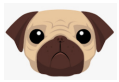



express



2023

Programmation Web

Gildas Ménier gildas.menier@univ-ubs.fr

2023

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)

Javascript

Université Bretagne Sud
ubs

G.Ménier

Javascript : historique

- EcmaScript
- Mosaic → Netscape : liveScript
- liveScript → javascript
- 1995 : interpréteur dans Netscape
- Microsoft propose JScript dans IE
- Netscape râle et demande standardisation
  - ECMA
  - European association for standardizing information and communication systems

Javascript

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

Application	Dénomination	Correspondance ECMAScript
Navigateurs de type Gecko avec le moteur embarqué SpiderMonkey, dont Mozilla Firefox	JavaScript	ECMA-262, edition 3 [1]
Internet Explorer	JScript	ECMA-262, edition 3 [6]
Opera	ECMAScript, avec des extensions JavaScript etJScript	ECMA-262, edition 3
KHTML based browsers, including KDE's Konqueror	JavaScript	ECMA-262
Framework .NET de Microsoft	JScript .NET et Managed JScript	ECMA-262, edition 3 [2]
Adobe Flash	ActionScript	ECMA-262, edition 3 [3] ECMA-262, edition 4 [4]
Adobe Acrobat	JavaScript	ECMA-262, edition 3
General purpose scripting language	DMDScript	ECMA-262
OpenLaszlo Platform	JavaScript	ECMA-262, edition 3 [5]
iCab	InScript	ECMA-262, edition 3
Implémentation d'XML dans les navigateurs basés sur Gecko et les programmes embarqués comme SpiderMonkey	E4X	ECMA-357, edition 2

