



# JavaScript Avancé

---

Romain Bohdanowicz  
Twitter : @bioub



# Sommaire

---

- Introduction
- IDEs
- Frameworks HTML/CSS/JS
- Préprocesseurs CSS
- Modules JavaScript
- Supersets JavaScript
- Gestion de dépendances
- Git
- Grunt
- Tests Automatisés
- Scaffolding

# Introduction

---

# Présentation

---

- ▶ **Romain Bohdanowicz**  
Ingénieur EFREI 2008, spécialité en Ingénierie Logicielle
- ▶ **Expérience**  
Formateur/Développeur Freelance depuis 2006  
Plus de 5000 heures de formation
- ▶ **Langages**  
Expert : HTML / CSS / JavaScript / PHP / Java  
Notions : C / C++ / Objective-C / C# / Python / Bash / Batch
- ▶ **Certifications**  
PHP 5 / PHP 5.3 / PHP 5.5 / Zend Framework 1
- ▶ **Et vous ?**  
Langages ? Expérience ? Utilité de cette formation ?

# Rappels JavaScript

---

# JavaScript - Introduction

---

- Langage créé en 1995 par Netscape
- Objectif : permettre le développement de scripts légers qui s'exécutent une fois le chargement de la page terminé
- Exemples de l'époque :
  - Valider un formulaire
  - Permettre du rollover
- Netscape ayant un partenariat avec Sun, nomme le langage JavaScript pour qu'il soit vu comme le petit frère de Java (dont il est inspiré syntaxiquement)
- Fin 1995 Microsoft introduit JScript dans Internet Explorer
- Une norme est créée en 1997 : ECMAScript

# JavaScript - ECMAScript

---

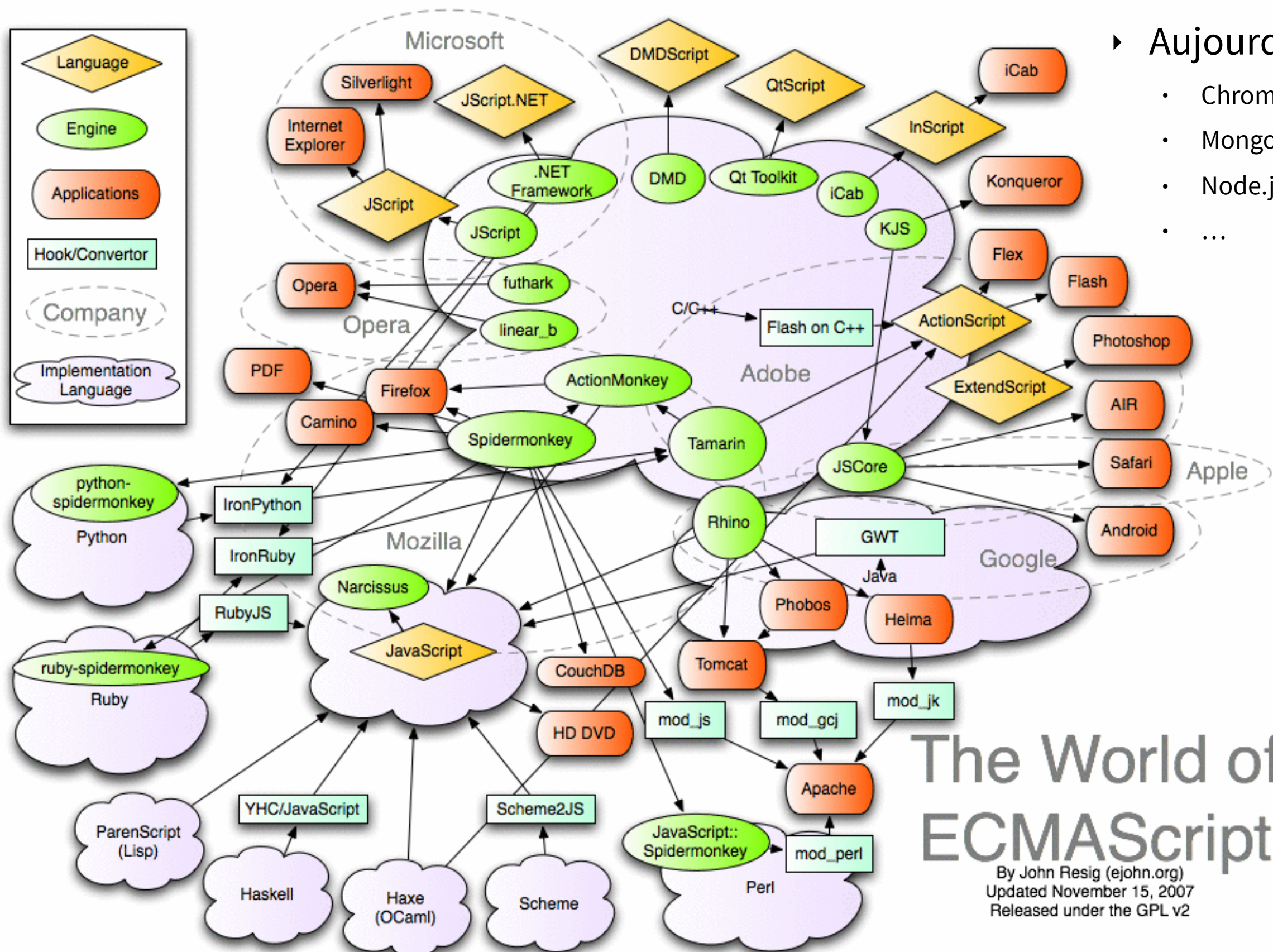
- JavaScript est une implémentation de la norme ECMAScript 262
- La norme la plus récente est ECMAScript 2015, souvent appelée ECMAScript 6 (juin 2015)  
<http://www.ecma-international.org/ecma-262/6.0/>
- Les navigateurs actuels implémentent principalement ECMAScript 5.1, Node.js depuis sa version 4 et la réunion avec io.js implémente en partie la nouvelle norme 6.
- Pour connaître la compatibilité des moteurs JS :  
<http://kangax.github.io/compat-table/es6/>
- Pour découvrir les nouveautés d'ECMAScript 2015  
<http://es6-features.org/>
- Certaines sociétés développent déjà dans la nouvelle norme et utilisent des transpileurs (Babel, Traceur) pour convertir vers du code ES5



# JavaScript - A world of JavaScript

► Aujourd'hui

- Chrome
- MongoDB
- Node.js
- ...





# JavaScript - Syntaxe

---

- La syntaxe s'inspire de Java (lui même inspiré de C)
- JavaScript est sensible à la casse, attention aux majuscules/minuscules !
- Les instructions se termine au choix par un point-virgule ou un retour à la ligne (même si les conventions incitent à la l'utilisation du point-virgule)
- 3 types de commentaires
  - `//` le commentaire s'arrête à la fin de la ligne
  - `/*` commentaire ouvrant/fermant `*/`
  - `/**` Documentation `*/`

# JavaScript - Identifiants

---

- ▶ Les identifiants (noms de variables, de fonctions) doivent respecter les règles suivantes :
  - Contenir uniquement lettres Unicode, Chiffres, \$ et \_
  - Ne commencent pas par un chiffre
- ▶ Bonnes pratiques :
  - ne pas utiliser d'accents (passage d'un éditeur à un autre)
  - séparer les mots dans l'identifiant par des majuscules (camelCase), ou des \_ (snake\_case)
  - les identifiants qui commencent par des \$ ou \_ sont utilisés par certaines conventions
- ▶ Exemples :
  - Valides  
i, maVariable, \$div, v1, prénom
  - Invalides  
1var, ma-variable

# JavaScript - Identifiants

---

## ▶ Attentions aux mots clés réservés (ES6) :

- |            |            |              |          |
|------------|------------|--------------|----------|
| ▶ break    | ▶ do       | ▶ in         | ▶ typeof |
| ▶ case     | ▶ else     | ▶ instanceof | ▶ var    |
| ▶ catch    | ▶ export   | ▶ new        | ▶ void   |
| ▶ class    | ▶ extends  | ▶ return     | ▶ while  |
| ▶ const    | ▶ finally  | ▶ super      | ▶ with   |
| ▶ continue | ▶ for      | ▶ switch     | ▶ yield  |
| ▶ debugger | ▶ function | ▶ this       |          |
| ▶ default  | ▶ if       | ▶ throw      |          |
| ▶ delete   | ▶ import   | ▶ try        |          |

## ▶ Ou réservés pour une utilisation futures (ES7...)

- |        |         |
|--------|---------|
| ▶ enum | ▶ await |
|--------|---------|

# JavaScript - Types

---

## ▶ Voici les types primitifs en JS

- number
- boolean
- string

## ▶ Les types complexes

- object
- array

## ▶ Les types spéciaux

- undefined
- null

# JavaScript - Variables

---

- ▶ Contrairement à certains langages, on ne déclare pas le type au moment de la création

```
var firstName = 'Romain';  
var lastName = 'Bohdanowicz';
```

- ▶ Historiquement on pouvait déclarer une variable sans le mot-clé var, qui devenait alors globale (mauvaise pratique que l'on peut limiter depuis ECMAScript 5)

# JavaScript - Opérateurs

---

## ► Affectation

Nom	Opérateur composé	Signification
Affectation	$x = y$	$x = y$
Affectation après addition	$x += y$	$x = x + y$
Affectation après soustraction	$x -= y$	$x = x - y$
Affectation après multiplication	$x *= y$	$x = x * y$
Affectation après division	$x /= y$	$x = x / y$
Affectation du reste	$x \% = y$	$x = x \% y$
Affectation après exponentiation	$x ** = y$	$x = x ** y$



# JavaScript - Opérateurs

## ► Comparaison

Opérateur	Description	Exemples qui renvoient true
Égalité (==)	Renvoie true si les opérandes sont égaux après conversion en valeurs de mêmes types.	<pre>3 == var1 "3" == var1 3 == '3'</pre>
Inégalité (!=)	Renvoie true si les opérandes sont différents.	<pre>var1 != 4 var2 != "3"</pre>
Égalité stricte (===)	Renvoie true si les opérandes sont égaux et de même type. Voir <a href="#">Object.is()</a> et <a href="#">égalité de type en JavaScript</a> .	<pre>3 === var1</pre>
Inégalité stricte (!==)	Renvoie true si les opérandes ne sont pas égaux ou s'ils ne sont pas de même type.	<pre>var1 !== "3" 3 !== '3'</pre>
Supériorité stricte (>)	Renvoie true si l'opérande gauche est supérieur (strictement) à l'opérande droit.	<pre>var2 &gt; var1 "12" &gt; 2</pre>
Supériorité ou égalité (>=)	Renvoie true si l'opérande gauche est supérieur ou égal à l'opérande droit.	<pre>var2 &gt;= var1 var1 &gt;= 3</pre>
Infériorité stricte (<)	Renvoie true si l'opérande gauche est inférieur (strictement) à l'opérande droit.	<pre>var1 &lt; var2 "2" &lt; "12"</pre>
Infériorité ou égalité (<=)	Renvoie true si l'opérande gauche est inférieur ou égal à l'opérande droit.	<pre>var1 &lt;= var2 var2 &lt;= 5</pre>

# JavaScript - Opérateurs

## ► Arithmétiques

En plus des opérations arithmétiques standards (+, -, \*, /), JavaScript fournit également d'autres opérateurs arithmétiques, listés dans le tableau qui suit :

Opérateur	Description	Exemple
Reste (%)	Opérateur binaire. Renvoie le reste entier de la division entre les deux opérandes.	12 % 5 renvoie 2.
Incrément (++)	Opérateur unaire. Ajoute un à son opérande. S'il est utilisé en préfixe (++x), il renvoie la valeur de l'opérande après avoir ajouté un, s'il est utilisé comme opérateur de suffixe (x++), il renvoie la valeur de	Si x vaut 3, ++x incrémente x à 4 et renvoie 4, x++ renvoie 3 et seulement
Décrément (--)	Opérateur unaire. Il soustrait un à son opérande. Il fonctionne de manière analogue à l'opérateur d'incrément.	Si x vaut 3, --x décrémente x à 2 puis renvoie 2, x-- renvoie 3
Négation unaire (-)	Opérateur unaire. Renvoie la valeur opposée de l'opérande.	Si x vaut 3, alors -x renvoie -3.
Plus unaire (+)	Opérateur unaire. Si l'opérande n'est pas un nombre, il tente de le convertir en une valeur numérique.	+"3" renvoie 3. +true renvoie 1.
Opérateur d'exponentiation (**) (puissance)	Calcule un nombre (base) élevé à une puissance donnée (soit basepuissance)	2 ** 3 renvoie 8 10 ** -1 renvoie -1

# JavaScript - Opérateurs

---

## ► Logiques

Opérateur	Usage	Description
ET logique (&&)	expr1 && expr2	Renvoie expr1 s'il peut être converti à false, sinon renvoie expr2. Dans le cas où on utilise des opérandes booléens, && renvoie true si les deux opérandes valent true, false sinon.
OU logique (  )	expr1    expr2	Renvoie expr1 s'il peut être converti à true, sinon renvoie expr2. Dans le cas où on utilise des opérandes booléens,    renvoie true si l'un des opérandes vaut true, si les deux valent false, il renvoie false.
NON logique (!)	!expr	Renvoie false si son unique opérande peut être converti en true, sinon il renvoie true.

# JavaScript - Opérateurs

---

## ► Concaténation

```
console.log("ma " + "chaîne"); // affichera "ma chaîne" dans la console
```

## ► Ternaire

```
var statut = (âge >= 18) ? "adulte" : "mineur";
```

## ► Voir aussi

Opérateurs binaires, in, instanceof, delete, typeof...

# JavaScript - Opérateurs

---

## ► Priorités

Type d'opérateur	Opérateurs individuels
membre	<code>.</code> <code>[]</code>
appel/création d'instance	<code>()</code> <code>new</code>
négation/incrémentation	<code>!</code> <code>~</code> <code>-</code> <code>++</code> <code>--</code> <code>typeof</code> <code>void</code> <code>delete</code>
multiplication/division	<code>*</code> <code>/</code> <code>%</code>
addition/soustraction	<code>+</code> <code>-</code>
décalage binaire	<code>&lt;&lt;</code> <code>&gt;&gt;</code> <code>&gt;&gt;&gt;</code>
relationnel	<code>&lt;</code> <code>&lt;=</code> <code>&gt;</code> <code>&gt;=</code> <code>in</code> <code>instanceof</code>
égalité	<code>==</code> <code>!=</code> <code>===</code> <code>!==</code>
ET binaire	<code>&amp;</code>
OU exclusif binaire	<code>^</code>
OU binaire	<code> </code>
ET logique	<code>&amp;&amp;</code>
OU logique	<code>  </code>
conditionnel	<code>?:</code>
assignation	<code>=</code> <code>+=</code> <code>-=</code> <code>*=</code> <code>/=</code> <code>%=</code> <code>&lt;&lt;=</code> <code>&gt;&gt;=</code> <code>&gt;&gt;&gt;=</code> <code>&amp;=</code> <code>^=</code> <code> =</code>
virgule	<code>,</code>

# JavaScript - Conversions

---

## ► Conversions implicites

```
console.log(3 * '3'); // 9  
console.log(3 + '3'); // '33'  
console.log(!'texte'); // false
```

## ► Conversions explicites

```
console.log(parseInt('33.33')); // 33  
console.log(parseFloat('33.33')); // 33.33  
console.log(Number('33.33')); // 33.33  
console.log(Boolean('texte')); // true  
console.log(String(33.33)); // '33.33'
```



# JavaScript - Objets Globaux

---

- ▶ **Standard Built-in Objects**

Les objets prédéfinies par le langage, voir la doc de Mozilla pour une liste exhaustive

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects)

- ▶ **Ex : String, Array, Date, Math, RegExp, JSON...**

# JavaScript - Structures

---

## ▸ if ... else

```
if (typeof console === 'object') {  
    console.log('console est un objet');  
}  
else {  
    // oups  
}
```

## ▸ switch

```
switch (alea) {  
    case 0:  
        console.log('zéro');  
        break;  
    case 1:  
    case 2:  
    case 3:  
        console.log('un, deux ou trois');  
        break;  
    default:  
        console.log('entre quatre et neuf');  
}
```

# JavaScript - Structures

---

## ▶ while

```
var alea = Math.floor(Math.random() * 10);

while (alea > 0) {
    console.log(alea);
    alea = parseInt(alea / 2);
}
```

## ▶ do ... while

```
do {
    var alea = Math.floor(Math.random() * 10);
}
while (alea % 2 === 1);

console.log(alea);
```

## ▶ for

```
for (var i=0; i<10; i++) {
    aleas.push(Math.floor(Math.random() * 10));
}

console.log(aleas.join(', ')); // 6, 6, 7, 0, 5, 1, 2, 8, 9, 7
```

# JavaScript - Fonctions

---

- ▶ JavaScript est très consommateur de fonctions
  - réutilisation
  - factorisation
  - récursivité
  - fonction de rappel / écouteur
  - closure
  - module
- ▶ Syntaxe
  - mot clé function
  - nom de fonction (optionnel)
  - arguments (optionnels)
  - déclaration
  - paramètre de retour (optionnel)

# JavaScript - Fonctions

---

## ► Function declaration

```
function addition(nb1, nb2) {  
    return Number(nb1) + Number(nb2);  
}  
  
console.log(addition(2, 3)); // 5
```

## ► Anonymous function expression

```
var addition = function (nb1, nb2) {  
    return Number(nb1) + Number(nb2);  
}  
  
console.log(addition(2, 3)); // 5
```

