



express



# Programmation Web

Gildas Ménier [gildas.menier@univ-ubs.fr](mailto:gildas.menier@univ-ubs.fr)

# Programmation Web



Clé d'inscription : web\_2023

# Introduction

- Technologies et développement Web
- Base :
  - HTML & CSS, PHP
  - Mécanismes Client Serveur
  - HTTP (à connaître)
  - Sockets
  - Servlets
  - Bases de la programmation Java
  - Bases de la programmation C++
  - Fonctionnel (Java / Scala)
  - XML / XSLT (à connaître)
  - GIT !! <https://git-scm.com/> (à connaître)
- Conditions cours : voir programmation multi paradigme

# Introduction

**HTTP 2 & HTTP 1.1  
(+RFC, IETF, FYI etc)**

# JS Javascript



# Javascript : historique



- EcmaScript
- Mosaic → Netscape : liveScript
- liveScript → javascript
- 1995 : interpréteur dans Netscape
- Microsoft propose JScript dans IE
- Netscape rôle et demande standardisation

ECMA

European association for  
standardizing information and  
communication systems



# Javascript



<http://www.ecma-international.org/>

Application	Dénomination
Navigateurs de type <a href="#">Gecko</a> avec le moteur embarqué <a href="#">SpiderMonkey</a> , dont <a href="#">Mozilla Firefox</a>	<a href="#">JavaScript</a>
<a href="#">Internet Explorer</a>	<a href="#">JScript</a>
<a href="#">Opera</a>	ECMAScript, avec des extensions <a href="#">JavaScript</a> et <a href="#">JScript</a>
<a href="#">KHTML</a> based browsers, including KDE's <a href="#">Konqueror</a>	<a href="#">JavaScript</a>
Framework <a href="#">.NET</a> de Microsoft	<a href="#">JScript .NET</a> et <a href="#">Managed JScript</a>
<a href="#">Adobe Flash</a>	<a href="#">ActionScript</a>
<a href="#">Adobe Acrobat</a>	<a href="#">JavaScript</a>
General purpose scripting language	<a href="#">DMDScript</a>
<a href="#">OpenLaszlo Platform</a>	<a href="#">JavaScript</a>
<a href="#">iCab</a>	<a href="#">InScript</a>
Implémentation d' <a href="#">XML</a> dans les navigateurs basés sur <a href="#">Gecko</a> et les programmes embarqués comme <a href="#">SpiderMonkey</a>	<a href="#">E4X</a>

Correspondance ECMAScript
ECMA-262, édition 3 <sup>1</sup>
ECMA-262, édition 3 <sup>6</sup>
ECMA-262, édition 3
ECMA-262
ECMA-262, édition 3 <sup>2</sup>
ECMA-262, édition 3 <sup>3</sup>
ECMA-262, édition 4 <sup>4</sup>
ECMA-262, édition 3
ECMA-262
ECMA-262, édition 3 <sup>5</sup>
ECMA-262, édition 3
ECMA-357, édition 2

# Environnement



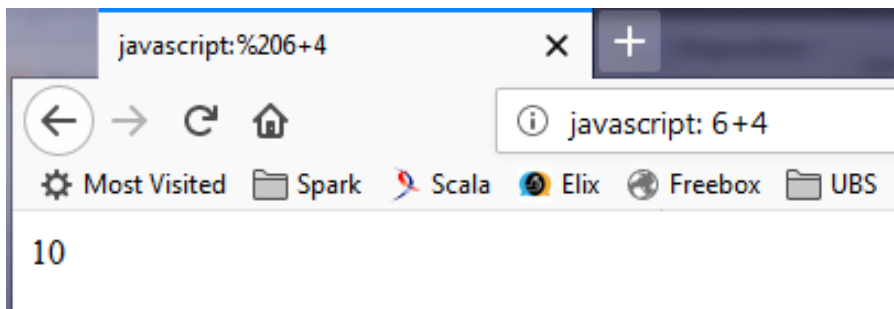
- Navigateur

- HTML

dans un fichier .html  
possible sans `<html>` etc

```
<script>  
document.write("hello")  
</script>
```

- javascript:



javascript:2+2

javascript:for(i=0; i<10; i++) alert(i);

url



# Environnement



- Java

- etc...