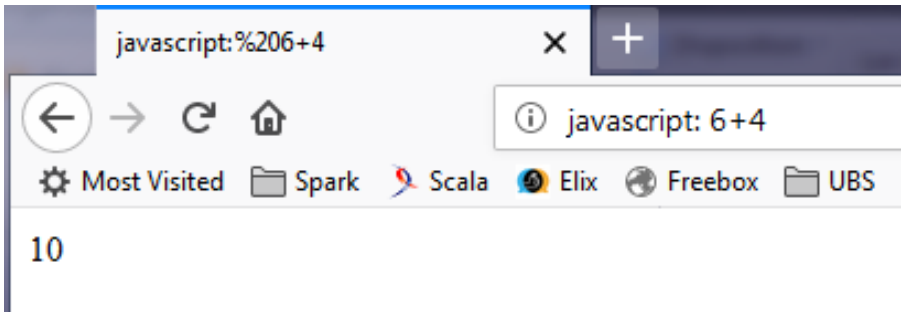
Environnement

- 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

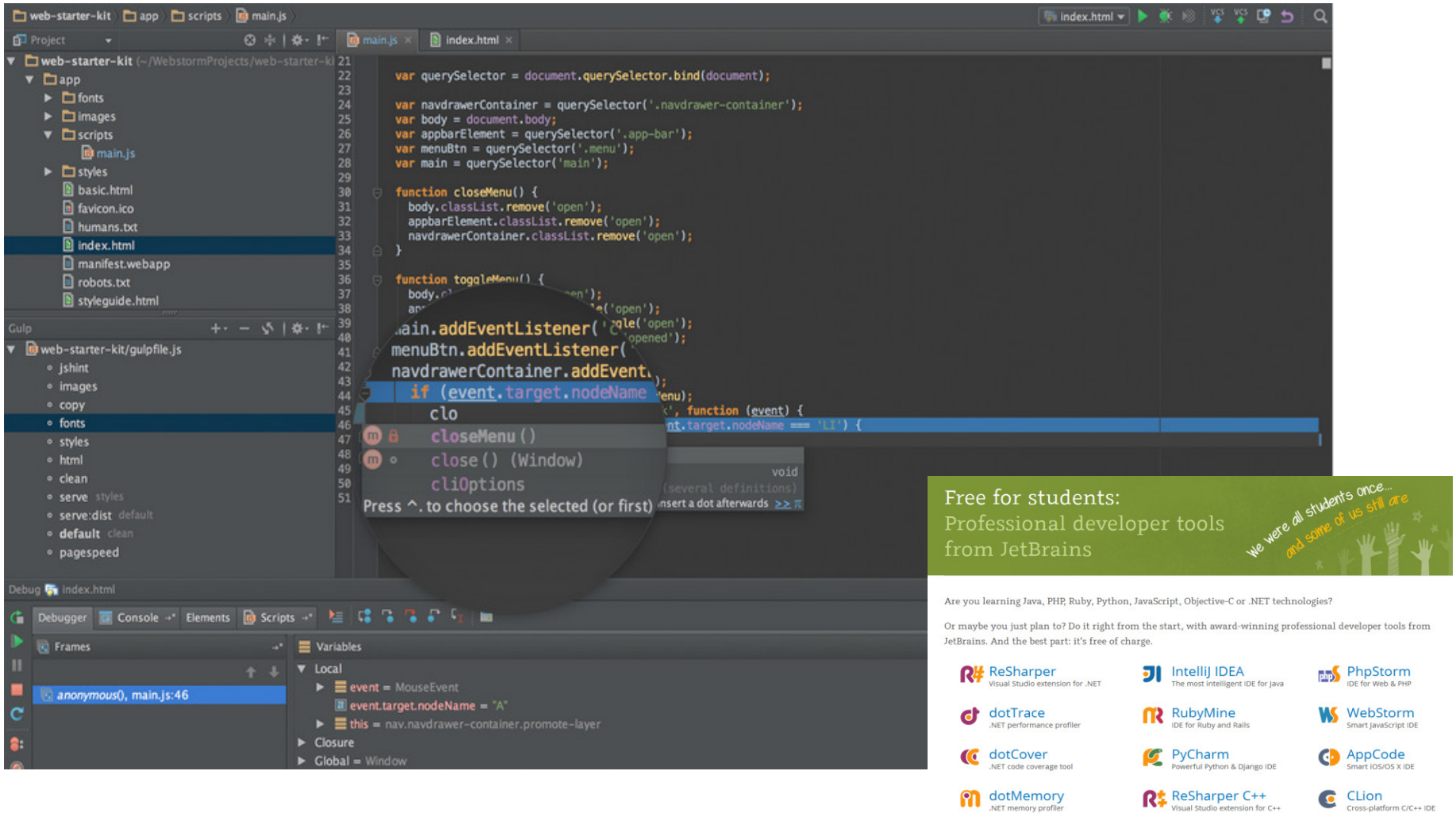
- Java
  - etc…
  - Java 19 : jrunscript / GraalVM
  - Java 7 : + Interpréteur Rhino
  - Java 8 : Nashorn
  - Lien avec Java
  - Interpréteur en Java
- Node.js
  - JIT

voir introduction à node.js

Développement

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

WebStorm



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 connait 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.[14] Its features include the `Object.valu` `Object.getOwnPropertyDescriptors` functions for easy manipulation of Objects, `async/await` constructions whi and additional features for concurrency and atomics.[38][14]

9th Edition – ECMAScript 2018 [edit]

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

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.[16] Added features include, but are not `Array.prototype.flat`, `Array.prototype.flatMap`, changes to `Array.sort` and `Object.fromEntries`.[

`Array.sort` is now guaranteed to be stable, meaning that elements with the same sorting precedence will appear in the `Array.prototype.flat(depth=1)` flattens an array to a specified depth, meaning that all subarray elements (up to th recursively.