## ► Named function expression

```
var addition = function addition(nb1, nb2) {  
    return Number(nb1) + Number(nb2);  
}  
  
console.log(addition(2, 3)); // 5
```

# JavaScript - Fonctions

---

## ► Variadic function

```
function addition() {  
    var sum = 0;  
  
    for (var i=0; i<arguments.length; i++) {  
        sum += arguments[i];  
    }  
  
    return sum;  
}  
  
console.log(addition(1, 2, 3)); // 6
```



# JavaScript - Fonctions

---

## ► Fonctions imbriquées

```
function addSquares(a, b) {  
    function square(x) {  
        return x * x;  
    }  
    return square(a) + square(b);  
}  
  
console.log(addSquares(2, 3)); // 13
```

## ► Portées

```
function addSquares(a) {  
    function square() {  
        return a * a;  
    }  
    return square() + square();  
}  
  
console.log(addSquares(2)); // 8
```

# JavaScript - Fonctions

---

## ► Closure

```
function powClosure(exp) {  
    return function(base) {  
        var result = 1;  
        for(var i=0; i<exp; i++) {  
            result *= base;  
        }  
        return result;  
    }  
}  
  
var square = powClosure(2);  
var cube = powClosure(3);  
  
console.log(square(4)); // 16  
console.log(cube(4)); // 64
```

# JavaScript - Fonctions

---

## ► Sans Closure

```
// affiche 4 4 4 dans 1 seconde  
for(var i = 1; i <= 3; i++) {  
    setTimeout(function() {  
        console.log(i);  
    }, 1000);  
}
```

## ► Avec Closure

```
// affiche 1 2 3 dans 1 seconde  
for(var i = 1; i <= 3; i++) {  
    setTimeout(function(rememberI) {  
        return function() {  
            console.log(rememberI);  
        }  
    }(i), 1000);  
}
```

# JavaScript - Fonctions

---

## ► Callback

```
function onSeCallDansTroisSecondes(callback) {  
    setTimeout(callback, 3000);  
}  
  
function ditBonjour() {  
    console.log('Bonjour');  
}  
  
onSeCallDansTroisSecondes(ditBonjour); // affichera Bonjour dans 3 sec
```

- Les callbacks (fonctions de rappel) permettent la mise en place de programmation asynchrone (via une boucle d'événements)

# JavaScript - Fonctions

---

## ► Valeurs par défaut

```
function valeursParDefault(largeur, hauteur, couleur, valeur) {  
    // Version 1 avec le type  
    if (typeof valeur === 'undefined') {  
        valeur = 'Valeur par défaut';  
    }  
  
    // Version 2 avec la conversion (attention pas de valeur  
    // équivalente à false)  
    if (!valeur) valeur = 'Valeur par défaut';  
  
    // Version 3 avec conversion + OR  
    valeur = valeur || 'Valeur par défaut';  
}  
  
valeursParDefault(100, 30, 'blue');
```

# JavaScript - Fonctions

---

## ► Object function

```
var contact = {  
  prenom: 'Romain',  
  nom: 'Bohdanowicz'  
};  
  
function saluer(prenom) {  
  return 'Bonjour ' + prenom + ' je suis ' + this.prenom;  
}  
  
console.log(saluer('Eric')); // Bonjour Eric je suis undefined  
console.log(saluer.call(contact, 'Eric')); // Bonjour Eric je suis Romain  
console.log(saluer.apply(contact, ['Eric'])); // Bonjour Eric je suis Romain  
console.log(saluer.bind(contact)('Eric')); // Bonjour Eric je suis Romain
```



# JavaScript - Fonctions

---

## ► Module

### Immediately Invoked Function Expression (IIFE)

```
(function($, global) {  
    'use strict';  
  
    function MonBouton(options) {  
        this.options = options || {};  
        this.value = options.value || 'Valider';  
    }  
  
    MonBouton.prototype.creer = function(container) {  
        $(container).append('<button>'+this.value+'</button>')  
    };  
  
    global.MonBouton = MonBouton;  
})(jQuery, window));
```

# JavaScript - Objet

---

- ▶ Création d'un objet avec l'objet global Object :

```
var instructor = new Object();  
instructor.firstName = 'Romain';  
instructor.hello = function() {  
    return 'Hello my name is ' + this.firstName;  
};  
  
console.log(instructor.hello()); // Hello my name is Romain
```

- ▶ Création d'un objet avec la syntaxe Object Literal (recommandé) :

```
var instructor = {  
    firstName: 'Romain',  
    hello: function() {  
        return 'Hello my name is ' + this.firstName;  
    }  
};  
  
console.log(instructor.hello()); // Hello my name is Romain
```

# JavaScript - Objet

---

## ► Accès aux objets possible :

- Avec l'opérateur .
- Avec des crochets

```
var instructor = {  
  firstName: 'Romain',  
  hello: function() {  
    return 'Hello my name is ' + this.firstName;  
  }  
};  
  
instructor.firstName = 'Jean';  
console.log(instructor.hello()); // Hello my name is Jean  
  
instructor['firstName'] = 'Eric';  
console.log(instructor['hello']()); // Hello my name is Eric
```

# JavaScript - Typage/famille d'objets

---

- En utilisant une fonction constructeur :

```
function Person(firstName) {  
    this.firstName = firstName;  
  
    this.hello = function () {  
        return 'Hello my name is ' + this.firstName;  
    }  
}  
  
var instructor = new Person('Romain');  
  
console.log(instructor.hello()); // Hello my name is Romain  
console.log(typeof instructor); // object  
console.log(instructor instanceof Object); // true  
console.log(instructor instanceof Person); // true  
  
for (var prop in instructor) {  
    if (instructor.hasOwnProperty(prop)) {  
        console.log(prop); // firstName puis hello  
    }  
}
```

# JavaScript - Typage/famille d'objets

---

- En utilisant une fonction constructeur + son prototype :

```
function Person(firstName) {  
    this.firstName = firstName;  
}  
  
Person.prototype.hello = function () {  
    return 'Hello my name is ' + this.firstName;  
};  
  
var instructor = new Person('Romain');  
  
console.log(instructor.hello()); // Hello my name is Romain  
console.log(typeof instructor); // object  
console.log(instructor instanceof Object); // true  
console.log(instructor instanceof Person); // true  
  
for (var prop in instructor) {  
    if (instructor.hasOwnProperty(prop)) {  
        console.log(prop); // firstName  
    }  
}
```

# JavaScript - Héritage

- En utilisant une fonction constructeur + son prototype :

```
function Person(firstName) {
    this.firstName = firstName;
}

Person.prototype.hello = function () {
    return 'Hello my name is ' + this.firstName;
};

function Instructor(firstName, speciality) {
    Person.apply(this, arguments); // héritage des propriétés de l'objet (recopie dynamique)
    this.speciality = speciality;
}

Instructor.prototype = new Person; // héritage du type

// Redéfinition de méthode
Instructor.prototype.hello = function() {
    // Appel de la méthode parent
    return Person.prototype.hello.call(this) + ', my speciality is ' + this.speciality;
};

var instructor = new Instructor('Romain', 'JavaScript');

console.log(instructor.hello()); // Hello my name is Romain
console.log(typeof instructor); // object
console.log(instructor instanceof Object); // true
console.log(instructor instanceof Person); // true
console.log(instructor instanceof Instructor); // true

for (var prop in instructor) {
    if (instructor.hasOwnProperty(prop)) {
        console.log(prop); // firstName, speciality
    }
}
```

# JavaScript - Prototype

---

## ▸ Définition Wikipedia :

La programmation orientée prototype est une forme de programmation orientée objet sans classe, basée sur la notion de prototype. Un prototype est un objet à partir duquel on crée de nouveaux objets.

## ▸ Comparaison des modèles à classes et à prototypes

- Objets à classes :
  - Une classe définie par son code source est statique ;
  - Elle représente une définition abstraite de l'objet ;
  - Tout objet est instance d'une classe ;
  - L'héritage se situe au niveau des classes.
- Objets à prototypes :
  - Un prototype défini par son code source est mutable ;
  - Il est lui-même un objet au même titre que les autres ;
  - Il a donc une existence physique en mémoire ;
  - Il peut être modifié, appelé ;
  - Il est obligatoirement nommé ;
  - Un prototype peut être vu comme un exemplaire modèle d'une famille d'objet ;
  - Un objet hérite des propriétés (valeurs et méthodes) de son prototype ;

# JavaScript - Prototype

---

- En ECMAScript/JavaScript, l'écriture `foo.bar` s'interprète de la façon suivante :
  1. Le nom `foo` est recherché dans la liste des identifiants déclarés dans le contexte d'appel de fonction courant (déclarés par `var`, ou comme paramètre de la fonction) ;
  2. S'il n'est pas trouvé :
    - Continuer la recherche (retour à l'étape 1) dans le contexte de niveau supérieur (s'il existe),
    - Sinon, le contexte global est atteint, et la recherche se termine par une erreur de référence.
  3. Si la valeur associée à `foo` n'est pas un objet, il n'a pas de propriétés ; la recherche se termine par une erreur de référence.
  4. La propriété `bar` est d'abord recherchée dans l'objet lui-même ;
  5. Si la propriété ne s'y trouve pas :
    - Continuer la recherche (retour à l'étape 4) dans le prototype de cet objet (s'il existe) ;
    - Si l'objet n'a pas de prototype associé, la valeur indéfinie (`undefined`) est retournée ;
  6. Sinon, la propriété a été trouvée et sa référence est retournée.



# JavaScript - JSON

---

- JSON, JavaScript Object Notation est la sérialisation d'un objet JavaScript
- Seuls les types string, number, boolean et array sont sérialisable, les fonctions et prototype sont perdus
- On se sert de ce format pour échanger des données entre 2 programmes ou pour créer de la config
- Le format résultant est proche de Object Literal, les clés sont obligatoirement entre guillemets "", un code JSON est une syntaxe Object Literal valide

```
{  
  "name": "My Address Book",  
  "contacts": [  
    {  
      "firstName": "Bill",  
      "lastName": "Gates"  
    },  
    {  
      "firstName": "Steve",  
      "lastName": "Jobs"  
    }  
  ]  
}
```

# JavaScript - JSON

---

- ▶ JavaScript depuis EcmaScript 5 fourni l'objet global JSON qui contient 2 méthodes, parse (désérialiser) et stringify (sérialiser)

```
var contact = {  
  prenom: 'Romain',  
  nom: 'Bohdanowicz'  
};  
  
var json = JSON.stringify(contact);  
console.log(json); // {"prenom":"Romain","nom":"Bohdanowicz"}  
  
var object = JSON.parse(json);  
console.log(object.prenom); // Romain
```

# JavaScript - Mode Strict

---

- ▶ Limite des mauvaises pratiques historiques de JavaScript

```
(function($, global) {  
    'use strict';  
  
    function MonBouton(options) {  
        this.options = options || {};  
        this.value = options.value || 'Valider';  
    }  
  
    MonBouton.prototype.creer = function(container) {  
        $(container).append('<button>'+this.value+'</button>')  
    };  
  
    global.MonBouton = MonBouton;  
})(jQuery, window);
```

# JavaScript - Programmation Événementielle

---

- Les moteurs JS sont par défaut mono-thread et mono-processus, ils ne peuvent donc exécuter qu'une seule tâche à la fois.
- Une boucle d'événements permet de passer d'un callback à l'autre de manière très performante, ex : traiter le clic d'un bouton entre 2 étapes d'une animation
- JavaScript est non-bloquant, il stocke les événements à traiter sous la forme d'une file de message et appellera les callbacks lorsqu'il sera disponible
- Bonne pratique : les callbacks doivent avoir un temps d'exécution court pour ne pas ralentir l'appel des callbacks suivants

```
setTimeout(function() {  
    console.log('1 fois dans 3 secondes');  
}, 3000);  
  
var intervalId = setInterval(function() {  
    console.log('toutes les 2 secondes');  
}, 2000);  
  
setTimeout(function() {  
    console.log('Bye bye');  
    clearInterval(intervalId);  
}, 15000);
```

# JavaScript - DOM

---

- ▶ Document Object Model

API créé par Netscape en même temps que le JavaScript qui permet la manipulation du HTML en mémoire sous la forme d'un arbre

- ▶ W3C

Une norme existe depuis 1998, aujourd'hui dans sa 4e édition

DOM Level 4 : <http://www.w3.org/TR/dom/>

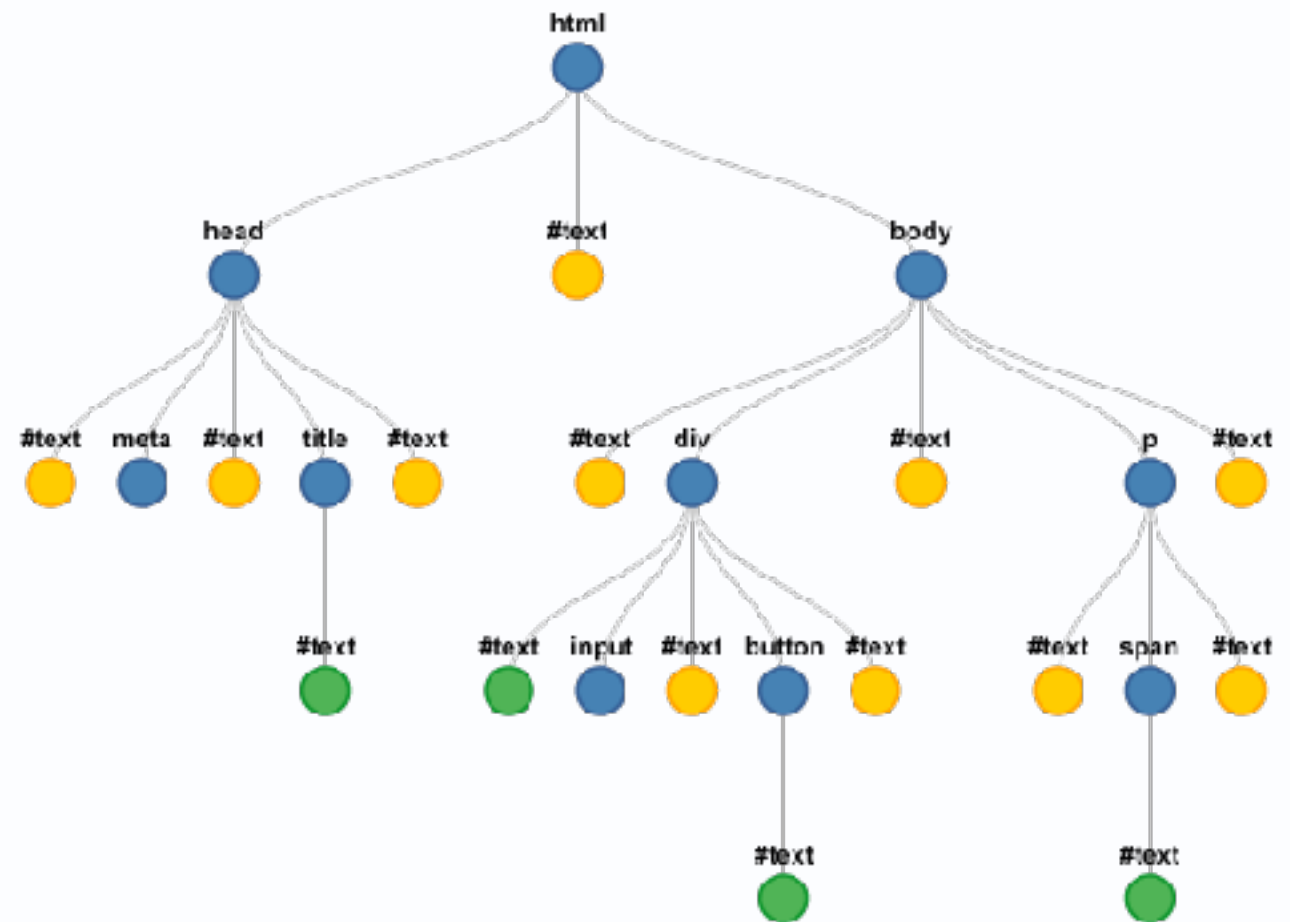
DOM Level 3 Events : <http://www.w3.org/TR/DOM-Level-3-Events/>

HTML5 : <http://www.w3.org/TR/html5/>

# JavaScript - DOM

- Représentation sous la forme d'un arbre

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Helloworld</title>
</head>
<body>
  <div>
    Prénom :
    <input name="prenom" id="prenom">
    <button id="clicme">Hello</button>
  </div>
  <p>
    Bonjour <span id="result"> </span>
  </p>
</body>
</html>
```



# JavaScript - DOM

---

## ▸ DOM 1 (années 1990)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Helloworld</title>
  <script>
    function helloworld() {
      var input = document.getElementsByName('prenom')[0];
      var span = document.getElementsByTagName('span')[0];

      span.firstChild.nodeValue = input.value;
    }
  </script>
</head>
<body>
<div>
  Prénom :
  <input type="text" name="prenom" id="prenom">
</div>
<p>
  Bonjour <span id="result"> </span>
</p>
</body>
</html>
```