- Java 19 : jrunscript / GraalVM

- Java 7 : + Interpréteur Rhino

- Java 8 : Nashorn

- Lien avec Java

- Interpréteur en Java

- Node.js

- JIT

voir introduction à node.js



# Développement



- Eclipse + JSDT
- Komodo IDE
- Netbeans
- Sublime Text
- WebStorm
  - IntelliJ IDEA
- Notepad++
- TextMate
- Etc..

# WebStorm

A screenshot of the WebStorm IDE interface. The top-left pane shows a project tree for "web-starter-kit" with folders like "app", "scripts", and "styles". The main editor displays JavaScript code in "main.js" with a function "closeMenu()" and a "toggleMenu()" function. A tooltip is visible over the "closeMenu()" function call. The bottom pane shows the "Debugger" window with the "Frames" tab active, displaying the call stack for "anonymous(), main.js:46". The "Variables" tab shows local variables like "event" and "this".

Free for students:  
Professional developer tools  
from JetBrains

We were all students once...  
and some of us still are

Are you learning Java, PHP, Ruby, Python, JavaScript, Objective-C or .NET technologies?  
Or maybe you just plan to? Do it right from the start, with award-winning professional developer tools from JetBrains. And the best part: it's free of charge.

<b>ReSharper</b> Visual Studio extension for .NET	<b>IntelliJ IDEA</b> The most intelligent IDE for Java	<b>PhpStorm</b> IDE for Web & PHP
<b>dotTrace</b> .NET performance profiler	<b>RubyMine</b> IDE for Ruby and Rails	<b>WebStorm</b> Smart JavaScript IDE
<b>dotCover</b> .NET code coverage tool	<b>PyCharm</b> Powerful Python & Django IDE	<b>AppCode</b> Smart iOS/OS X IDE
<b>dotMemory</b> .NET memory profiler	<b>ReSharper C++</b> Visual Studio extension for C++	<b>CLion</b> Cross-platform C/C++ IDE

# Références

- Prérequis
  - CSS 3-4 (à connaître)
  - HTML 5 (à connaître)
  - HTTP 2/ 1.1/ RFC 2616 (à connaître)
- Livres & citations
  - **JavaScript: The Definitive Guide** David Flanagan
  - **Javascript Cookbook** Shelley Powers
  - **Secrets of the JavaScript Ninja** John Resig
  - **Javascript: The Good Parts** D.Crockford
  - **Functionnal programming** in Javascript Luis Atencio

# Introduction

- Industrie & formation :
  - classiquement
  - C, C++ en développement ..
  - Java ..
  - Objective-C et développements Android, MacOS etc..
  - Web : PHP avec un peu de Javascript ..
  - Langage coté serveur
  - Langage coté client
  - Plateformes hétérogènes : pleins d'outils, des langages différents
  - Compatibilités des matériels
  - Mises à jour (versions suivantes) et rétrocompatibilité
  - Passage au cloud difficile : hétérogénéité à gérer...
  - 10 ans d'études et 5-6 langages

# Introduction

- Industrie & formation :
  - Full stack & fonctionnel
  - Développement d'applications web
  - Développements d'applications desktop
  - Toutes les applications sont compatibles et multi plateformes
  - Pas de problème pour les changements de version
  - Il est même possible d'utiliser la version suivante même si pas encore supportée
  - Ios, Android, Windows, micro contrôleurs
  - Cloud & Web
  - Un seul langage : Javascript
  - 2 ans de formation : un seul langage