11th Edition – ECMAScript 2020 [edit]

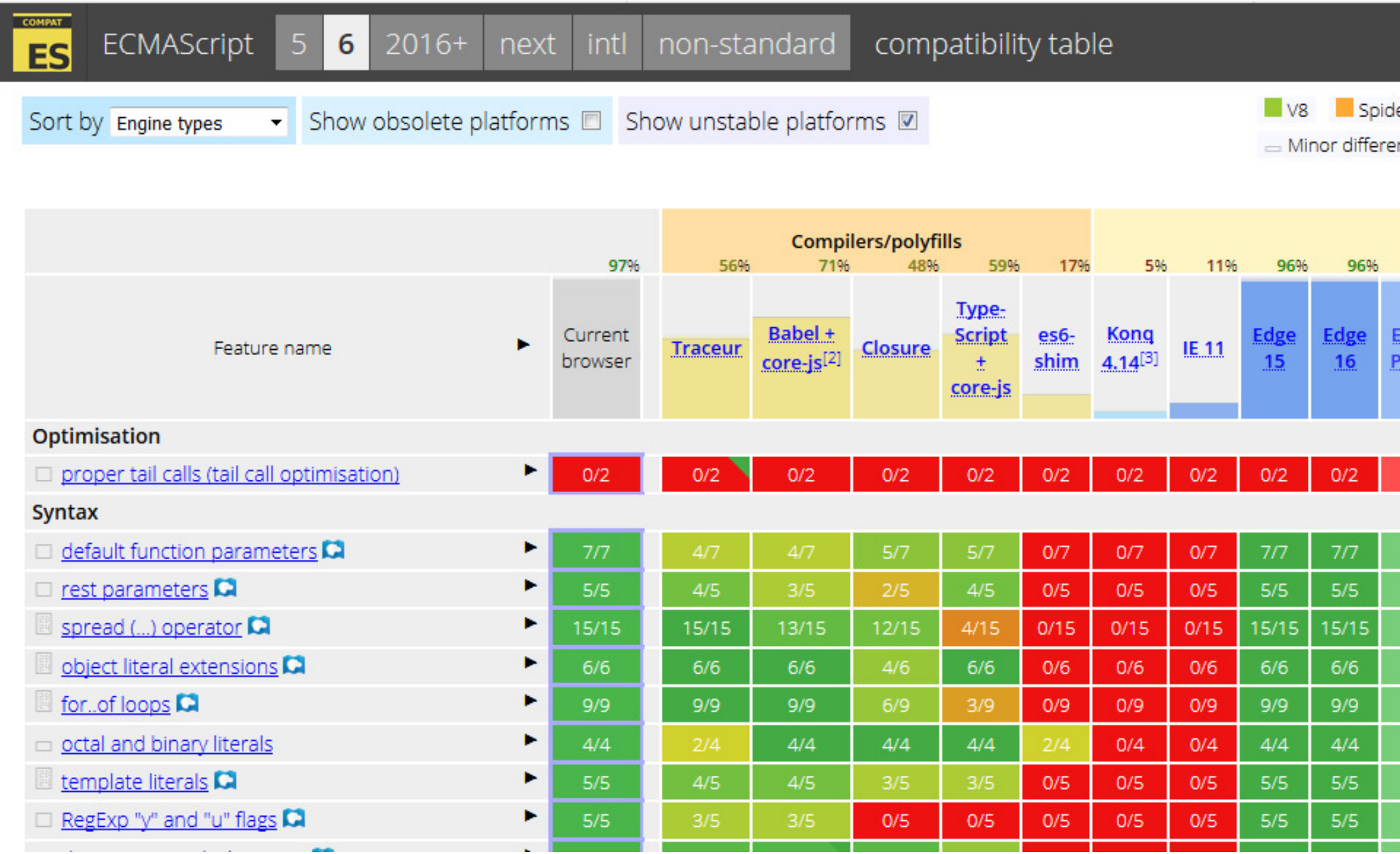
The 11th edition, officially known as ECMAScript 2020, was published in June 2020.[10] In addition to new functions, this ver type for arbitrary-sized integers, the nullish coalescing operator, and the globalThis object.[10]