# JavaScript - DOM

---

## ▸ DOM 2 / DOM 3 (années 2000)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Helloworld</title>
  <script>
    window.addEventListener('load', function() {
      var input = document.getElementById('prenom');
      var span = document.getElementById('result');

      input.addEventListener('input', function () {
        span.textContent = input.value;
      });
    });
  </script>
</head>
<body>
<div>
  Prénom :
  <input type="text" name="prenom" id="prenom">
</div>
<p>
  Bonjour <span id="result"> </span>
</p>
</body>
</html>
```



# JavaScript - DOM

---

## ► DOM 4 (années 2010)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Helloworld</title>
</head>
<body>
<div>
  Prénom :
  <input type="text" name="prenom" id="prenom">
</div>
<p>
  Bonjour <span id="result"> </span>
</p>
<script>
  (function() {
    'use strict';

    var input = document.querySelector('input[type=text][name=prenom]');
    var span = document.querySelector('body > p span#result');

    input.addEventListener('input', function () {
      span.innerHTML = '<b>' + input.value + '</b>';
    });
  })();
</script>
</body>
</html>
```

# IDEs

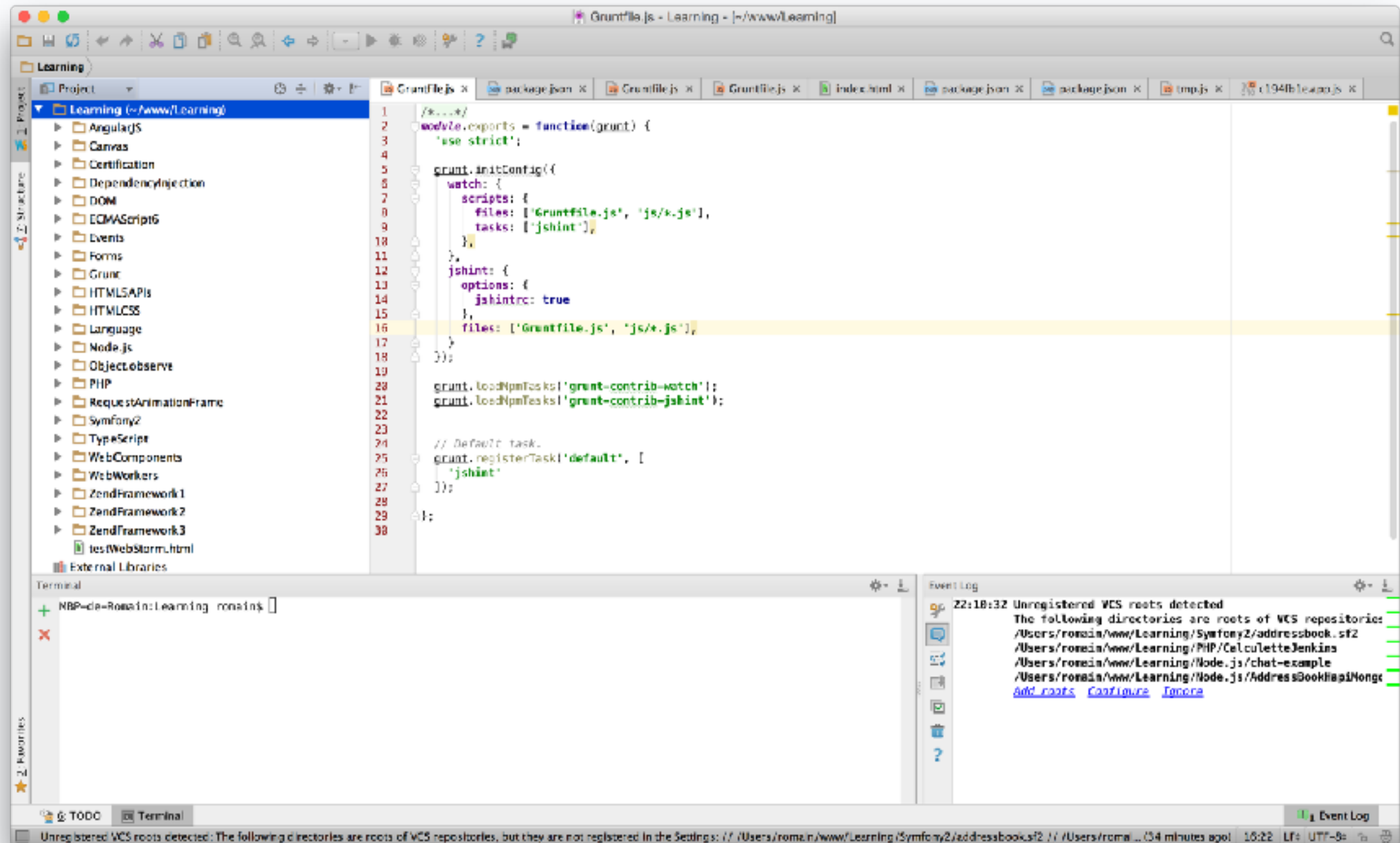
---



- ▶ Version orientée Web de IntelliJ IDEA de l'éditeur JetBrains
- ▶ Licence : Commercial  
Licence entre 49 et 99 euros selon profil.
- ▶ Plugins :  
Annuaire (390 en juillet 2015) : <https://plugins.jetbrains.com/webStorm>  
Langage de création : Java



# IDE - WebStorm

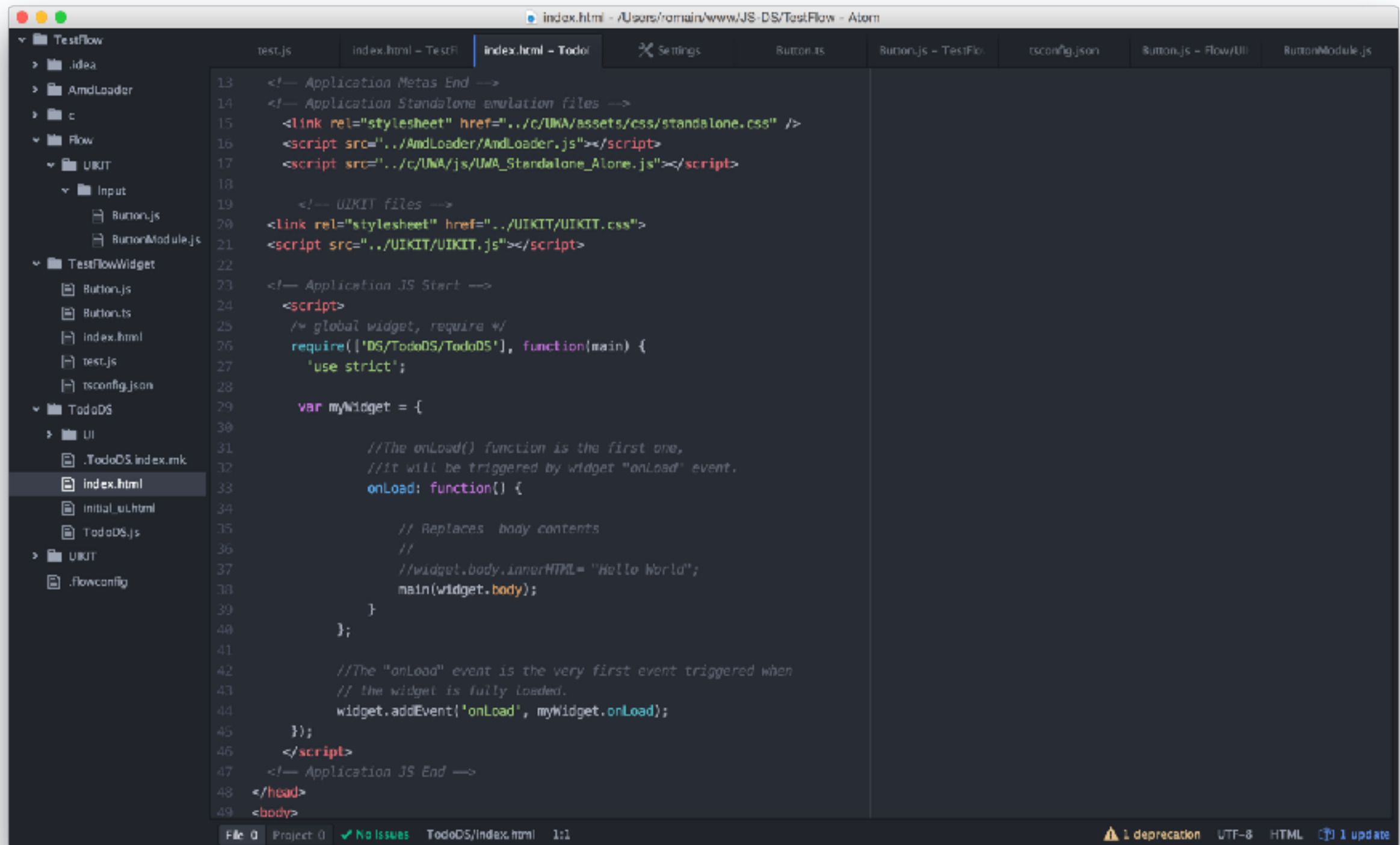




- ▶ IDE créé par Github, tourne sous Electron (Chromium + io.js)
- ▶ Licence : MIT  
La licence open-source la plus permissive
- ▶ Plugins :  
Annuaire (2381 en juillet 2015) : <https://atom.io/packages>  
Langage de création : JavaScript sous io.js  
Exemples : atom-ternjs, linter, JavaScript Snippets, autocomplete+, autoprefixer...)



# IDE - Atom



# IDE - Brackets

---

- ▶ **Créé par Adobe**

Basé sur Chromium (a inspiré Atom)

- ▶ **Licence : MIT**

La licence open-source la plus permissive

- ▶ **Plugins :**

Annuaire intégré (854 en juillet 2015) (voir aussi <https://ingorichter.github.io/BracketsExtensionTweetBot/>)


Intégration d'un Live Preview

Génération de CSS depuis un fichier PSD Photoshop



# IDE - Brackets

---



```
app.js (TodoListREST) - Brackets

1  var express = require('express');
2  var logger = require('morgan');
3  var port = 8080;
4
5  var app = express();
6  app.use(logger('dev'));
7
8  var apiTodos = require('./routes/api-todos');
9
10
11 app.use('/api/v1/todos', apiTodos);
12
13 app.listen(port);
14
15 console.log("App started at http://localhost:" + port);
16
```

Ligne 1, colonne 1 - 16 lignes

INS JavaScript  Espaces : 2



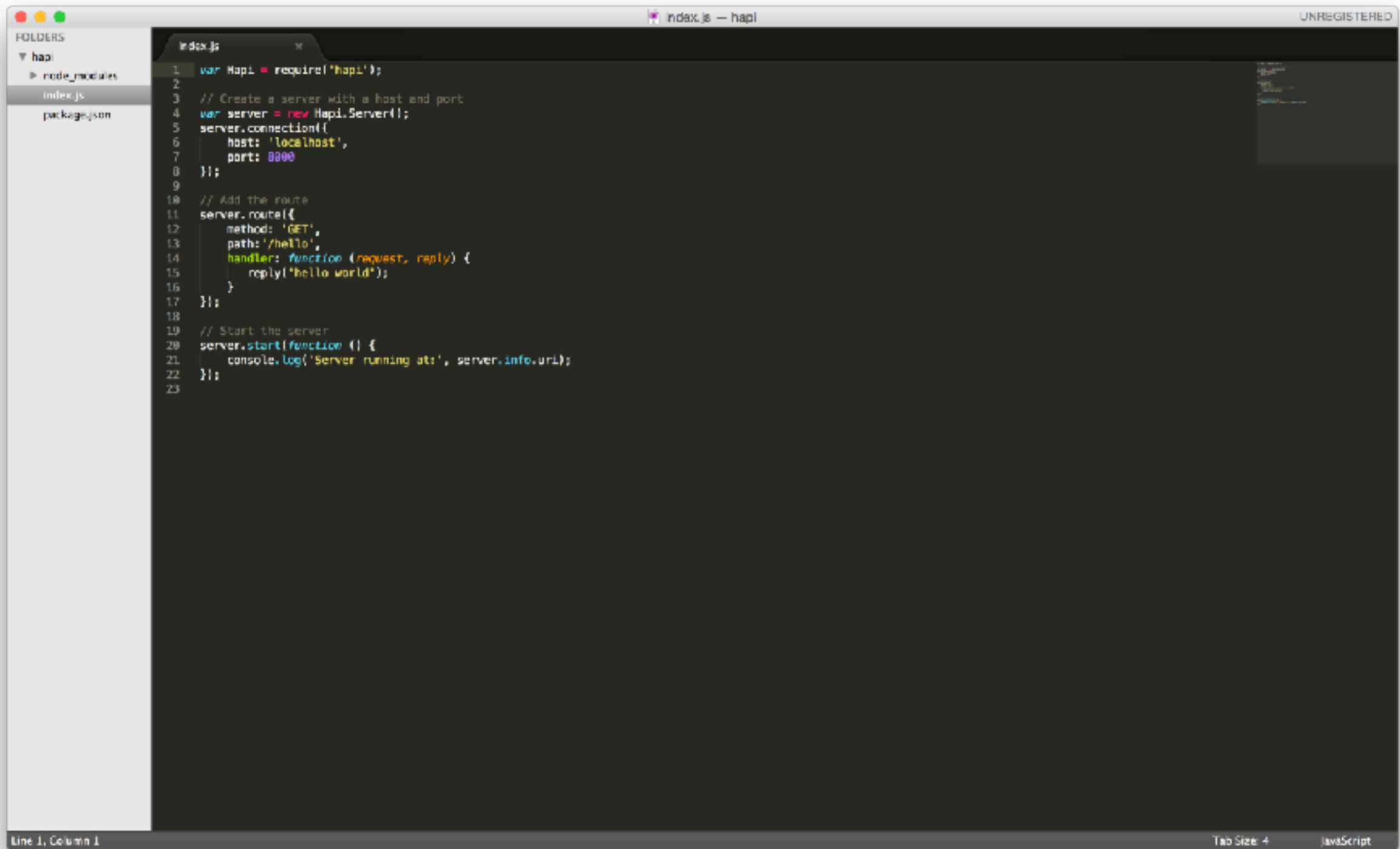
# IDE - Sublime Text

---

- ▶ IDE dont s'est inspiré Atom
- ▶ Licence : Commercial  
Licence entre 70 dollars.
- ▶ Plugins :  
Pas d'annuaire officiel (mais voir <https://packagecontrol.io>, 3102 en juillet 2015)  
Langage de création : Python



# IDE - Sublime Text



```
1 var Hapi = require('hapi');
2
3 // Create a server with a host and port
4 var server = new Hapi.Server();
5 server.connection({
6   host: 'localhost',
7   port: 8888
8 });
9
10 // Add the route
11 server.route({
12   method: 'GET',
13   path: '/hello',
14   handler: function (request, reply) {
15     reply('hello world');
16   }
17 });
18
19 // Start the server
20 server.start(function () {
21   console.log('Server running at:', server.info.uri);
22 });
23
```

Line 1, Column 1

Tab Size: 4 JavaScript

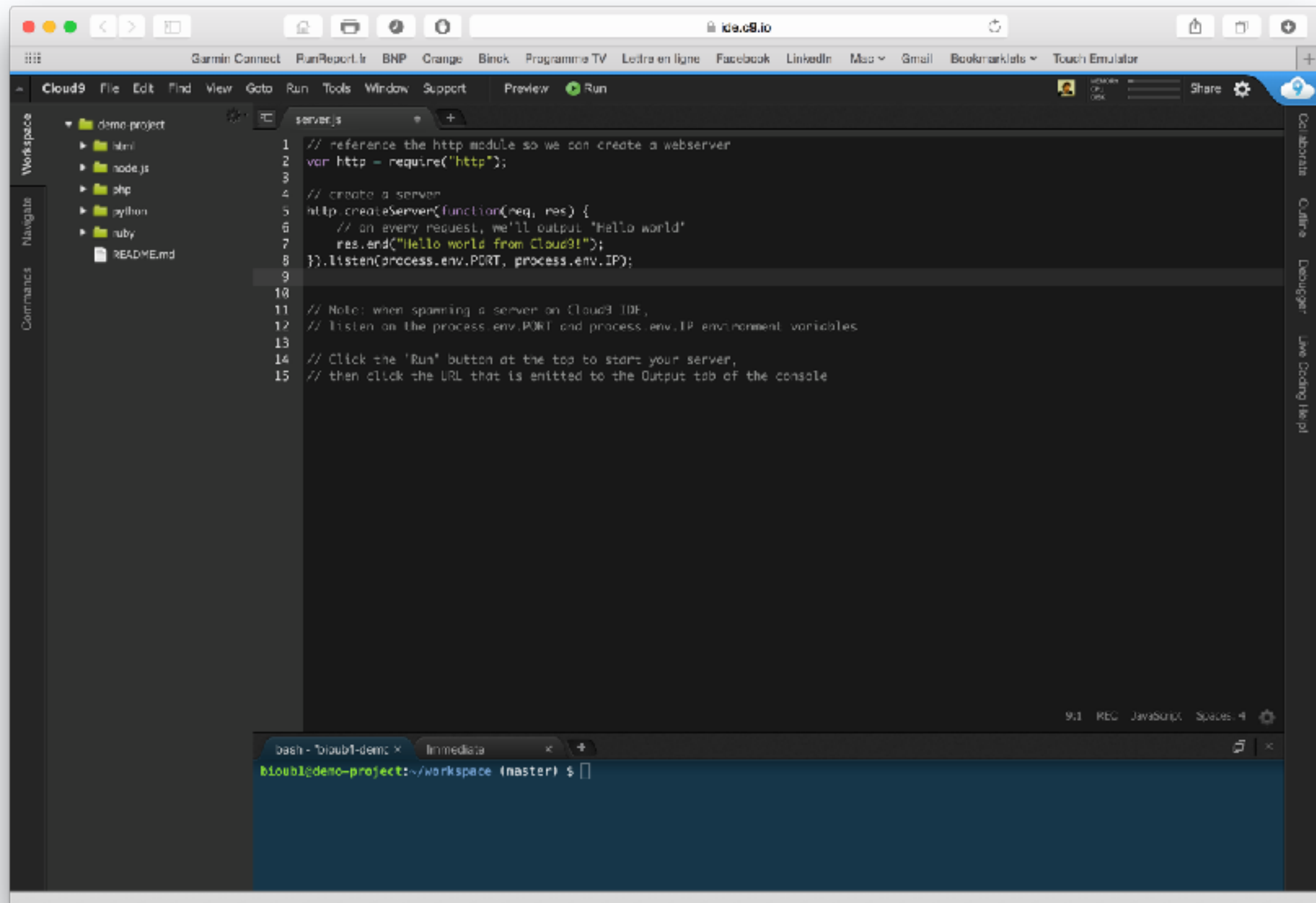
# IDE - Cloud9

---

- ▶ IDE entièrement en ligne
- ▶ Licence :
  - Gratuit pour 512Mo de RAM, 1Go de disque et 1 workspace privé
  - Payant au delà (de 9 à 79\$ par mois)



# IDE - Cloud9



- ▶ **Les trop lourds : Eclipse, Netbeans**

Très consommateurs en ressources et moins complets que les précédents.