# Introduction

- « Oui mais si on connaît un peu de Java, ou de C (ou de C#), alors Javascript ne pose pas de problème » (?!)
- Javascript est un langage orienté **fonctionnel**
  - Pas (ou peu) Java
  - Pas (ou peu) C
  - Formation fonctionnelle académique : Scheme (par exemple)

```
(define (factorial n)
  (let loop ((fact 1)
            (n n))
    (cond ((= n 0) fact)
          (else (loop (* n fact) (- n 1))))))
```

- Applications industrielles ?
- Java & C : procédural impératif
- Évènementiel

# Introduction

- Autre manière de penser et de développer
- Caractéristiques de Javascript (que n'ont pas Java, C etc..)
  - Les fonctions sont des objets (première classe)
    - On peut faire des variables qui référencent une fonction
    - On peut passer des fonctions comme paramètres d'autres fonctions
    - Une fonction peut renvoyer une autre fonction
  - Les clôtures de fonction et de contexte
    - Une fonction qui fait référence à une variable hors de son bloc, peut en maintenir l'état jusqu'à son utilisation
  - Javascript est un langage orienté objet mais
    - Pas orienté « classe » (comme java ou C++)
    - Orienté « prototypes » (comme self)
    - Programmer Javascript en utilisant les classes comme en java est un risque
      - Logique différente
      - Des objets de même classe peuvent avoir des attributs et des méthodes différentes (?!)



# Introduction

- Javascript permet de gérer l'asynchronisme
  - Générateurs
    - Des fonctions capables de générer des résultats successifs, de s'interrompre et de reprendre au besoin, leur évaluation
  - Promesses
    - Mécanisme qui permet d'utiliser des variables avant qu'elles n'aient de valeur définie. Contrairement à Java / C avec le passage par valeur (une valeur est évaluée avant de participer à un calcul)
  - Proxies, Map, Modules, RegExp etc..
- Erreur : Programmer Javascript comme un Java un peu différent syntaxiquement
- Javascript est **très** différent de Java

## 8th Edition – ECMAScript 2017 [\[edit\]](#)

The 8th edition, officially known as ECMAScript 2017, was finalized in June 2017.<sup>[14]</sup> Its features include the `Object.values`, `Object.getPrototypeOf` functions for easy manipulation of Objects, `async/await` constructions and additional features for concurrency and `atomics`.<sup>[38][14]</sup>

## 9th Edition – ECMAScript 2018 [\[edit\]](#)

The 9th edition, officially known as ECMAScript 2018, was finalized in June 2018.<sup>[15]</sup> New features include rest/spread operators (`...identifier`), asynchronous iteration, `Promise.prototype.finally` and additions to `RegExp`.<sup>[15]</sup>

The spread operator allows for the easy copying of object properties, as shown below.

```
let object = {a: 1, b: 2}

let objectClone = Object.assign({}, object) // before ES9
let objectClone = {...object} // ES9 syntax

let otherObject = {c: 3, ...object}
console.log(otherObject) // -> {c: 3, a: 1, b: 2}
```

## 10th Edition – ECMAScript 2019 [\[edit\]](#)

The 10th edition, officially known as ECMAScript 2019, was published in June 2019.<sup>[16]</sup> Added features include, but are not limited to, `Array.prototype.flat`, `Array.prototype.flatMap`, changes to `Array.sort` and `Object.fromEntries`.<sup>[17]</sup> `Array.sort` is now guaranteed to be stable, meaning that elements with the same sorting precedence will appear in the same order. `Array.prototype.flat(depth=1)` flattens an array to a specified depth, meaning that all subarray elements (up to the specified depth) are included in the resulting array recursively.

## 11th Edition – ECMAScript 2020 [\[edit\]](#)

The 11th edition, officially known as [ECMAScript 2020](#), was published in June 2020.<sup>[10]</sup> In addition to new functions, this version introduced BigInts for arbitrary-sized integers, the [nullish coalescing operator](#), and the [globalThis object](#).<sup>[10]</sup>

BigInts are created either with the `BigInt` constructor or with the syntax `10n`, where "n" is placed after the number literal. Operations on BigInts and manipulation of integers beyond `Number.MAX_SAFE_INTEGER`, while Numbers are represented by a double-precision floating-point format in functions in `Math` are not compatible with BigInts; for example, exponentiation of BigInts must be done with the `**` operator.