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

Introduction

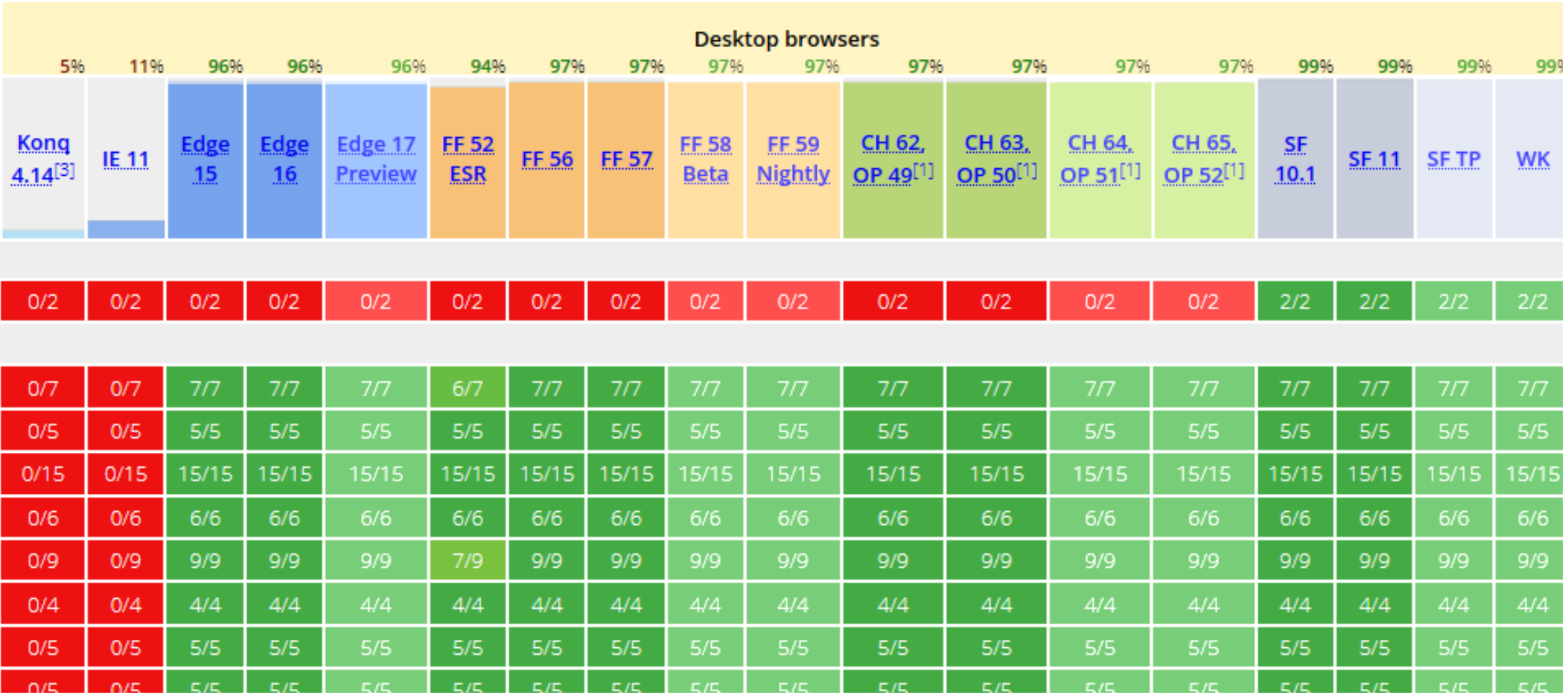
- Javascript : ECMAScript
- Comité de standardisation
- ECMA : **European association for standardizing information and communication systems**
- ES6, ES7 (maintenant ES11)
- Javascript est exécuté dans un interpréteur
  - Dans un navigateur Web
  - Ou bien Node.js (sorte de JVM pour Javascript)
- Compatibilités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