Excellents pour du Java, un peu moins pour du Front (même si Netbeans s'en sort plutôt bien).

Des plugins existents pour les étendre (Langage de création : Java)

- ▶ **Les trop légers : NotePad++, Textmate, VIM...**

Offrent moins de fonctionnalités que les précédents mais sont beaucoup plus légers et termes de ressources, certains développeurs ne jurent que par eux.

Des plugins existents pour la plupart.

- ▶ **Les petits derniers : Visual Studio Code, Nuclide**

Basés sur Atom, VSCode parfait pour du code TypeScript ! Nuclide pour les technos Facebook : React, Hack, Flow...

# IDE - Le fichier .editorconfig

---

- ▶ Permet de standardiser les configs des IDEs

<http://editorconfig.org>

- ▶ Supporté par la plupart des IDE

```
# EditorConfig helps developers define and maintain consistent
# coding styles between different editors and IDEs
# editorconfig.org

root = true

[*]

# Change these settings to your own preference
indent_style = space
indent_size = 2

# We recommend you to keep these unchanged
end_of_line = lf
charset = utf-8
trim_trailing_whitespace = true
insert_final_newline = true

[*.md]
trim_trailing_whitespace = false
```

# Frameworks HTML/CSS/JS

---

# Frameworks HTML/CSS/JS - Introduction

---

- ▶ Popularisé lorsque Twitter a proposé sa bibliothèque UI en open source sous le nom de Bootstrap en 2011
- ▶ Unifie et accélère le développement, la majeure partie du CSS est déjà développée
- ▶ Inverse les responsabilités : le HTML fait la mise en forme en s'intégrant à un CSS existant



# Frameworks HTML/CSS/JS - Bootstrap

---



- ▶ Créé par Twitter
- ▶ Open Source depuis 2011
- ▶ Projet le plus populaire sur GitHub  
Contributeurs : 658 - Watches : 5092 - Stars 83109 - Forks 33538 (juillet 2015)
- ▶ Ecrit avec jQuery, Less, QUnit, Grunt...
- ▶ Documentation  
<http://getbootstrap.com>
- ▶ Support : IE8 avec HTML5 shim et Respond.js

# Frameworks HTML/CSS/JS - Bootstrap

---

- Téléchargement :  
<https://github.com/twbs/bootstrap/archive/v3.3.5.zip>
- CDN  
<https://www.bootstrapcdn.com>
- Git  
git clone <https://github.com/twbs/bootstrap.git>
- Bower  
bower install bootstrap
- npm  
npm install bootstrap
- Meteor  
meteor add twbs:bootstrap
- Composer  
composer require twbs/bootstrap

# Frameworks HTML/CSS/JS - Bootstrap

---

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <!-- The above 3 meta tags *must* come first in the head; any other head content must come *after* these tags -->
  <title>Bootstrap 101 Template</title>

  <!-- Bootstrap -->
  <link href="css/bootstrap.min.css" rel="stylesheet">

  <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
  <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
  <!--[if lt IE 9]>
  <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
  <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
  <![endif]-->
</head>
<body>
<h1>Hello, world!</h1>

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<!-- Include all compiled plugins (below), or include individual files as needed -->
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

- Nécessite jQuery + HTML5 shim et Respond.js (IE8)

# Frameworks HTML/CSS/JS - Bootstrap

---

- Mise en forme de balises existantes (bouton, formulaires...)
- Inclus Normalize.css
- Composants mis en forme :
  - Container
  - Grid system
  - Typography
  - Code
  - Tables
  - Forms
  - Buttons
  - Images
  - Helper classes
  - Responsive utilities

# Frameworks HTML/CSS/JS - Bootstrap

---

## ▸ HTML + CSS de composants plus haut niveau

### ▸ Composants :

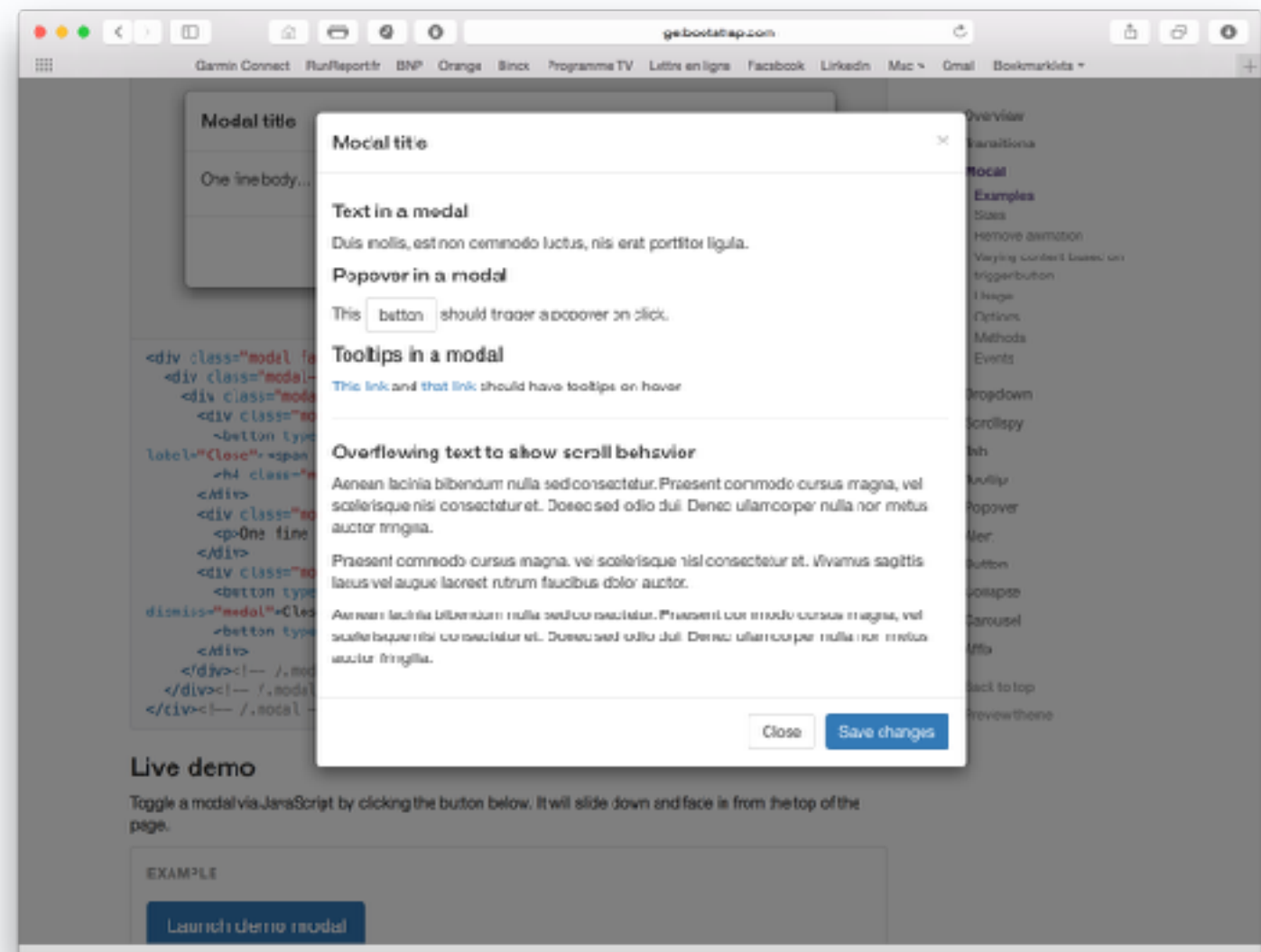
- Glyphicons
- Dropdowns
- Button groups
- Button dropdowns
- Input groups
- Navs
- Navbar
- Breadcrumbs
- Pagination
- Labels
- Badges
- Jumbotron
- Page header
- Thumbnails
- Alerts
- Progress bars
- Media object
- List group
- Panels
- Responsive embed
- Wells

# Frameworks HTML/CSS/JS - Bootstrap

## ► Plugins pour jQuery

## ► Composants :

- Transitions
- Modal
- Dropdown
- Scrollspy
- Tab
- Tooltip
- Popover
- Alert
- Button
- Collapse
- Carousel
- Affix



# Frameworks HTML/CSS/JS - Foundation

---

- ▶ 2e framework HTML/CSS/JS sur GitHub



- ▶ Documentation

<http://foundation.zurb.com>

- ▶ Stats Github :

Contributeurs : 705 - Watches : 1431 - Stars 20611 - Forks 4394 (juillet 2015)

- ▶ Ecrit avec jQuery, SASS, Jasmine, Grunt...

- ▶ Support : IE9+

# Frameworks HTML/CSS/JS - Semantic UI

---



- ▶ 3e framework HTML/CSS/JS sur GitHub
- ▶ Documentation  
<http://semantic-ui.com>
- ▶ Stats Github :  
Contributeurs : 111 - Watches : 994 - Stars 19191 - Forks 2129 (juillet 2015)
- ▶ Ecrit avec jQuery, LESS, Jasmine, Gulp...
- ▶ Support : Last 2 Versions FF, Chrome, IE 10+, Safari Mac





# Préprocesseurs CSS

---

# Préprocesseurs CSS - Introduction

---

- ▶ Les préprocesseurs CSS sont des technologies qui permettent à des langages proches de CSS de transpirer en CSS en y ajoutant des fonctionnalités
- ▶ Comparateurs de préprocesseurs CSS  
<http://csspre.com/compare/>

# Préprocesseurs CSS - Less

---

- Apparu en 2009, inspiré par SASS  
<http://lesscss.org/>



## ▸ Variables

```
@link-color: #428bca; // sea blue

a, .link {
  color: @link-color;
}

.widget {
  color: #fff;
  background: @link-color;
}
```

## ▸ Héritage

```
nav ul {
  &:extend(.inline);
  background: blue;
}
```

# Préprocesseurs CSS - Less

---

## ► Mixins (fonctions)

```
.border-radius(@radius) {  
  -webkit-border-radius: @radius;  
  -moz-border-radius: @radius;  
  border-radius: @radius;  
}  
  
#header {  
  .border-radius(4px);  
}  
  
.button {  
  .border-radius(6px);  
}
```

## ► Imports

```
.foo {  
  background: #900;  
}  
@import "this-is-valid.less";
```

# Préprocesseurs CSS - Less

---

## ► Imbriquer

```
#header {  
  color: black;  
  .navigation {  
    font-size: 12px;  
  }  
  .logo {  
    width: 300px;  
  }  
}
```

# Préprocesseurs CSS - Less

---

## ► Transpiler

- Côté client :

```
<link rel="stylesheet/less" type="text/css" href="styles.less" />  
<script src="less.js" type="text/JavaScript"></script>
```

- En ligne de commande
- Installer

```
npm install -g less
```

- Transpiler

```
lessc styles.less > styles.css
```

# Préprocesseurs CSS - SASS/SCSS



- Apparue en 2007

<http://sass-lang.com>

- A peu de choses près fonctionnalités égales à Less

- 2 syntaxes :

- SASS

```
$primary-color: #333  
  
body  
  color: $primary-color
```

- SCSS (inspirée de Less)

```
$primary-color: #333;  
  
body {  
  color: $primary-color;  
}
```

# Préprocesseurs CSS - Stylus

---

- Apparue en 2010, inspiré par SASS et LESS  
<http://learnboost.github.io/stylus/>
- Syntaxe encore plus concise

```
body
  font 12px Helvetica, Arial, sans-serif

a.button
  -webkit-border-radius 5px
  -moz-border-radius 5px
  border-radius 5px
```






# Préprocesseurs CSS - Myth et cssnext

## ► Less et Sass ont inspiré le W3C

Des normes sur les variables, opérateurs existe désormais nativement mais sont mal supportées

calc() as CSS unit value  - CR

Global 76.09% + 5.16% = 81.25%

unprefixed: 75.55% + 5.16% = 80.71%

Method of allowing calculated values for length units, i.e. width:  
calc(100% - 3em)

Current aligned Usage relative Show all

IE / Edge	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
8							4.1	
<sup>2</sup> 9		31					4.3	
10		42					<sup>1</sup> 4.4	
11	38	43	7.1		7.1		<sup>1</sup> 4.4.4	
Edge	39	44	8	30	8.4	8	40	42
	40	45	9	31	9			
	41	46		32				
	42	47						

# Préprocesseurs CSS - Myth

---

## ► Apparu fin 2013

<http://www.myth.io>

## ► Variables

```
:root {  
  --purple: #847AD1;  
  --large: 10px;  
}  
  
a {  
  color: var(--purple);  
}  
  
pre {  
  padding: var(--large);  
}
```

## ► Opérateurs

```
pre {  
  margin: calc(var(--large) * 2);  
}
```



- ▶ Apparue mi-2014

<http://cssnext.io>

- ▶ Supporte plus de nouveautés CSS que Myth

automatic vendor prefixes, custom properties & ``var()``, reduced ``calc()``, custom media queries, media queries ranges, custom selectors, ``color()``, ``hwb()``, ``gray()``, `#rrggbbaa`, ``rebeccapurple``, font-variant, filter, ``rem`` units, ``:any-link`` pseudo-class, ``:matches`` pseudo-class, ``:not`` pseudo-class, pseudo-elements, Alpha colors, Bonus features, ``@import``, minification, @todo

**cssnext** {||||}

# Modules JavaScript

---

# Modules JavaScript - Introduction

---

## ► JavaScript inventé en 1995 par Netscape

Objectif : créer des interactions côté client, après chargement de la page

Exemple de l'époque :

- Menu en rollover (image ou couleur change au survol)
- Validation de formulaire

## ► JavaScript aujourd'hui

- Permet la création d'application front-end, back-end, en ligne de commande, application de bureau
- Ces applications peuvent contenir plusieurs centaines de milliers de lignes de codes
- Il faut faciliter le travail collaboratif, en plusieurs fichiers et en limitant les risques de conflit

# Modules JavaScript - IIFE

---

## ► Immediately-invoked function expression (IIFE)

```
// jquery-button.js
(function($, global) {
  'use strict';

  function MonBouton(options) {
    this.options = options || {};
    this.value = options.value || 'Valider';
  }

  MonBouton.prototype.creer = function(container) {
    $(container).append('<button>'+this.value+'</button>');
  };

  global.MonBouton = MonBouton;
})(jQuery, window);
```

## ► Une fonction anonyme appelée immédiatement

- Limite la portée des variables
- Permet de renommer localement des dépendances

# Modules JavaScript - IIFE

---

## ► Utilisation

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Exemple</title>
</head>
<body>
  <div id="container"></div>
  <script src="http://code.jquery.com/jquery-1.11.3.min.js"></script>
  <script src="jquery-button.js"></script>
  <script>
    var button = new MonBouton({
      value: 'Cliquez ici'
    });

    button.creer('#container');
  </script>
</body>
</html>
```

## ► Inconvénients

- L'ordre d'inclusion des scripts doit être connu (ici jQuery avant jquery-button)
- Les modules reçoivent leur dépendances via des variables globales (jQuery, window)
- Les modules exposent leur code via des variables globales (global.MonBouton)

# Modules JavaScript - YUI

---

## ► Modules YUI

Yahoo User Interface library (plus maintenue depuis mi-2014)

Première bibliothèque à introduire la notion de modules

<http://yuilibrary.com/yui/docs/yui/create.html>

```
// yui-button.js
YUI().add('mon-bouton', function (Y) {
    'use strict';

    function MonBouton(options) {
        this.options = options || {};
        this.value = options.value || 'Valider';
    }

    MonBouton.prototype.creer = function(container) {
        Y.one(container).append('<button>'+this.value+'</button>')
    };

    Y.MonBouton = MonBouton;
}, '0.0.1', {
    requires: ['node']
});
```

- Un module YUI décrit ses dépendances (requires: ['node'] pour accéder aux méthodes on et append)
- Pas d'utilisation de variables globales



# Modules JavaScript - YUI

---

## ► Utilisation

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Exemple</title>
</head>
<body>
  <div id="container"></div>
  <script src="http://yui.yahooapis.com/3.18.1/build/yui/yui-min.js"></script>
  <script>
    YUI({
      modules: {
        'mon-bouton': 'yui-button.js'
      }
    }).use('mon-bouton', function (Y) {
      var bouton = new Y.MonBouton({
        value: 'Cliquez ici'
      });

      bouton.creer('#container');
    });
  </script>
</body>
</html>
```

- Le fichier yui-button.js est inclus automatiquement
- Pas besoin de connaître l'ordre d'inclusion des dépendances



# Modules JavaScript - CommonJS

---

## ► CommonJS

Projet visant à créer des API communs pour du développement JavaScript hors navigateur (console, GUI...)

