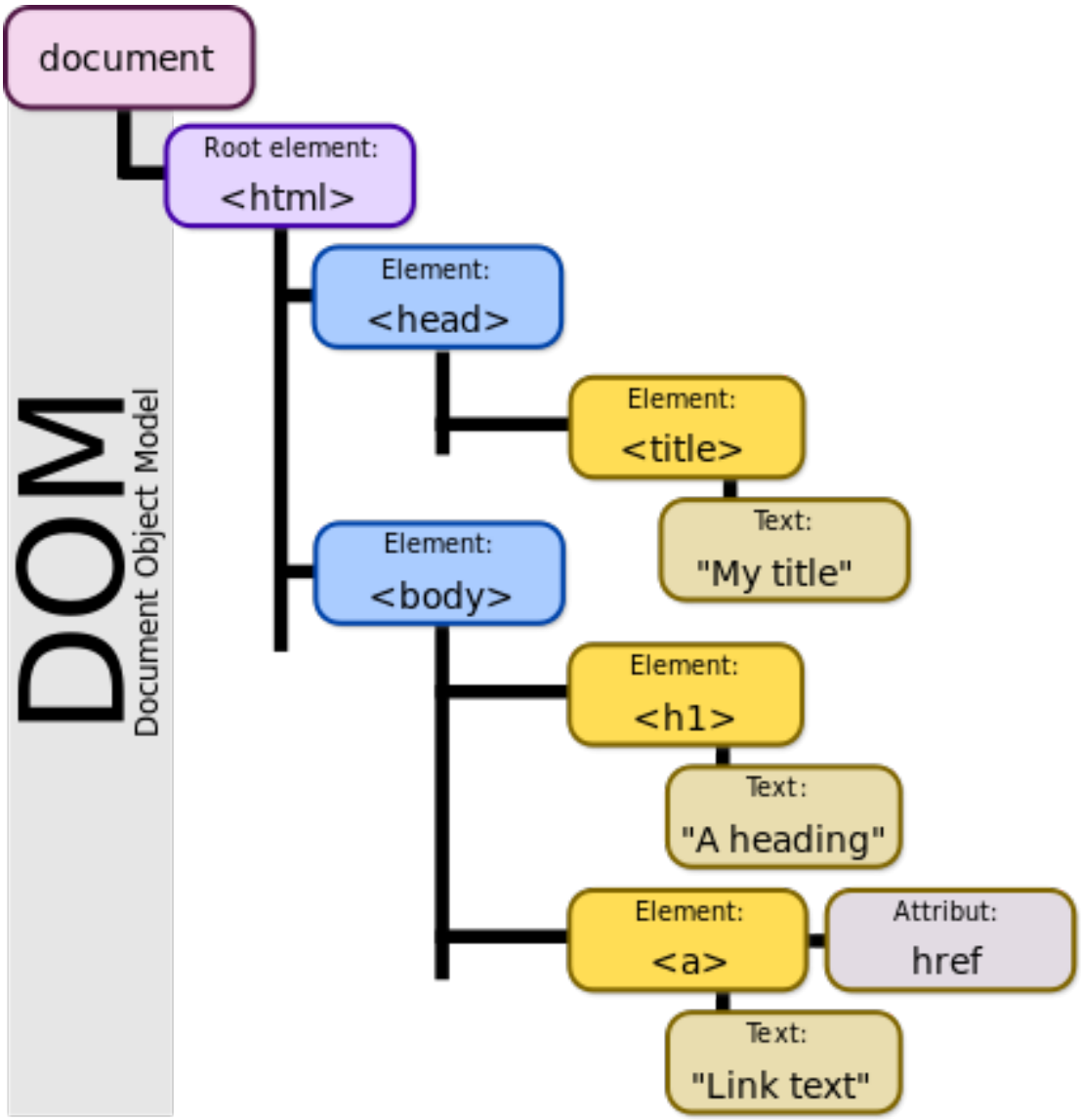

Fig. 1 : DOM tree

- 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 allGrafs = document.getElementsByTagName("p");

   if (allGrafs.length > 1) {
      var lastGraf = allGrafs.item (allGrafs.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 denier du tableau `allGrafs`
- Suppression :
  - Suppression du nœud sélectionné depuis son parent[32]

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 denier du tableau `allGrafs`
- Insertion :
  - Recherche du parent
  - Recherche du frère gauche
  - Insertion depuis le parent

Avec jQuery

- Création et ajout :

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

- Sélection et Suppression :

[32] https://developer.mozilla.org/en-US/docs/Web/API/Node/parentNode

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

Références

- Une réintroduction à JavaScript[33]
- How does it feel to learn JS in 2016[34]
- Référence MDN[35]
- Tutoriels The Modern JS Tuto[36] w3schools[37]
- Outils de développement Chrome et Firefox (F12, Ctrl+Shift I)
- Visualisation du DOM[38]
- Outils web
  - JSFiddle[39]
  - JSLint[40]

Sources

[33] https://developer.mozilla.org/fr/docs/Web/JavaScript/Une_r%C3%A9introduction_%C3%A0_JavaScript
[34] https://hackernoon.com/how-it-feels-to-learn-javascript-in-2016-d3a717dd577f
[35] https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference
[36] https://javascript.info/
[37] http://www.w3schools.com/js/
[38] http://bioub.github.io/dom-visualizer/
[39] https://jsfiddle.net/
[40] http://www.jslint.com/