Introduction

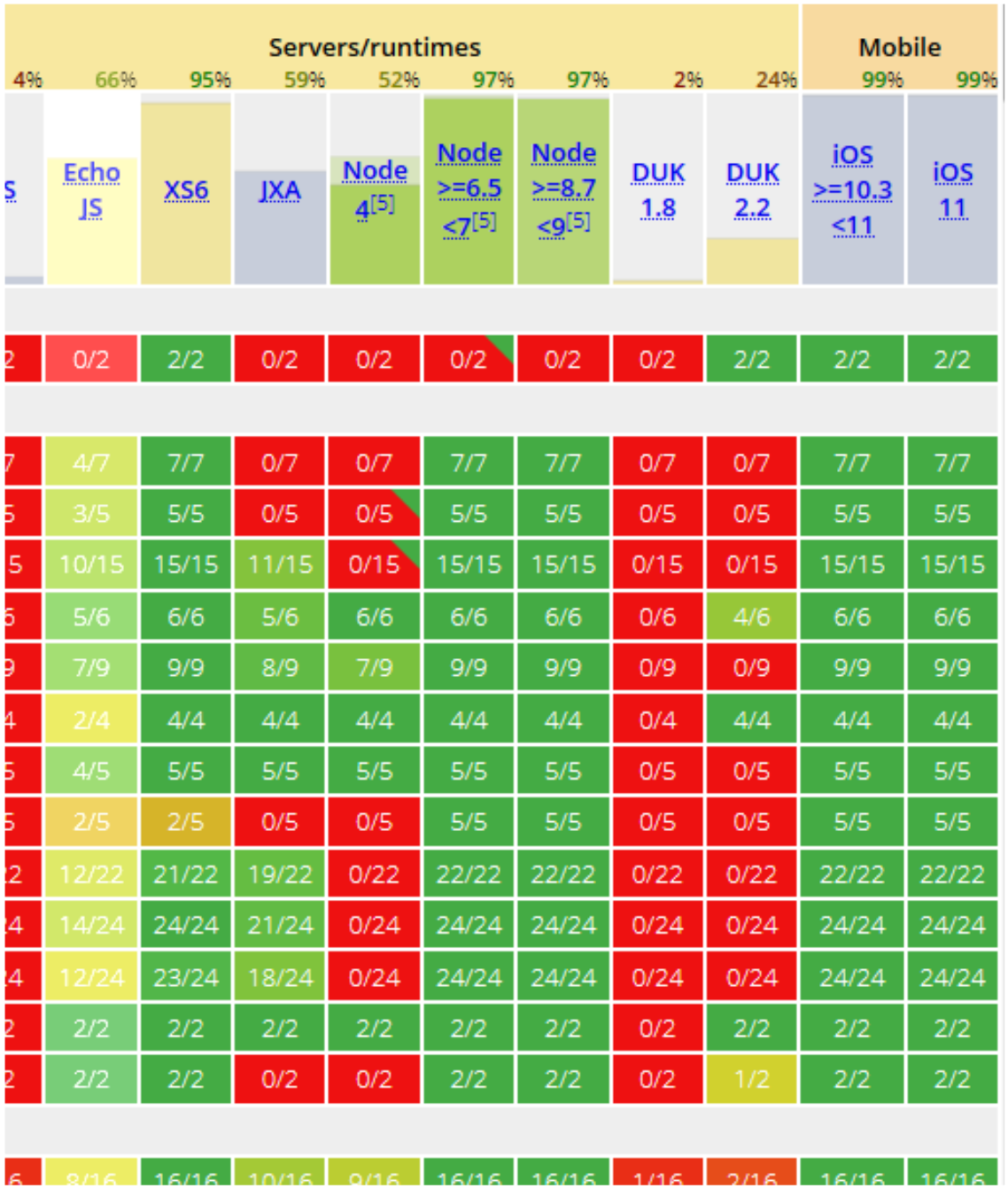


Introduction

SpiderMonkey V8



Desktop browsers																	
5%	11%	96%	96%	96%	94%	97%	97%	97%	97%	97%	97%	97%	97%	99%	99%	99%	99%
Konq 4.14[3]	IE 11	Edge 15	Edge 16	Edge 17 Preview	FF 52 ESR	FF 56	FF 57	FF 58 Beta	FF 59 Nightly	CH 62, OP 49[1]	CH 63, OP 50[1]	CH 64, OP 51[1]	CH 65, OP 52[1]	SF 10.1	SF 11	SF TP	WK
0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	2/2	2/2	2/2	2/2
0/7	0/7	7/7	7/7	7/7	6/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7
0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5
0/15	0/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15
0/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6
0/9	0/9	9/9	9/9	9/9	7/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9
0/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4
0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5
0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5