Exemple : standardiser l'accès aux fichiers

Le projet propose une norme pour le chargement de modules utilisé entre autre par Node.js

<http://www.commonjs.org/specs/modules/1.0/>

## ► Création d'un module

```
// calculatrice.js
exports.ajouter = function(nb1, nb2) {
  return Number(nb1) + Number(nb2);
};
```

- Les modules communs JS exposent à l'intérieur d'un module une variable exports de type object (et qui peut être écrasée si besoin)

# Modules JavaScript - CommonJS

---

## ► Utilisation

```
// main.js  
var calc = require('./Calculette');  
  
console.log(calc.ajouter(2, 3)); // 5
```

- CommonJS propose une méthode require pour le chargement de modules, dont le retour correspond à la variable exports
- Cependant CommonJS ne s'applique pas au navigateur où le chargement de fichiers se fait via la balise script

# Modules JavaScript - CommonJS + browserify

---

- ▶ **Browserify**

Permet de charger des modules CommonJS côté client.

- ▶ **Installation :**

`npm install -g browserify`

- ▶ **Transformation en code client :**

`browserify main.js > calculette-browser.js`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <script src="calculette-browser.js"></script>
</body>
</html>
```

# Modules JavaScript - AMD

---

- ▶ **Asynchronous Module Definition**

CommonJS ne permettant pas d'exécuter de charger des modules côté client, AMD est né.

- ▶ **RequireJS**

Plusieurs bibliothèques permettent de charger des modules AMD, RequireJS est la plus connue.

<http://requirejs.org/>

- ▶ RequireJS définit 2 fonctions globales `require` et `define`. `define` permet de définir un module, `require` est le point d'entrée de l'application.



# Modules JavaScript - AMD

---

```
// number-converter.js
define(function() {
    var exports = {};

    exports.convert = function(nb) {
        return Number(nb);
    };

    return exports;
});
```

```
// calculette.js
define(['number-converter'], function(numberConverter) {
    var exports = {};

    exports.ajouter = function(nb1, nb2) {
        return numberConverter.convert(nb1) + numberConverter.convert(nb2);
    };

    return exports;
});
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
    <script src="bower_components/requirejs/require.js"></script>
    <script>
        require(['calculette'], function(calc) {
            console.log(calc.ajouter(2, 3)); // 5
        });
    </script>
</body>
</html>
```

# Modules JavaScript - ECMAScript 2015 / ES6

---

- ▶ **ECMAScript 2015 / ECMAScript 6**

La nouvelle version de JavaScript prévoit une syntaxe pour l'utilisation de module. A l'heure actuelle (juillet 2015), ni les navigateurs ni Node.js ou io.js ne supportent cette syntaxe.

- ▶ **Babel / Traceur**

Babel et Traceur sont des bibliothèques qui permettent de transpiler du code ES6 en ES5 et ainsi l'utiliser sur les moteurs actuels.

- ▶ **Installation :**

`npm install -g babel-cli`

- ▶ **Utilisation (toutes les sources du répertoires src vers le répertoire dist) :**

`babel src --out-dir dist/`

# Modules JavaScript - ECMAScript 2015 / ES6

---

```
// src/number-converter.js
var exports = {};

exports.convert = function(nb) {
  return Number(nb);
};

export default exports;
```

```
// src/calcullette.js
import numberConverter from './number-converter';

var exports = {};

exports.ajouter = function(nb1, nb2) {
  return numberConverter.convert(nb1) + numberConverter.convert(nb2);
};

export default exports;
```

```
// src/main.js
import calc from './calcullette';

console.log(calc.ajouter(2, 3)); // 5
```



# Modules JavaScript - UMD

## ► Universal Module Definition

L'objectif d'UMD est de proposer des modules compatibles CommonJS, AMD ou en utilisant des variables globales si le contexte ne permet pas d'utiliser les 2 précédents.

<https://github.com/umdjs/umd>

```
// number-converter.js
(function (root, factory) {
  if (typeof exports === 'object') {
    // CommonJS
    module.exports = factory();
  } else if (typeof define === 'function' && define.amd) {
    // AMD
    define(function () {
      return (root.numberConverter = factory());
    });
  } else {
    // Global Variables
    root.numberConverter = factory();
  }
})(this, function () {
  var exports = {};

  exports.convert = function(nb) {
    return Number(nb);
  };

  return exports;
});
```

```
// calculette.js
(function (root, factory) {
  if (typeof exports === 'object') {
    // CommonJS
    module.exports = factory(require('./number-converter'));
  } else if (typeof define === 'function' && define.amd) {
    // AMD
    define(['./number-converter'], function (numberConverter) {
      return (root.calculette = factory(numberConverter));
    });
  } else {
    // Global Variables
    root.calculette = factory(root.numberConverter);
  }
})(this, function (numberConverter) {
  var exports = {};

  exports.ajouter = function(nb1, nb2) {
    return numberConverter.convert(nb1) +
    numberConverter.convert(nb2);
  };

  return exports;
});
```

# Modules JavaScript - System.js

---

- ▶ **System.js**

System.js est un loader universel qui sait charger des modules CommonJS, AMD, ES6 et IIFE dans les navigateurs et sous node.js

<https://github.com/systemjs/systemjs>

# Supersets JavaScript

# Supersets JavaScript - Introduction

---

- ▶ A l'instar des préprocesseur CSS, les Supersets JavaScript sont des surcouches qui transpilent en JavaScript
- ▶ Ils ajoutent des concepts inspirés d'autres langages (types, structures, sucre syntaxique...)
- ▶ La prochaine norme du langage JavaScript, ECMAScript 2015 (anciennement ECMAScript 6) peut également être utilisé comme un Superset, les navigateurs actuels (juillet 2015) ne le supportant pas encore.

# Supersets JavaScript - CoffeeScript

---

- Apparue en 2009  
<http://coffeescript.org/>
- Inspiré de Ruby (populaire chez ces développeurs, Ruby on Rails l'intègre depuis sa version 3.1)
- 42208 projets sur GitHub (juillet 2015)
- Offre principalement du sucre syntaxique



# Supersets JavaScript - CoffeeScript

---

- ▶ **Installation**

`npm install -g coffeescript`

- ▶ **Compilation**

`coffee --compile --output dist/ src/`

# Supersets JavaScript - CoffeeScript

---

## ► Variables et conditions

```
# CoffeeScript
aleatoire = 100 * Math.random()

console.log aleatoire # 46.67751151137054
console.log "Entre 25 et 50 " if 25 < aleatoire < 50 # Entre 25 et 50
```

```
// Generated by CoffeeScript 1.9.3
(function() {
  var aleatoire;

  aleatoire = 100 * Math.random();

  console.log(aleatoire);

  if ((25 < aleatoire && aleatoire < 50)) {
    console.log("Entre 25 et 50 ");
  }

}).call(this);
```

# Supersets JavaScript - CoffeeScript

---

## ► Arrow functions

```
# CoffeeScript
square = (x) -> x * x
console.log square 3 # 9
```

```
// Generated by CoffeeScript 1.9.3
(function() {
  var square;

  square = function(x) {
    return x * x;
  };

  console.log(square(3));
}).call(this);
```



# Supersets JavaScript - CoffeeScript

---

## ► Classes

```
# CoffeeScript
class Personne
  constructor: (@prenom) ->
  hello: -> "Je m'appelle #{@prenom}";

romain = new Personne "Romain"
console.log romain.hello() # Je m'appelle Romain
```

```
// Generated by CoffeeScript 1.9.3
(function() {
  var Personne, romain;

  Personne = (function() {
    function Personne(prenom) {
      this.prenom = prenom;
    }

    Personne.prototype.hello = function() {
      return "Je m'appelle " + this.prenom;
    };

    return Personne;
  })();

  romain = new Personne("Romain");

  console.log(romain.hello());
}).call(this);
```

# Supersets JavaScript - CoffeeScript

---

## ► Heritage

```
# CoffeeScript
class Personne
  constructor: (@prenom) ->
  hello: -> "Je m'appelle #{@prenom}"

class Formateur extends Personne
  constructor: (prenom, @specialite) -> super(prenom)
  hello: -> super() + ", ma spécialité est #{@specialite}"

romain = new Formateur "Romain", "CoffeeScript"
console.log romain.hello() # Je m'appelle Romain, ma spécialité est CoffeeScript
```

# Supersets JavaScript - CoffeeScript

```
// Generated by CoffeeScript 1.9.3
(function() {
  var Formateur, Personne, romain,
    extend = function(child, parent) { for (var key in parent) { if (hasProp.call(parent, key)) child[key] =
parent[key]; } function ctor() { this.constructor = child; } ctor.prototype = parent.prototype; child.prototype = new
ctor(); child.__super__ = parent.prototype; return child; },
    hasProp = {}.hasOwnProperty;

  Personne = (function() {
    function Personne(prenom1) {
      this.prenom = prenom1;
    }

    Personne.prototype.hello = function() {
      return "Je m'appelle " + this.prenom;
    };

    return Personne;
  })();

  Formateur = (function(superClass) {
    extend(Formateur, superClass);

    function Formateur(prenom, specialite) {
      this.specialite = specialite;
      Formateur.__super__.constructor.call(this, prenom);
    }

    Formateur.prototype.hello = function() {
      return Formateur.__super__.hello.call(this) + (" , ma spécialité est " + this.specialite);
    };

    return Formateur;
  })(Personne);

  romain = new Formateur("Romain", "CoffeeScript");

  console.log(romain.hello());
}).call(this);
```

# Supersets JavaScript - TypeScript

---

- ▶ Créé par Microsoft en 2012  
<http://www.typescriptlang.org>
- ▶ Supersets ECMAScript 2015
- ▶ 3881 projets sur GitHub (juillet 2015)
- ▶ La version 2 d'AngularJS incite à son utilisation (en plus de supporter Dart, ES5 et ES6)  
<https://angular.io/>

TypeScript

# Supersets JavaScript - TypeScript

---

- ▶ **Installation**

`npm install -g typescript`

- ▶ **Compilation**

`tsc --outDir dist/ src/*`

# Supersets JavaScript - TypeScript

---

## ► Types

```
// TypeScript
var prenom: string = "Romain";
console.log(prenom.toUpperCase());

var prenoms: Array<string> = ["Romain", "Eric", "Jean"];
prenoms.forEach(function(elt) {
    console.log(prenom.toUpperCase());
});
```

```
var prenom = "Romain";
console.log(prenom.toUpperCase());
var prenoms = ["Romain", "Eric", "Jean"];
prenoms.forEach(function (elt) {
    console.log(prenom.toUpperCase());
});
```

# Supersets JavaScript - TypeScript

---

## ► Interfaces

```
// TypeScript
function avecInterfaceAnonyme(config?: {couleur?: string, valeur?: string}) {
    config = config || {};
    config.couleur = config.couleur || 'blanc';
    config.valeur = config.valeur || 'texte';

    return 'Retourne un élément ' + config.couleur + ' dont le contenu est ' + config.valeur;
}

console.log(avecInterfaceAnonyme({couleur: 'bleu'})); // Retourne un élément bleu dont le
contenu est texte
```

```
function avecInterfaceAnonyme(config) {
    config = config || {};
    config.couleur = config.couleur || 'blanc';
    config.valeur = config.valeur || 'texte';
    return 'Retourne un élément ' + config.couleur + ' dont le contenu est ' + config.valeur;
}

console.log(avecInterfaceAnonyme({ couleur: 'bleu' })); // Retourne un élément bleu dont le
contenu est texte
```

# Supersets JavaScript - TypeScript

---

## ► Interfaces (nommée)

```
// TypeScript
interface Config {
    couleur?: string;
    valeur?: string;
}

function avecInterfaceConfig(config?: Config) {
    config = config || {};
    config.couleur = config.couleur || 'blanc';
    config.valeur = config.valeur || 'texte';

    return 'Retourne un élément ' + config.couleur + ' dont le contenu est ' + config.valeur;
}

console.log(avecInterfaceConfig({couleur: 'bleu'})); // Retourne un élément bleu dont le
contenu est texte
```

```
function avecInterfaceConfig(config) {
    config = config || {};
    config.couleur = config.couleur || 'blanc';
    config.valeur = config.valeur || 'texte';
    return 'Retourne un élément ' + config.couleur + ' dont le contenu est ' + config.valeur;
}

console.log(avecInterfaceConfig({ couleur: 'bleu' })); // Retourne un élément bleu dont le
contenu est texte
```



# Supersets JavaScript - TypeScript

---

## ► Classes

```
// TypeScript
class Personne {
    prenom: string;
    constructor(prenom: string) {
        this.prenom = prenom;
    }
    hello(): string {
        return "Je m'appelle " + this.prenom;
    }
}

var romain = new Personne("Romain")
console.log(romain.hello()); // Je m'appelle Romain
```

```
var Personne = (function () {
    function Personne(prenom) {
        this.prenom = prenom;
    }
    Personne.prototype.hello = function () {
        return "Je m'appelle " + this.prenom;
    };
    return Personne;
})();

var romain = new Personne("Romain");
console.log(romain.hello()); // Je m'appelle Romain
```

# Supersets JavaScript - TypeScript

---

## ► Héritage

```
// TypeScript
class Personne {
    prenom: string;
    constructor(prenom: string) {
        this.prenom = prenom;
    }
    hello(): string {
        return "Je m'appelle " + this.prenom;
    }
}

class Formateur extends Personne {
    specialite: string;
    constructor(prenom: string, specialite: string) {
        super(prenom);
        this.specialite = specialite;
    }
    hello(): string {
        return super.hello() + ', ma spécialité est ' + this.specialite;
    }
}

var romain = new Formateur('Romain', 'TypeScript')
console.log(romain.hello()); // Je m'appelle Romain, ma spécialité est TypeScript
```

# Supersets JavaScript - TypeScript

---

## ► Héritage

```
var __extends = (this && this.__extends) || function (d, b) {
    for (var p in b) if (b.hasOwnProperty(p)) d[p] = b[p];
    function __() { this.constructor = d; }
    __.prototype = b.prototype;
    d.prototype = new __();
};

var Personne = (function () {
    function Personne(prenom) {
        this.prenom = prenom;
    }
    Personne.prototype.hello = function () {
        return "Je m'appelle " + this.prenom;
    };
    return Personne;
})();

var Formateur = (function (_super) {
    __extends(Formateur, _super);
    function Formateur(prenom, specialite) {
        _super.call(this, prenom);
        this.specialite = specialite;
    }
    Formateur.prototype.hello = function () {
        return _super.prototype.hello.call(this) + ', ma spécialité est ' + this.specialite;
    };
    return Formateur;
})(Personne);

var romain = new Formateur('Romain', 'TypeScript');
console.log(romain.hello()); // Je m'appelle Romain
```

# Supersets JavaScript - TypeScript

---

## ► Modules

# Argument for '--module' option must be 'commonjs', 'amd', 'system' or 'umd'.

tsc --module amd --outDir dist/modules src/modules/\*

```
// Voiture.ts
var marqueDefault: string = 'Renault';

class Voiture {
    marque: string;
    constructor(marque?: string) {
        this.marque = marque || marqueDefault;
    }
    infos(): string {
        return 'Voiture de marque ' + this.marque;
    }
}

export = Voiture;
```

```
define(["require", "exports"], function (require, exports) {
    var marqueDefault = 'Renault';
    var Voiture = (function () {
        function Voiture(marque) {
            this.marque = marque || marqueDefault;
        }
        Voiture.prototype.infos = function () {
            return 'Voiture de marque ' + this.marque;
        };
        return Voiture;
    })();
    return Voiture;
});
```

# Supersets JavaScript - ECMAScript 2015 / ES6

---

- ▶ **ECMAScript 2015 / ECMAScript 6**

La nouvelle version de JavaScript prévoit une syntaxe pour l'utilisation de module. A l'heure actuelle (juillet 2015), ni les navigateurs ni Node.js ou io.js ne supportent cette syntaxe.

- ▶ **Babel / Traceur**

Babel et Traceur sont des bibliothèques qui permettent de transpiler du code ES6 en ES5 et ainsi l'utiliser sur les moteurs actuels.

- ▶ **Installation :**

`npm install -g babel`

- ▶ **Utilisation (toutes les sources du répertoires src vers le répertoire dist) :**

`babel src --out-dir dist/`



# Supersets JavaScript - ECMAScript 2015 / ES6

---

## ► Nouveautés

<https://babeljs.io/docs/learn-es2015/>

<http://es6-features.org/>

- Arrows
- Classes
- Template Strings
- Default + Rest + Spread
- Let + Const
- Modules
- Module Loaders
- Map + Set + WeakMap + WeakSet
- ...

## ► Compatibilité

<https://kangax.github.io/compat-table/es6/>

# Supersets JavaScript - Dart

---

- ▶ Créé par Google en 2013
- ▶ Permet de créer de programmes serveurs (VM), mobile (VM) et transpile en JavaScript pour les navigateurs
- ▶ Syntaxe identique à C/C++
- ▶ 3868 projets sur GitHub (juillet 2015)