# Introduction

- Javascript : ECMAScript
- Comité de standardisation
- ECMA : **European association for standardizing information and communication systems**
- ES6, ES7 (maintenant ES11)
- Javascript est exécuté dans un interpréteur
  - Dans un navigateur Web
  - Ou bien Node.js (sorte de JVM pour Javascript)
- Compatibilités des navigateurs ?
- <https://kangax.github.io/compat-table/es6/>

# Introduction

COMPAT ES

ECMAScript

5

6

2016+

next

intl

non-standard

compatibility table

Sort by Engine types

Show obsolete platforms

Show unstable platforms

V8

SpiderMonkey

Minor differences

Feature name	Current browser	Compilers/polyfills									
		Traceur	Babel + core-js <sup>[2]</sup>	Closure	TypeScript + core-js	es6-shim	Kong 4.14 <sup>[3]</sup>	IE 11	Edge 15	Edge 16	
Optimisation		97%	56%	71%	48%	59%	17%	5%	11%	96%	96%
<input type="checkbox"/> <a href="#">proper tail calls (tail call optimisation)</a>	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
Syntax											
<input type="checkbox"/> <a href="#">default function parameters</a>	7/7	4/7	4/7	5/7	5/7	0/7	0/7	0/7	7/7	7/7	7/7
<input type="checkbox"/> <a href="#">rest parameters</a>	5/5	4/5	3/5	2/5	4/5	0/5	0/5	0/5	5/5	5/5	5/5
<input type="checkbox"/> <a href="#">spread (...) operator</a>	15/15	15/15	13/15	12/15	4/15	0/15	0/15	0/15	15/15	15/15	15/15
<input type="checkbox"/> <a href="#">object literal extensions</a>	6/6	6/6	6/6	4/6	6/6	0/6	0/6	0/6	6/6	6/6	6/6
<input type="checkbox"/> <a href="#">for..of loops</a>	9/9	9/9	9/9	6/9	3/9	0/9	0/9	0/9	9/9	9/9	9/9
<input type="checkbox"/> <a href="#">octal and binary literals</a>	4/4	2/4	4/4	4/4	4/4	2/4	0/4	0/4	4/4	4/4	4/4
<input type="checkbox"/> <a href="#">template literals</a>	5/5	4/5	4/5	3/5	3/5	0/5	0/5	0/5	5/5	5/5	5/5
<input type="checkbox"/> <a href="#">RegExp "y" and "u" flags</a>	5/5	3/5	3/5	0/5	0/5	0/5	0/5	0/5	5/5	5/5	5/5

V8

[illegible]



Servers/runtimes									Mobile	
4%	66%	95%	59%	52%	97%	97%	2%	24%	99%	99%
S	Echo JS	XS6	JXA	Node 4 <sup>[5]</sup>	Node >=6.5 <7 <sup>[5]</sup>	Node >=8.7 <9 <sup>[5]</sup>	DUK 1.8	DUK 2.2	iOS >=10.3 <11	iOS 11
2	0/2	2/2	0/2	0/2	0/2	0/2	0/2	2/2	2/2	2/2
7	4/7	7/7	0/7	0/7	7/7	7/7	0/7	0/7	7/7	7/7
5	3/5	5/5	0/5	0/5	5/5	5/5	0/5	0/5	5/5	5/5
5	10/15	15/15	11/15	0/15	15/15	15/15	0/15	0/15	15/15	15/15
6	5/6	6/6	5/6	6/6	6/6	6/6	0/6	4/6	6/6	6/6
9	7/9	9/9	8/9	7/9	9/9	9/9	0/9	0/9	9/9	9/9
4	2/4	4/4	4/4	4/4	4/4	4/4	0/4	4/4	4/4	4/4
5	4/5	5/5	5/5	5/5	5/5	5/5	0/5	0/5	5/5	5/5
5	2/5	2/5	0/5	0/5	5/5	5/5	0/5	0/5	5/5	5/5
2	12/22	21/22	19/22	0/22	22/22	22/22	0/22	0/22	22/22	22/22
4	14/24	24/24	21/24	0/24	24/24	24/24	0/24	0/24	24/24	24/24
4	12/24	23/24	18/24	0/24	24/24	24/24	0/24	0/24	24/24	24/24
2	2/2	2/2	2/2	2/2	2/2	2/2	0/2	2/2	2/2	2/2
2	2/2	2/2	0/2	0/2	2/2	2/2	0/2	1/2	2/2	2/2
6	8/16	16/16	10/16	0/16	16/16	16/16	1/16	2/16	16/16	16/16