Servers/runtimes
Mobile
4%
66%
95%
59%
52%
97%
97%
2%
24%
99%
99%
S
Echo JS
XS6
JXA
Node 4[5]
Node >=6.5 <7[5]
Node >=8.7 <9[5]
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
15
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
22
12/22
21/22
19/22
0/22
22/22
22/22
0/22
0/22
22/22
22/22
24
14/24
24/24
21/24
0/24
24/24
24/24
0/24
0/24
24/24
24/24
24
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
16
8/16
16/16
10/16
9/16
16/16
16/16
1/16
2/16
16/16
16/16

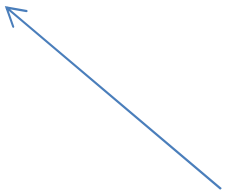
Introduction

- 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 transpilateurs :
  - 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/traceur-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;
    }
   };
```

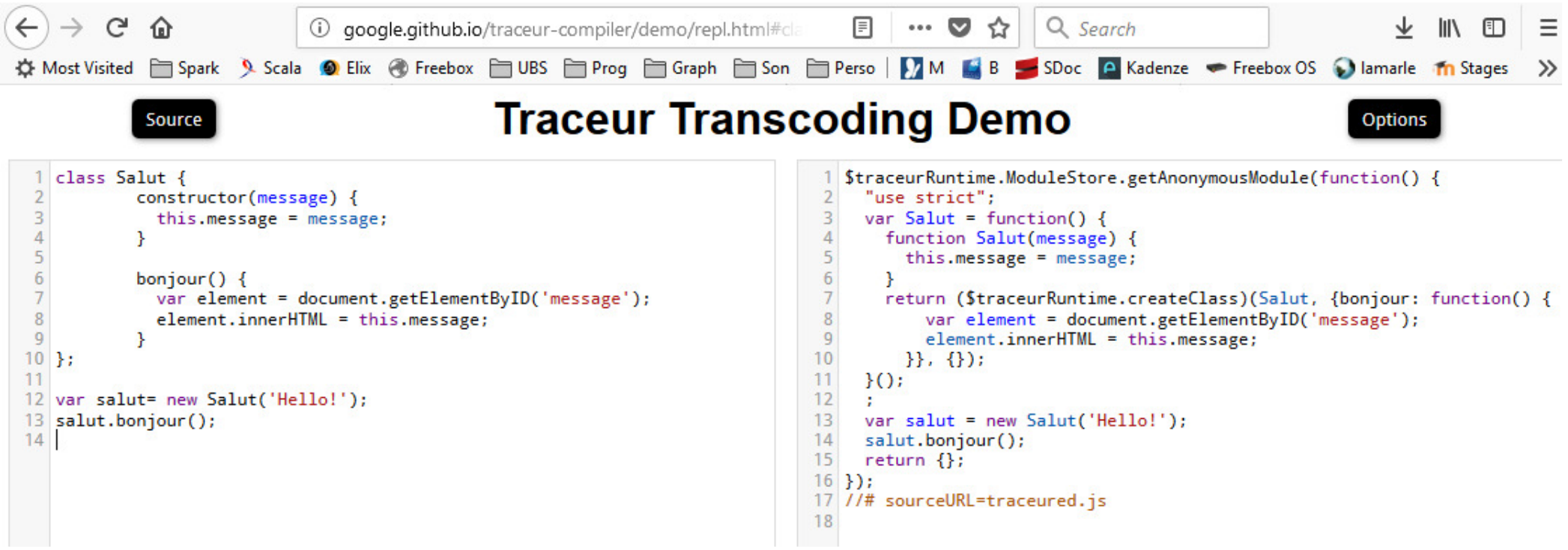
Exemple : Traceur

- Code à insérer :

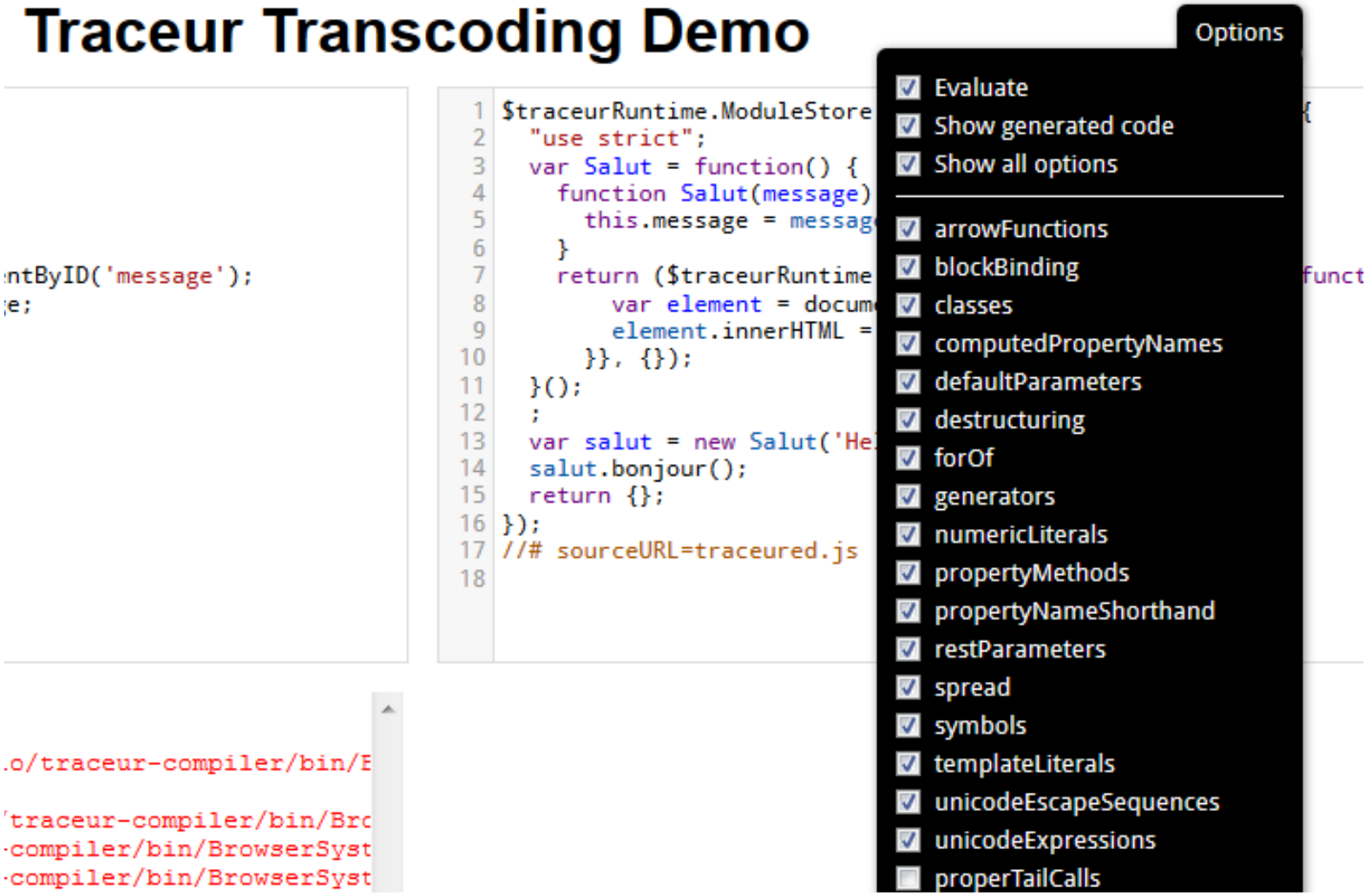
```
    var salut= new Salut('Hello!');
    salut.bonjour();
  </script>
  ...
 </body>
</html>
```

- 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



Traceur Transcoding Demo



Idem pour babeljs

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

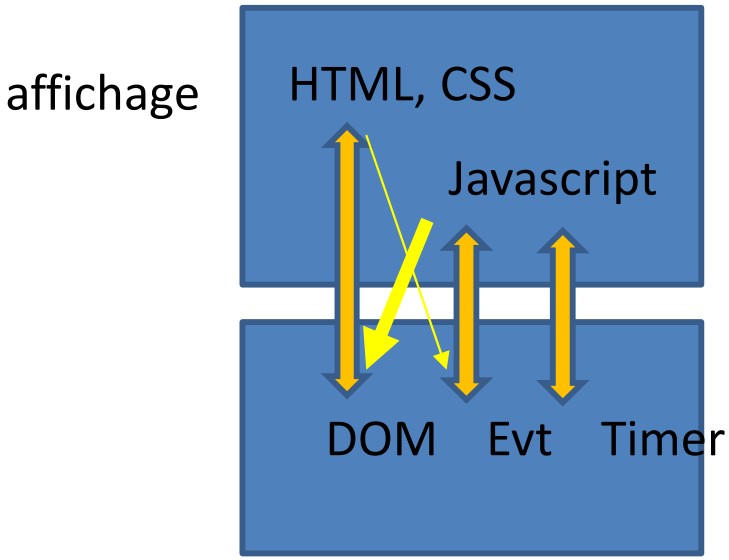
- Deux environnements d'exécution pour Javascript
  - Le navigateur
    - L'interpréteur est intégré au navigateur (et donc figé)
    - Pas d'entrée/sortie fichier
    - Entrées sorties socket limitées