```
void main()  
{  
  print("Hello !");  
}
```



# Supersets JavaScript - Haxe

---

- Apparu en 2005
- Projet français permettant de transpiler dans différents language de programmation dont JavaScript
- 3178 projets sur GitHub (juillet 2015)

```
class HelloWorld {  
    static public function main() {  
        trace("Hello World");  
    }  
}
```





# Node.js

---

# Node.js - Introduction

---

- ▶ Créé 2009 par Ryan Dahl

A l'origine, Ryan Dahl voulait simplifier la création d'une barre d'upload.

- ▶ Sponsorisé par la société Joyent.

- ▶ Un programme en ligne de commande combinant :

- le moteur JavaScript V8 de Chrome
- une boucle d'événement
- une gestion bas niveau des entrées/sorties

- ▶ Un système en production :

- Chez des startups à la pointe : Airbnb, ...
- Dans des grands groupes : Microsoft, PayPal, Walmart, LinkedIn

# Node.js - Installation

---

- ▶ **Windows**

Exécutables : <https://nodejs.org/download/>

- ▶ **OS X**

Exécutables : <https://nodejs.org/download/>

Ou via homebrew : `brew install node`

- ▶ **Debian / Ubuntu**

`sudo apt-get update`

`sudo apt-get install nodejs npm`

- ▶ **Pensez à ajouter le répertoire de Node au Path.**

# Node.js - Exécution

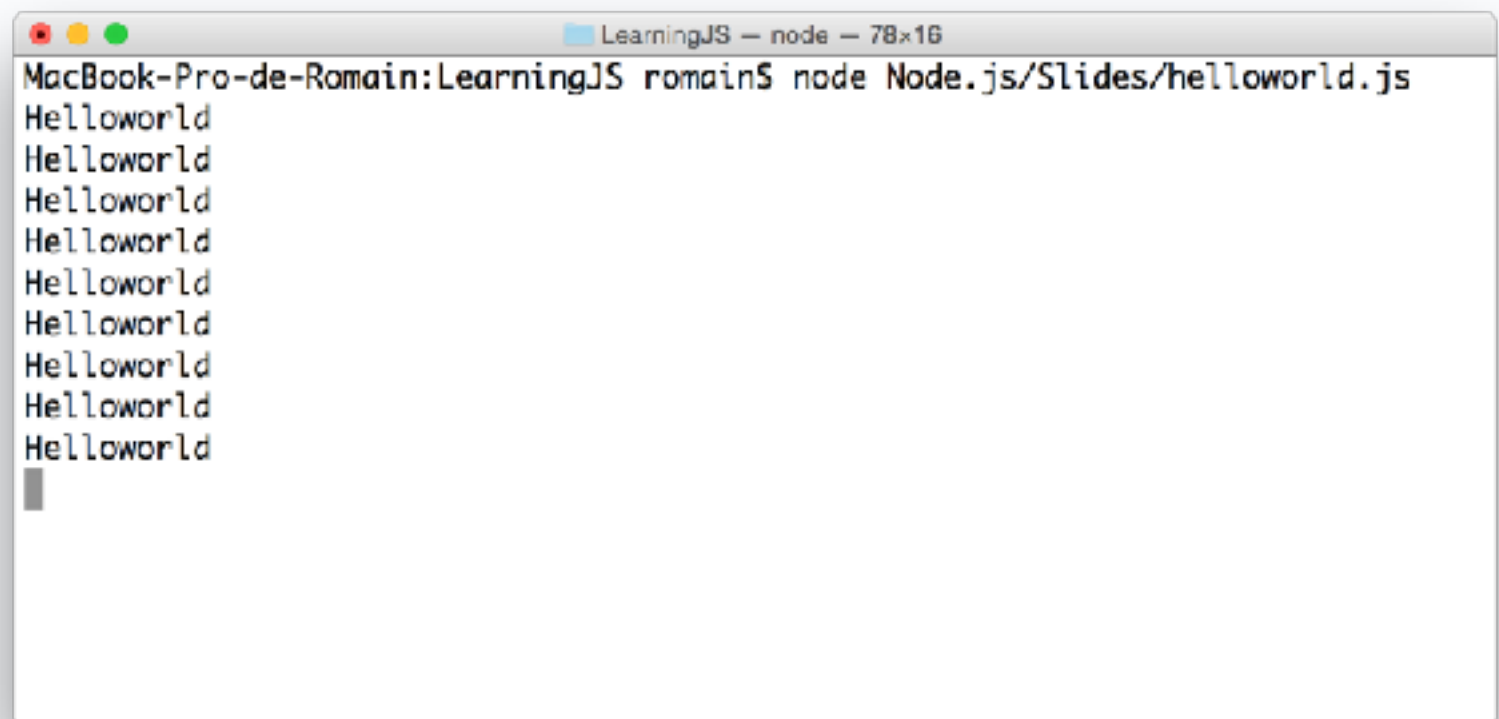
```
/* Un simple helloworld */  
  
/** @function helloworld */  
function helloworld() {  
  'use strict'; // bonne pratique  
  console.log('Helloworld');  
}  
  
setInterval(helloworld, 1000);
```

- ▶ Lancement du programme

node FILE\_PATH.js

- ▶ Interruption

CTRL-C



A terminal window titled "LearningJS - node - 78x16" showing the execution of the script. The prompt is "MacBook-Pro-de-Romain:LearningJS romain\$ node Node.js/Slides/helloworld.js". The output consists of ten lines of "Helloworld" printed at regular intervals. A cursor is visible at the bottom of the terminal.

# Node.js - Module File System

---

- Parcourir un répertoire

## Asynchrone

```
var fs = require('fs');  
fs.readdir('.', function (err, files) {  
  if (err) throw err;  
  console.log(files);  
});
```

## Synchrone

```
var fs = require('fs');  
console.log(fs.readdirSync('.'));
```

# Node.js - Module File System

---

- Lire un fichier

## Asynchrone

```
var fs = require('fs');

fs.readFile('lorem.txt', {encoding: 'UTF-8'}, function (err, content) {
  if (err) throw err;
  console.log(content);
});
```

## Synchrone

```
var fs = require('fs');

var content = fs.readFileSync('lorem.txt', {encoding: 'UTF-8'});
console.log(content);
```

# Node.js - Module File System

---

## ▸ Ecrire dans un fichier

### Asynchrone

```
var fs = require('fs');
var moment = require('moment');

var date = moment().format('DD/MM/YYYY à HH:mm:ss');
var message = 'Ligne loguée le ' + date;

fs.writeFile('log.txt', message + '\n', {flag: 'a'}, function (err) {
  if (err) throw err;
  console.log('Log enregistré ! ('+message+')');
});
```

### Synchrone

```
var fs = require('fs');
var moment = require('moment');

var date = moment().format('DD/MM/YYYY à HH:mm:ss');
var message = 'Ligne loguée le ' + date;

fs.writeFileSync('log.txt', message + '\n', {flag: 'a'});
```

# Gestion de dépendances

---



# Gestion de dépendances - npm

---

- Gestionnaire de dépendance de node.js (s'installe en même temps que node)
- Equivalent pour du code JavaScript à apt-get
- Plutôt destiné à du code console ou serveur, bien que des bibliothèques comme jQuery ou Bootstrap y soient présentes



# Gestion de dépendances - npm

---

- ▶ Trouver des packages

<https://www.npmjs.com>

- ▶ Créer un package

npm init

- ▶ Le fichier package.json

<http://browsenpm.org/package.json>

# Gestion de dépendances - npm

---

- ▶ **Installer un package**

`npm install <package>`

`npm install <package> --save`

`npm install <package>@<version> --save`

Ex : `npm install jquery@1.11.*`

- ▶ **Mettre à jour les packages installés**

`npm update`

- ▶ **Désinstaller**

`npm uninstall lodash`

`npm uninstall --save lodash`

# Gestion de dépendances - bower

---

- ▶ **Bower**

Gestionnaire de dépendance pour bibliothèques front-end (CSS/JS/Polices...). Créé par Twitter en 2012

- ▶ **Pré-requis**

Node.js

Git

- ▶ **Installation**

`npm install -g bower`

- ▶ **Créer un projet**

`bower init`

- ▶ **Trouver des packages**

<http://bower.io/search/>



# Gestion de dépendances - bower

---

- ▶ Installer un package

`bower install <package>`

`bower install <package>#<version>`

Ex : `bower install jquery#1.11.*`

- ▶ Mettre à jour

`bower update`

- ▶ Configuration

Fichier `.bower_rc`

<http://bower.io/docs/config/>

- ▶ Dépôts privés :

<https://github.com/bower/registry>



# Git

---

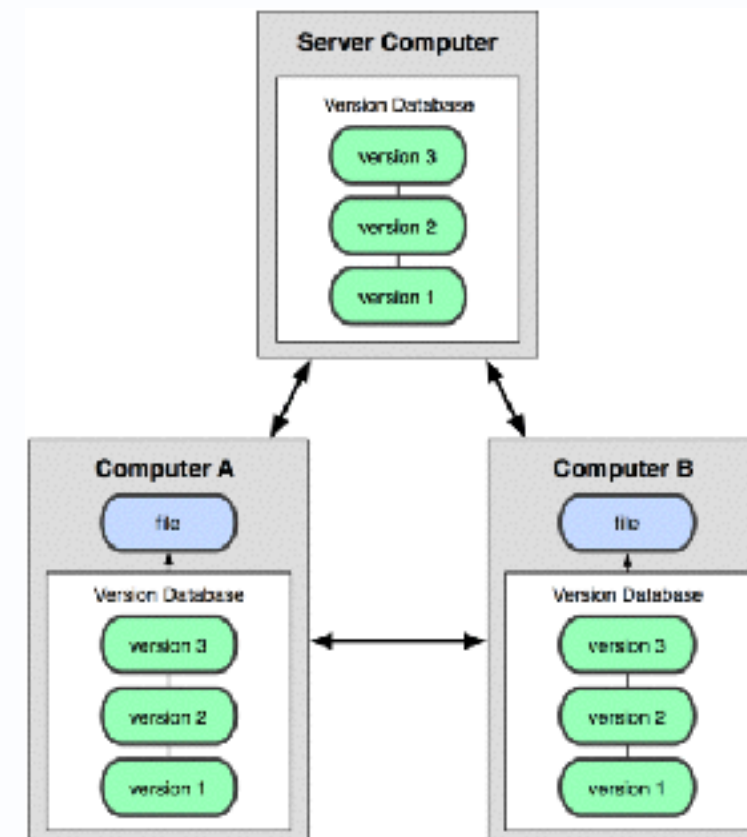
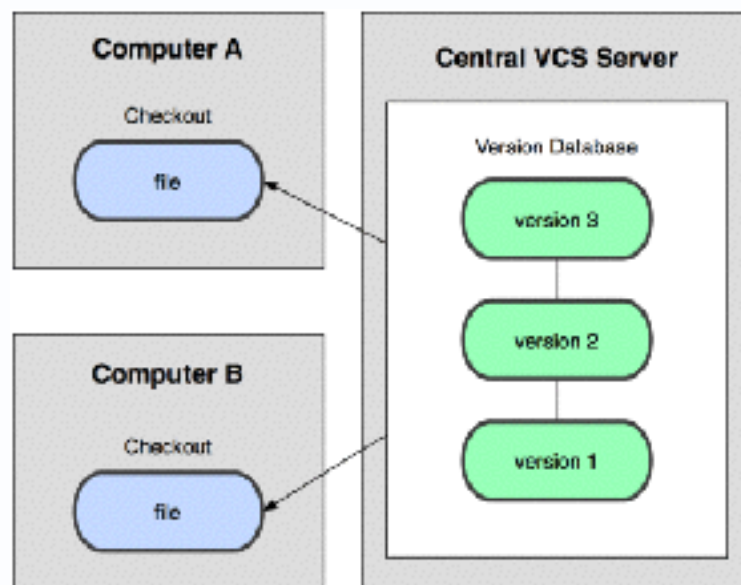
# Git - Introduction

---

- ▶ Système de gestion de version distribué (DVCS)
- ▶ Créé par Linus Torvalds en 2005 pour gérer le code source du noyau Linux
- ▶ Permet de sauvegarder les différences entre plusieurs versions de fichiers (plutôt texte), principalement du code source
- ▶ Facilite la collaborations entre plusieurs développeurs

# Git - DVCS vs CVCS

- ▶ Système de gestion de version centralisé (CVCS)
- ▶ Ex : CVS, Subversion
- ▶ Système de gestion de version distribué (DVCS)
- ▶ Ex : Git, Mercurial





# Git - DVCS vs CVCS

---

## ► Avantages du DVCS

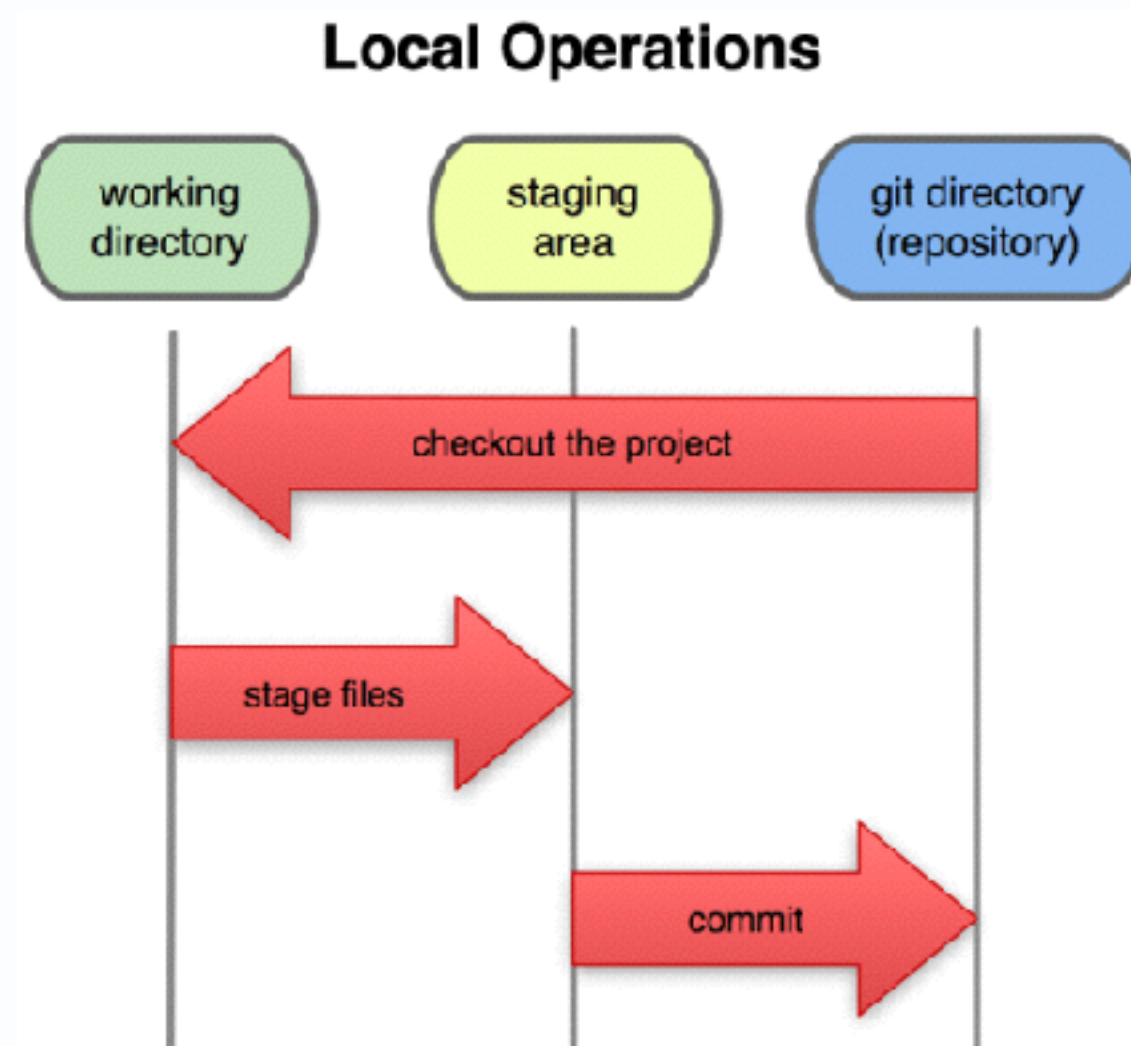
- l'historique des sources est présent sur plusieurs machines (crash de disque...)
- ne pas avoir à être connecté au réseau pour versionner
- permet de collaborer à un projet et obtenir l'autorisation des mainteneurs à posteriori
- plus rapide (accès locaux)

# Git - 3 états

---

## ► 3 états pour les fichiers

- working directory (non-versionnés)
- staging (indexés, à publier lors d'un prochain commit)
- git repository (modifications enregistrées)



# Git - Installation

---

## ▶ Linux

- `yum install git`
- `apt-get install git`

## ▶ Mac OS X

- <http://sourceforge.net/projects/git-osx-installer/>
- `brew install git`

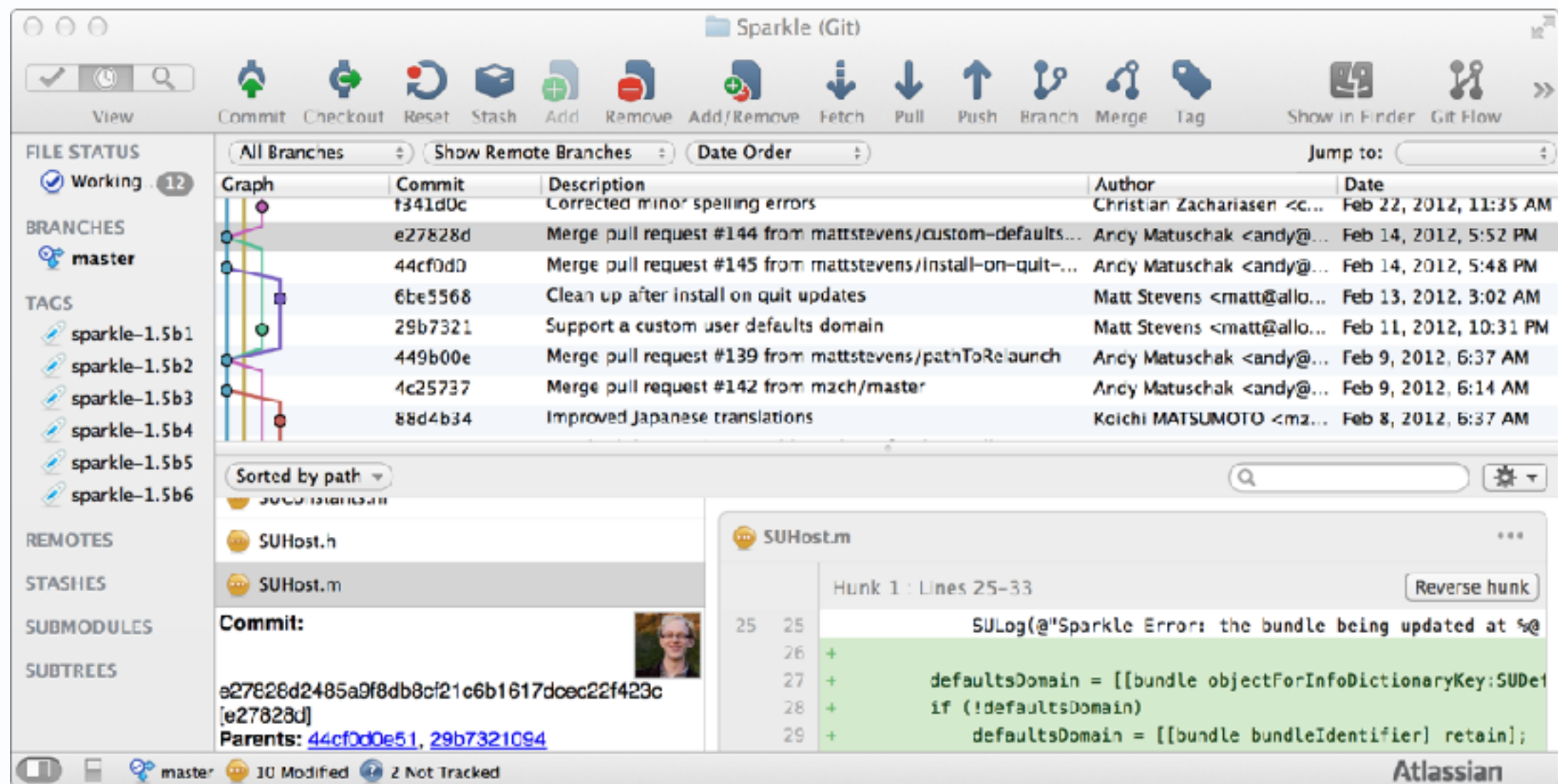
## ▶ Windows

- <http://msysgit.github.io>

# Git - GUI

## ► Mac OS X / Windows

- SourceTree (open-source) : <https://www.sourcetreeapp.com>



# Git - Démarrage

---

## ► Configurer l'utilisateur

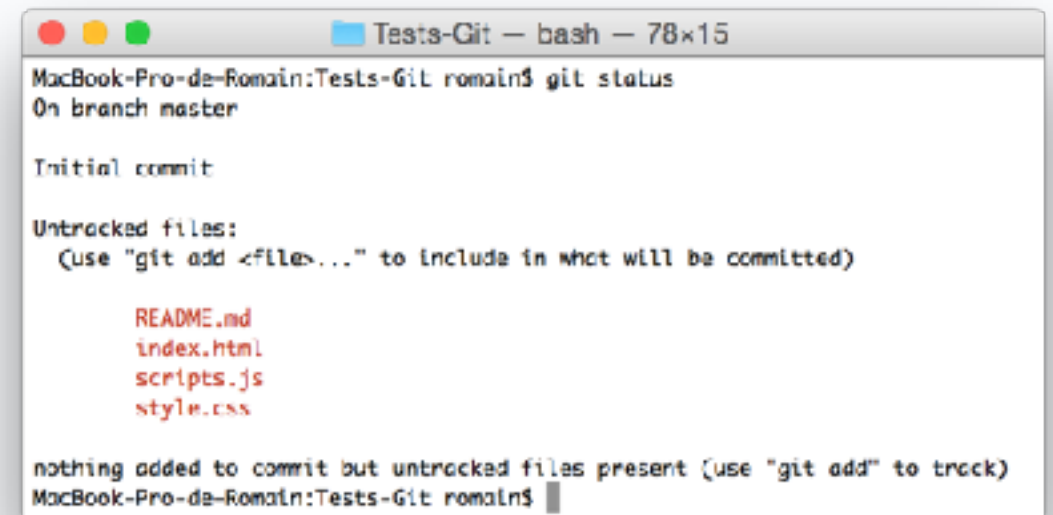
- `git config --global user.name "John Doe"`
- `git config --global user.email johndoe@example.com`

## ► Créer un repository

- `git init`

## ► Obtenir le status du projet

- `git status`

A terminal window titled "Tests-Git — bash — 78x15" showing the output of the `git status` command. The output indicates an initial commit on the master branch with untracked files: README.md, index.html, scripts.js, and style.css. It prompts the user to use `git add` to track these files.