# Introduction

- Un des problèmes de approches classiques est la disponibilité / compatibilité des nouvelles versions du langage
- En Javascript, il est possible de travailler en version ES7 alors que seule la version ES6 est disponible
- Utilisation de **transpileurs** :
  - Trans(late) + (com)piler = **transpiler**
  - Le code JS (ES7) est traduit à la volée en une version acceptable pour ES6
  - Pas de problème de compatibilité
  - <https://github.com/google/tracur-compiler>
  - <https://babeljs.io/>

# Exemple : Traceur

- Navigateur ECMAScript 5 (ECMA 6 Inconnu)
- Pas de classes
- Page Web :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World!</title>
  </head>
  <body>
    <h1 id="message"></h1>
  </body>
</html>
```



- On souhaite injecter un message là



# Exemple : Traceur

- Code ECMA 6 (impossible en ECM5)

```
class Salut {  
  constructor(message) {  
    this.message = message;  
  }  
}
```

```
  bonjour() {  
    var element = document.getElementById('message');  
    element.innerHTML = this.message;  
  }  
};
```

```
var salut= new Salut('Hello!');  
salut.bonjour();
```

`document.querySelector('#message')`

# Exemple : Traceur

- Code à insérer :

```
<!DOCTYPE html>
```

```
<html>
```

```
...
```

```
<body>
```

```
<h1 id="message"></h1>
```

```
<script src="https://google.github.io/traceur-compiler/bin/traceur.js"></script>
```

```
<script src="https://google.github.io/traceur-compiler/bin/BrowserSystem.js"></script>
```

```
<script src="https://google.github.io/traceur-compiler/src/bootstrap.js"></script>
```

```
<script type="module" >
```

```
class Salut{
```

```
  constructor(message) {
```

```
    this.message = message;
```

```
  }
```

```
  bonjour() {
```

```
    var element = document.getElementById('message');
```

```
    element.innerHTML = this.message;
```

```
  }
```

```
};
```

# Exemple : Traceur

- Code à insérer :

```
var salut= new Salut('Hello!');  
salut.bonjour();  
</script>  
...  
</body>  
</html>
```

- Le code JS de traceur est déjà traduit en ECMA5
- Il contient le traducteur à la volée
- Node peut aussi le faire hors-ligne
- Interface Web

# Exemple : Traceur

← → ↺ 🏠 google.github.io/traceur-compiler/demo/repl.html#cl... Search ⬇ ⌵ 📄 ☑ ☆

⚙ Most Visited 📁 Spark 🐣 Scala 🦋 Elix 🌐 Freebox 📁 UBS 📁 Prog 📁 Graph 📁 Son 📁 Perso | 👤 M 📱 B 📄 SDoc 🖨 Kadenze 🖨 Freebox OS 🌐 lamarle 📺 Stages >>

**Source** **Traceur Transcoding Demo** **Options**

```
1 class Salut {
2   constructor(message) {
3     this.message = message;
4   }
5
6   bonjour() {
7     var element = document.getElementById('message');
8     element.innerHTML = this.message;
9   }
10 };
11
12 var salut= new Salut('Hello!');
13 salut.bonjour();
14 |
```