    - Performances dépendantes du navigateur
    - L'interface graphique dépend du navigateur
    -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

Introduction

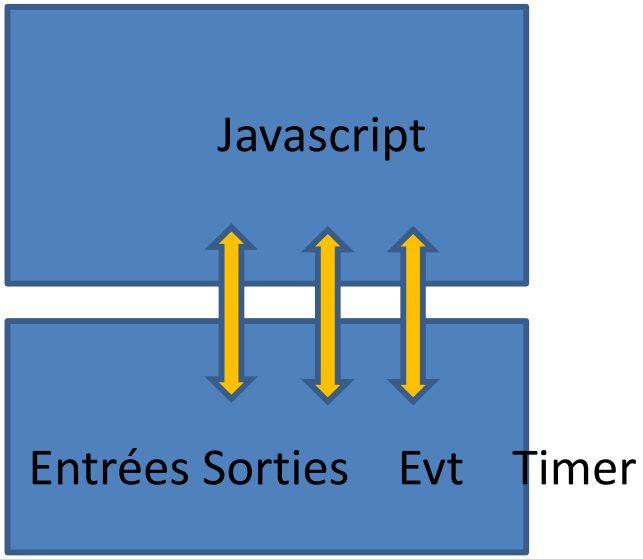
- Navigateur Web et Node.js

Application autonome (pas dans un navigateur)

Navigateur



Node.js

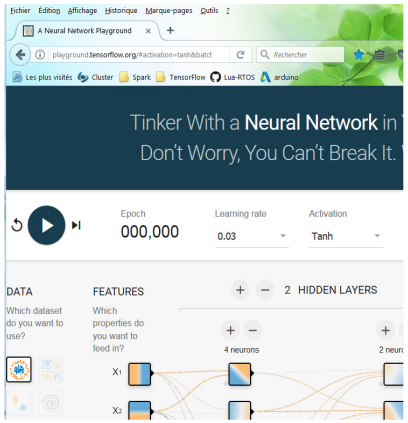


API Navigateur
Le navigateur donne ou pas accès à certaines fonctions, objets ou librairies

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'éxecution

Navigateur

Click souris
Changement d'un texte dans un formulaire
Réception d'information par le Web etc..

Node.js

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