```
MacBook-Pro-de-Romain:Tests-Git romain$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

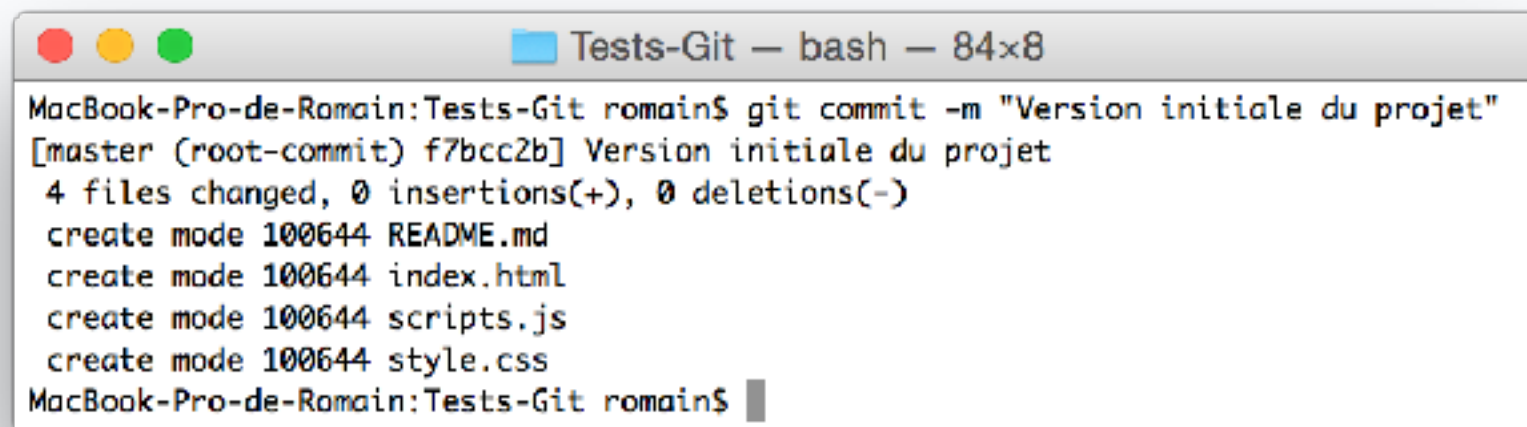
        README.md
        index.html
        scripts.js
        style.css

nothing added to commit but untracked files present (use "git add" to track)
MacBook-Pro-de-Romain:Tests-Git romain$
```

# Git - Démarrage

---

- ▶ Ajouter des sources (nouvelles ou modifiées) à l'index (add ou son alias stage)
  - `git add *.{css,js,html}`
  - `git stage README.md`
- ▶ Versionner
  - `git commit -m "Version initiale du projet"`

A screenshot of a macOS terminal window titled "Tests-Git — bash — 84x8". The terminal shows the execution of the command `git commit -m "Version initiale du projet"`. The output indicates a successful commit on the master branch with the hash `f7bcc2b`. It lists four files that were changed: `README.md`, `index.html`, `scripts.js`, and `style.css`, all created with mode `100644`. The prompt returns to `MacBook-Pro-de-Romain:Tests-Git romain$`.

```
MacBook-Pro-de-Romain:Tests-Git romain$ git commit -m "Version initiale du projet"
[master (root-commit) f7bcc2b] Version initiale du projet
4 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md
create mode 100644 index.html
create mode 100644 scripts.js
create mode 100644 style.css
MacBook-Pro-de-Romain:Tests-Git romain$
```

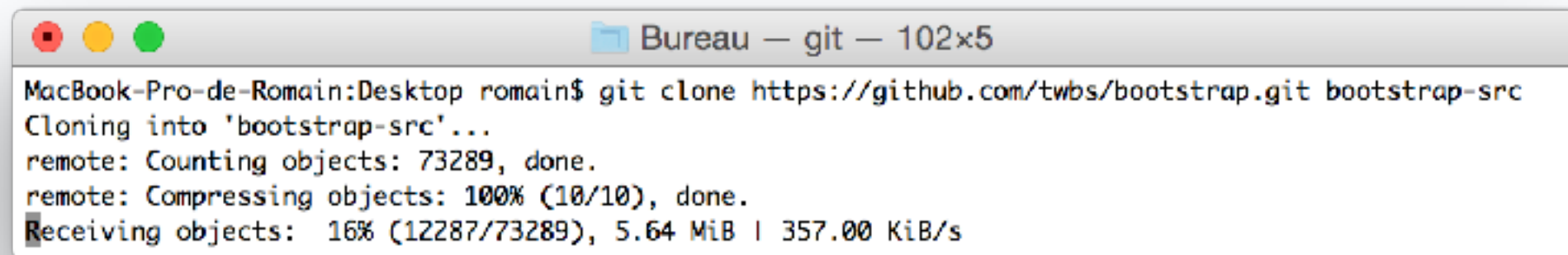
# Git - Démarrage

---

## ► Cloner un repository existant

Opération lente car il faut télécharger tout l'historique.

- `git clone https://github.com/twbs/bootstrap.git bootstrap-src`

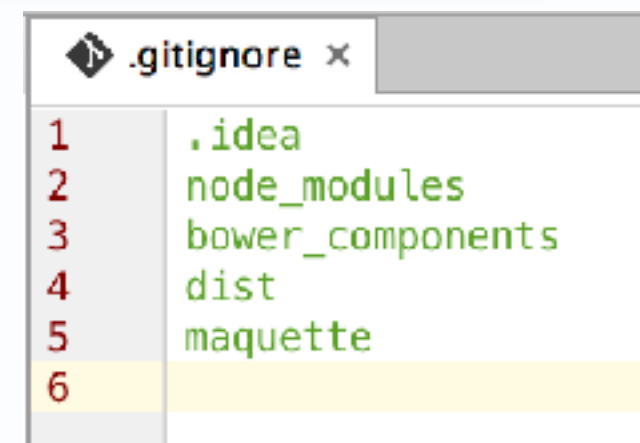


```
MacBook-Pro-de-Romain:Desktop romain$ git clone https://github.com/twbs/bootstrap.git bootstrap-src
Cloning into 'bootstrap-src'...
remote: Counting objects: 73289, done.
remote: Compressing objects: 100% (10/10), done.
Receiving objects: 16% (12287/73289), 5.64 MiB | 357.00 KiB/s
```

## ► Ignorer des fichiers

Créer un fichier `.gitignore`

(peut également se faire dans `.git/info/exclude`)

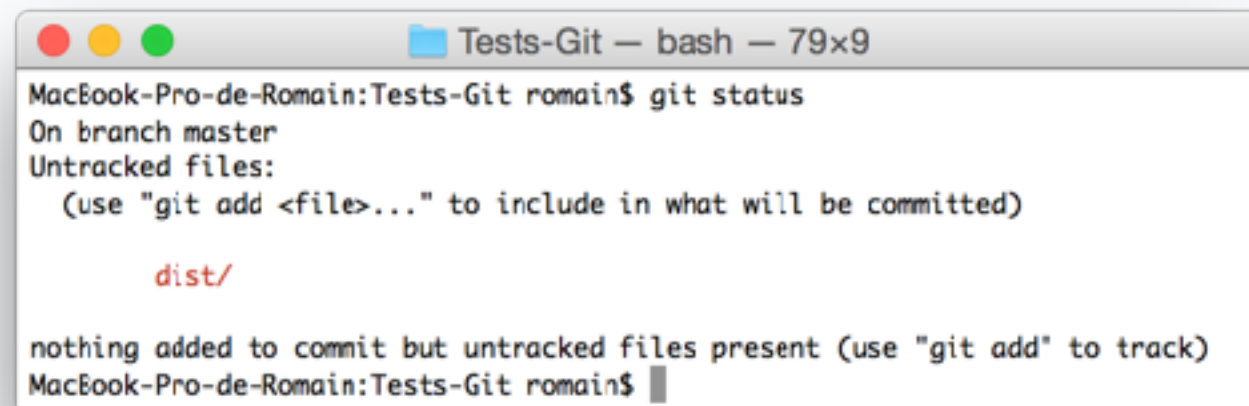


```
.gitignore x
1  .idea
2  node_modules
3  bower_components
4  dist
5  maquette
6
```

# Git - Supprimer un fichier

---

- ▶ Soit le dossier dist, contenant 3 fichiers index.html, scripts.js et style.css
- ▶ On vérifie le status (ici non-indexé)
  - git status

A screenshot of a macOS terminal window titled 'Tests-Git — bash — 79x9'. The terminal shows the output of the 'git status' command. The output indicates that the user is on the 'master' branch and that there are untracked files in the 'dist/' directory. It suggests using 'git add' to track these files. The prompt 'MacBook-Pro-de-Romain:Tests-Git romain\$' is visible at the bottom.

```
MacBook-Pro-de-Romain:Tests-Git romain$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        dist/

nothing added to commit but untracked files present (use "git add" to track)
MacBook-Pro-de-Romain:Tests-Git romain$
```

- ▶ On l'ajoute à l'index puis commit
  - git add dist (ou git stage dist)
  - git commit -m "Ajout du dossier dist"

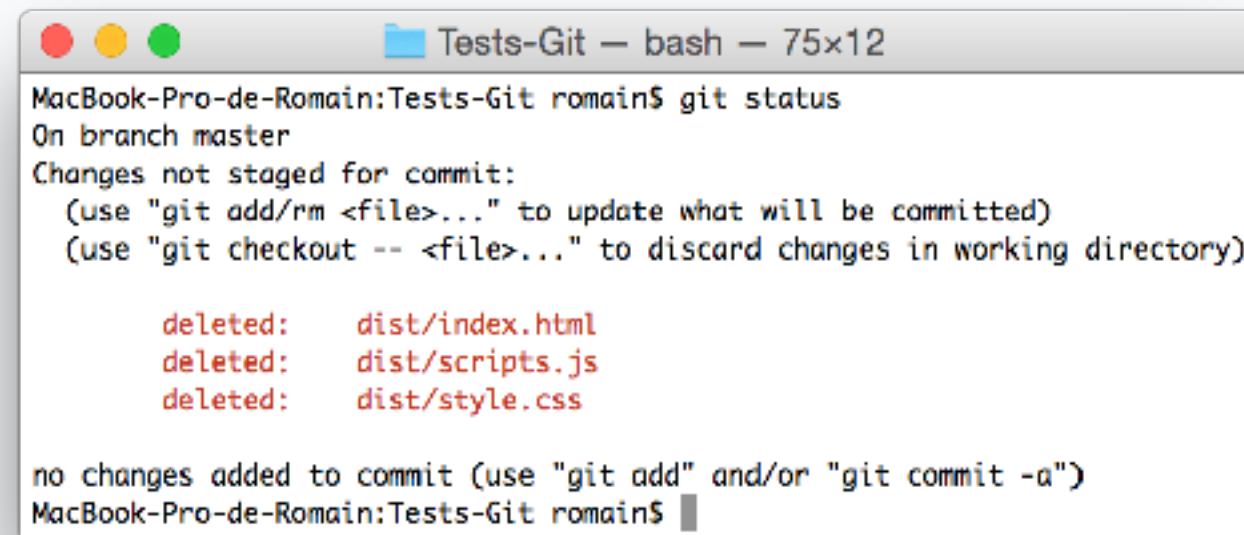


# Git - Supprimer un fichier

---

## ► Supprimer le dossier dist, 3 options :

1. Le supprimer du répertoire de travail (depuis l'explorateur de fichier)
2. Le supprimer de l'index en ligne de commande
  - `git rm -r dist`
3. Le supprimer de l'index tout en le conservant dans le répertoire de travail (utile lorsqu'on oublie un fichier dans `.gitignore`)
  - `git rm -r --cached dist/`

A screenshot of a macOS terminal window titled "Tests-Git — bash — 75x12". The window shows the output of the command "git status". The output indicates that the user is on the "master" branch and that there are changes not staged for commit. These changes are listed as "deleted" files: "dist/index.html", "dist/scripts.js", and "dist/style.css". The terminal also provides instructions on how to stage these changes or discard them. The prompt "MacBook-Pro-de-Romain:Tests-Git romain\$" is visible at the bottom.