```
1 $traceurRuntime.ModuleStore.getAnonymousModule(function() {
2   "use strict";
3   var Salut = function() {
4     function Salut(message) {
5       this.message = message;
6     }
7     return ($traceurRuntime.createClass)(Salut, {bonjour: function() {
8       var element = document.getElementById('message');
9       element.innerHTML = this.message;
10      }}, {});
11   };
12   ;
13   var salut = new Salut('Hello!');
14   salut.bonjour();
15   return {};
16 });
17 //# sourceMappingURL=traceured.js
18
```

# Traceur Transcoding Demo

```
ntByID('message');
e;
```

```
1 $traceurRuntime.ModuleStore
2   "use strict";
3   var Salut = function() {
4     function Salut(message)
5       this.message = message;
6     }
7     return ($traceurRuntime
8       var element = document
9       element.innerHTML =
10        }, {}));
11   }();
12   ;
13   var salut = new Salut('He
14   salut.bonjour();
15   return {};
16 });
17 //# sourceMappingURL=traceured.js
18
```

```
.o/traceur-compiler/bin/E
'traceur-compiler/bin/Brc
-compiler/bin/BrowserSyst
-compiler/bin/BrowserSyst
```

## Options

- ☒ Evaluate
- ☒ Show generated code
- ☒ Show all options
- ☒ arrowFunctions
- ☒ blockBinding
- ☒ classes
- ☒ computedPropertyNames
- ☒ defaultParameters
- ☒ destructuring
- ☒ forOf
- ☒ generators
- ☒ numericLiterals
- ☒ propertyMethods
- ☒ propertyNameShorthand
- ☒ restParameters
- ☒ spread
- ☒ symbols
- ☒ templateLiterals
- ☒ unicodeEscapeSequences
- ☒ unicodeExpressions
- ☐ properTailCalls

Idem pour babeljs

[Learn ES2015](#)[Docs](#) ▾[Try it out](#)[Blog](#)[FAQ](#)[Team](#)[Donate](#)[Forum](#)

# Babel is a JavaScript compiler.

Use next generation JavaScript, today.

Put in next-gen JavaScript

```
[1, 2, 3].map(n => n ** 2);
```

Get browser-compatible JavaScript out

```
[1, 2, 3].map(function (n) {  
  return Math.pow(n, 2);  
});
```

[Check out our REPL to experiment more with Babel!](#)

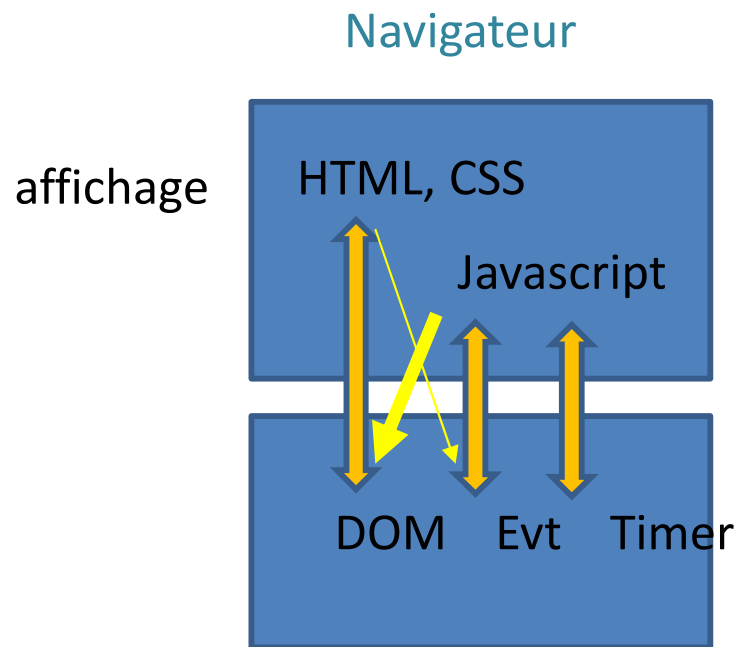
**Latest From Our Blog: Nearing the 7.0 Release**

# Introduction

- Deux environnements d'exécution pour Javascript
  - Le navigateur
    - L'interpréteur est intégré au navigateur (et donc figé)
    - Pas d'entrée/sortie fichier
    - Entrées sorties socket limitées
    - Performances dépendantes du navigateur
    - L'interface graphique dépend du navigateur
    - Généralement HTML, canvas, webgl ou librairies
  - Node.js
    - Pas de restriction
    - Mais pas (*a priori*) d'interface graphique (NW.js ou Electron)
    - Serveur

# Introduction

- Navigateur Web et Node.js

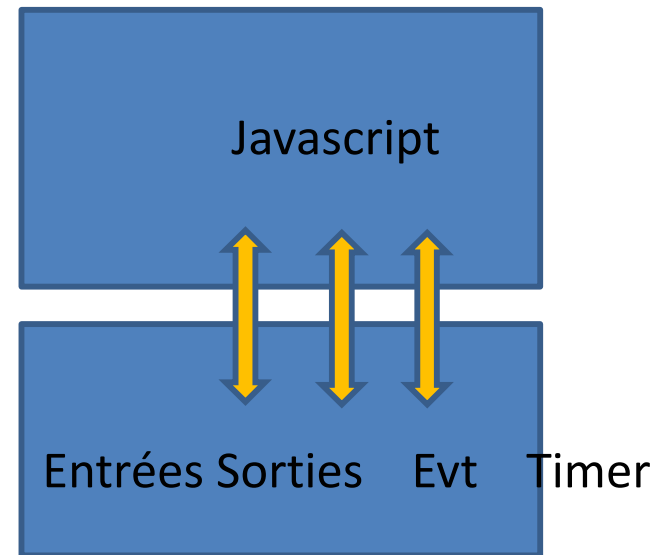


API Navigateur

Le navigateur donne ou pas accès à certaines fonctions, objets ou librairies

Application autonome (pas dans un navigateur)

Node.js



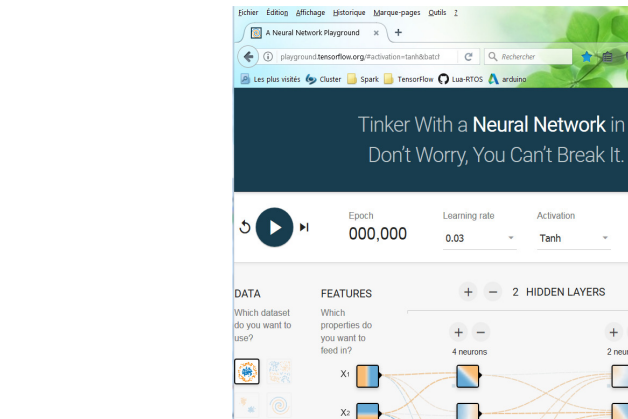
Node.js

Système extensible de librairies (NPM)



# Introduction

- DOM : Document Object Model
  - Hiérarchie d'objets qui représente le document HTML
  - Par exemple :



- *A priori*, pas de DOM coté Node.js
- Le javascript du navigateur modifie le DOM
- La modification du DOM change le document affiché
- L'affichage Node.js dépend des librairies disponibles
- Node.js ne s'exécute pas dans un navigateur
- Ligne de commande (comme la JVM)

```
C:\Users\menier>node programme.js_
```

# Introduction

- Les évènements
  - Déclenchent des fonctions javascript
  - Ils peuvent survenir n'importe quand
  - Leur nature dépend de l'environnement d'exécution

Navigateur

Node.js

Click souris

Changement d'un texte dans un formulaire

Réception d'information par le Web etc..

Entrées Sorties

Communication avec Internet

En fonction des librairies (clavier etc)

- *Asynchrone* : le code Javascript s'exécute sans prévoir de tester explicitement ces événements
- Dès qu'un événement a lieu, une fonction spécifique interrompt le code en cours d'exécution
- Cette fonction gère l'évènement et le réponse du code Javascript
- Puis le code reprend