```
MacBook-Pro-de-Romain:Tests-Git romain$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

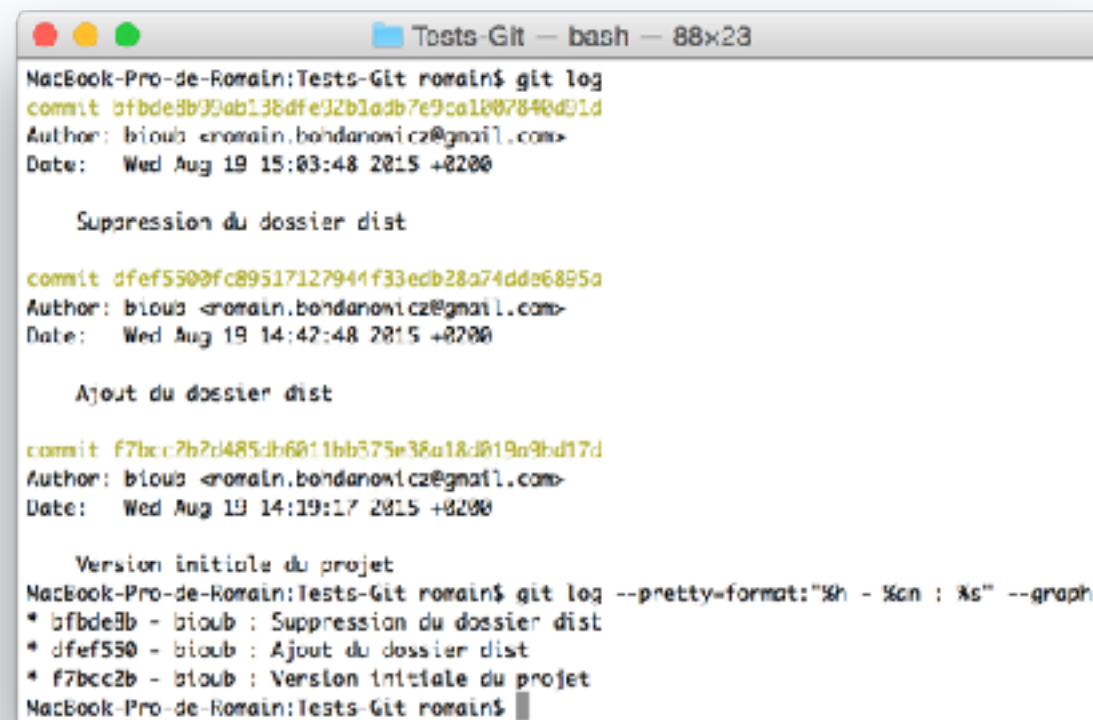
        deleted:    dist/index.html
        deleted:    dist/scripts.js
        deleted:    dist/style.css

no changes added to commit (use "git add" and/or "git commit -a")
MacBook-Pro-de-Romain:Tests-Git romain$
```

# Git - Autres commandes

---

- ▶ Renommer un fichier
  - `git mv nom_origine nom_cible`
- ▶ Historique des modifications
  - `git log`
  - `git log --pretty=format:"%h - %an : %s" --graph`



```
MacBook-Pro-de-Romain:Tests-Git romain$ git log
commit bfbde8b95ab138dfe92b1adb7e9ca1807840d91d
Author: bioub <romain.bondanowicz@gmail.com>
Date:   Wed Aug 19 15:03:48 2015 +0200

    Suppression du dossier dist

commit dfef5509fc89517127941f33edb28a74dde6895a
Author: bioub <romain.bondanowicz@gmail.com>
Date:   Wed Aug 19 14:42:48 2015 +0200

    Ajout du dossier dist

commit f7bcc2b2d485db6011bb575e38a18d019a9bd17d
Author: bioub <romain.bondanowicz@gmail.com>
Date:   Wed Aug 19 14:19:17 2015 +0200

    Version initiale du projet
MacBook-Pro-de-Romain:Tests-Git romain$ git log --pretty=format:"%h - %an : %s" --graph
* bfbde8b - bioub : Suppression du dossier dist
* dfef550 - bioub : Ajout du dossier dist
* f7bcc2b - bioub : Version initiale du projet
MacBook-Pro-de-Romain:Tests-Git romain$
```

# Git - Annuler des actions

---

## ► Modifier le dernier commit

- `git commit -m 'validation initiale'`
- `git add fichier_oublie`
- `git commit --amend`

## ► Désindexer un fichier déjà indexé

- `git reset HEAD fichier_a_desindexer`

## ► Réinitialiser un fichier modifié

- `git checkout fichier_a_reinitialiser`

# Git - Dépôts distants

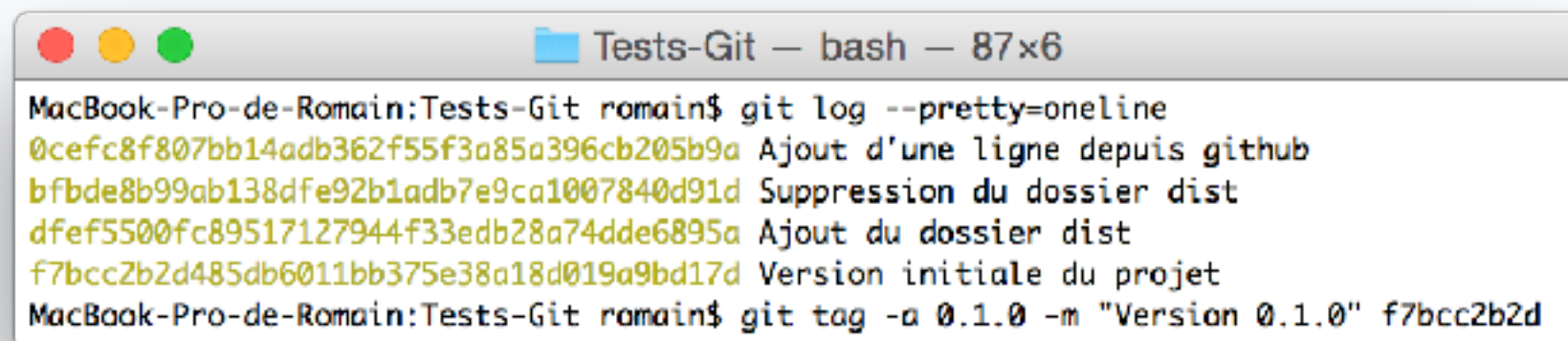
---

- ▶ Ajouter un dépôt distant
  - `git remote add origin https://github.com/bioub/tests-git.git`
- ▶ Listes les dépôts distants
  - `git remote -v`
- ▶ Publier des sources sur un dépôt distant
  - `git push origin master`  
(origin : nom du dépôt distant, master : branche locale)
- ▶ Récupérer les sources depuis un dépôt distant
  - `git pull origin master`

# Git - Tags

---

- Lister les tags
  - `git tag`
- Créer un nouveau tag
  - `git tag -a 0.9.0 -m "Version 0.9.0"`
- Tagger un précédent commit
  - `git tag -a 0.1.0 -m "Version 0.1.0" f7bcc2b2d`



```
MacBook-Pro-de-Romain:Tests-Git romain$ git log --pretty=oneline
0cefc8f807bb14adb362f55f3a85a396cb205b9a Ajout d'une ligne depuis github
bfbde8b99ab138dfe92b1adb7e9ca1007840d91d Suppression du dossier dist
dfef5500fc89517127944f33edb28a74dde6895a Ajout du dossier dist
f7bcc2b2d485db6011bb375e38a18d019a9bd17d Version initiale du projet
MacBook-Pro-de-Romain:Tests-Git romain$ git tag -a 0.1.0 -m "Version 0.1.0" f7bcc2b2d
```

- Partager les tags au serveur distant
  - `git push origin --tags`

# Git - Branches

---

## ▶ Créer une branche

- `git branch fonc12`  
(fonc12 : le nom de la nouvelle branche « fonctionnalité 12 »)

## ▶ Changer de branche

- `git checkout fonc12`

## ▶ Créer et changer de branche

- `git checkout -b fonc12`

## ▶ Récupérer les sources dans une nouvelle branch

- `git fetch origin`

# Git - Branches

## ► Fusionner une branche dans la branche courante

- `git merge fonc12`

## ► 3 cas possibles

- Fast-forward (pas de commit sur la branche d'origine entre temps)

```
MacBook-Pro-de-Romain:Tests-Git romain$ git merge fonc12
Updating 0cefc8f..795dca3
Fast-forward
 scripts.js | 1 +
 1 file changed, 1 insertion(+)
```

```
MacBook-Pro-de-Romain:Tests-Git romain$ git log --pretty=format:"%h - %an : %s" --graph
* 4e6fac9 - bioub : Ajout d'un fond pour body
* 795dca3 - bioub : Ajout d'une ligne dans scripts.js
* 0cefc8f - Romain Bohdanowicz : Ajout d'une ligne depuis github
* bfbde8b - bioub : Suppression du dossier dist
* dfeF550 - bioub : Ajout du dossier dist
* f7bcc2b - bioub : Version initiale du projet
```

- Recursive

```
MacBook-Pro-de-Romain:Tests-Git romain$ git merge fonc12
Merge made by the 'recursive' strategy.
 scripts.js | 1 +
 1 file changed, 1 insertion(+)
```

```
MacBook-Pro-de-Romain:Tests-Git romain$ git log --pretty=format:"%h - %an : %s" --graph
* 32ab820 - bioub : Merge branch 'fonc12'
|
| * e8eaf16 - bioub : Ajout d'un log dans scripts.js
| * 4e6fac9 - bioub : Ajout d'un fond pour body
|/
* 795dca3 - bioub : Ajout d'une ligne dans scripts.js
* 0cefc8f - Romain Bohdanowicz : Ajout d'une ligne depuis github
* bfbde8b - bioub : Suppression du dossier dist
* dfeF550 - bioub : Ajout du dossier dist
* f7bcc2b - bioub : Version initiale du projet
```

# Git - Branches

- Conflit

```
MacBook-Pro-de-Romain:Tests-Git romain$ git merge fonc12
Auto-merging style.css
CONFLICT (content): Merge conflict in style.css
Automatic merge failed; fix conflicts and then commit the result.
MacBook-Pro-de-Romain:Tests-Git romain$
```

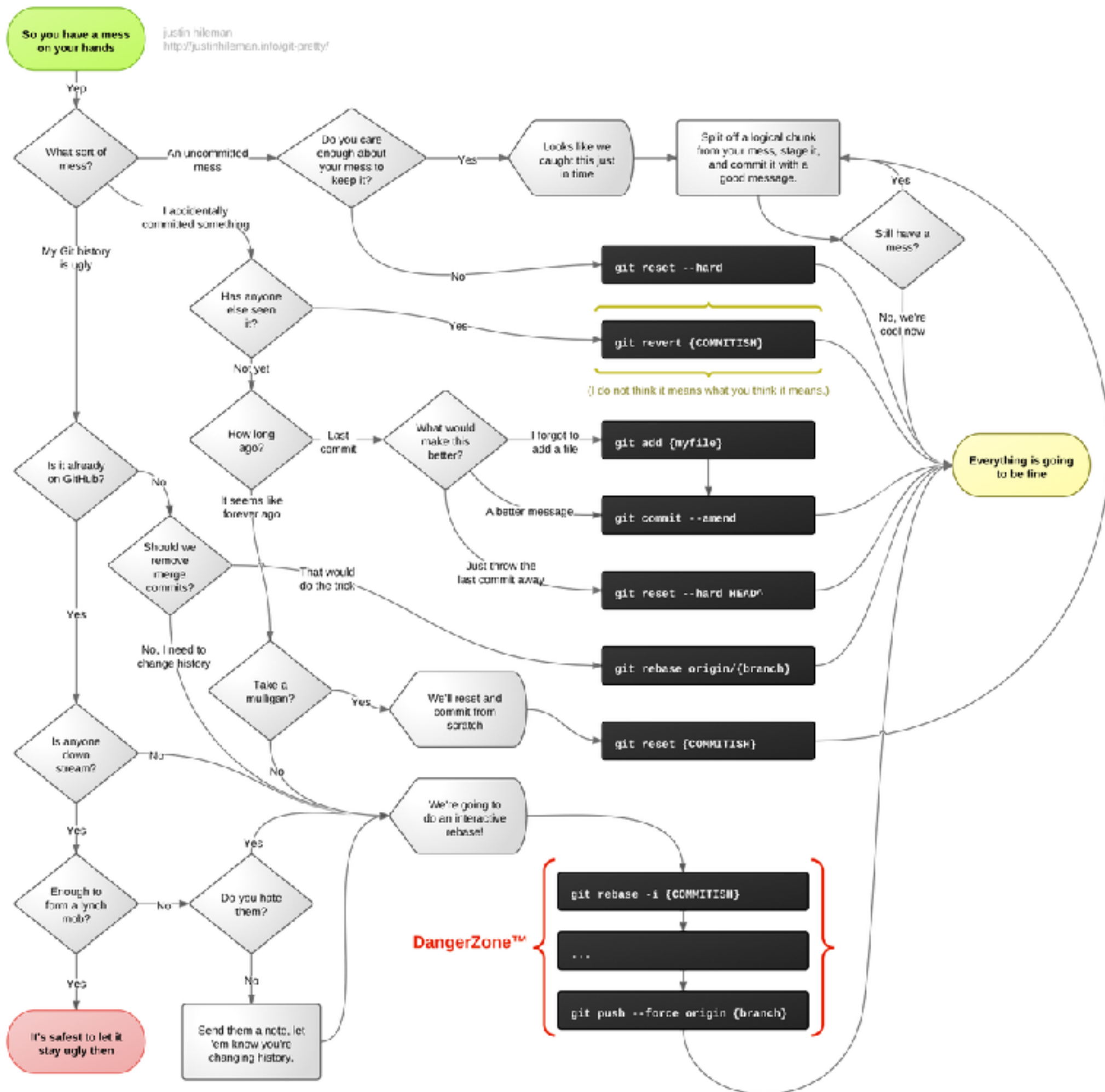
Corriger le fichier qui pose problème  
(penser à retirer les lignes <<<<, =====  
et >>>>)

```
style.css
1 body {
2   background-color: #eee;
3   <<<<<< HEAD
4   color: #222;
5   =====
6   color: #000;
7   >>>>>> fonc12
8 }
9 |
```

Puis commit

```
MacBook-Pro-de-Romain:Tests-Git romain$ git log --pretty=format:"%h - %an : %s" --graph
* 0ecc82e - bioub : Résolution des conflits (texte finalement en gris)
| \
| * b7c733a - bioub : Texte en noir
* | ef1500c - bioub : Texte en gris
* | 32ab820 - bioub : Merge branch 'fonc12'
| \
| \
| * e8eaf16 - bioub : Ajout d'un log dans scripts.js
* | 4e5fac9 - bioub : Ajout d'un fond pour body
| \
* 795dca3 - bioub : Ajout d'une ligne dans scripts.js
* 0cefc8f - Romain Bohdanowicz : Ajout d'une ligne depuis github
* bfbde8b - bioub : Suppression du dossier dist
* dfef550 - bioub : Ajout du dossier dist
* f7bcc2b - bioub : Version initiale du projet
```





# Git - Voir aussi

---

- ▶ Bonnes pratiques de gestion de branches  
<http://nvie.com/posts/a-successful-git-branching-model/>
- ▶ Mise en place simplifiée de ces pratiques  
[http://danielkummer.github.io/git-flow-cheatsheet/index.fr\\_FR.html](http://danielkummer.github.io/git-flow-cheatsheet/index.fr_FR.html)



# Grunt

---

# Grunt - Introduction

---

## ▸ Grunt JS

Permet l'automatisation de tâches de développement front-end.

## ▸ Exemples

- minifier ses fichiers JS
- compiler ses CSS
- compresser les images
- exécuter les tests
- vérifier les conventions de codage

# Grunt - Installation

---

- Installation via npm :  
npm install -g grunt-cli

## Gruntfile.js

```
/*global module:false*/
module.exports = function(grunt) {

  grunt.initConfig({
    copy: {
      dist: {
        src: 'index.html',
        dest: 'dist/index.html'
      }
    },
    uglify: {
      dist: {
        src: 'script.js',
        dest: 'dist/script.js'
      }
    }
  });

  grunt.loadNpmTasks('grunt-contrib-copy');
  grunt.loadNpmTasks('grunt-contrib-uglify');

  // Default task.
  grunt.registerTask('default', ['copy',
  'uglify']);
};
```

## package.json

```
{
  "engines": {
    "node": ">= 0.10.0"
  },
  "devDependencies": {
    "grunt": "^0.4.5",
    "grunt-contrib-copy": "^0.8.0",
    "grunt-contrib-uglify": "^0.9.1"
  }
}
```

# Grunt - Exemple Copie + Obfuscation

src/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <div>
    Prénom : <input type="text" id="prenom">
  </div>
  <p>
    Bonjour <span id="output"></span>
  </p>
  <script src="script.js"></script>
</body>
</html>
```

copy  
→

dist/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <div>
    Prénom : <input type="text" id="prenom">
  </div>
  <p>
    Bonjour <span id="output"></span>
  </p>
  <script src="script.js"></script>
</body>
</html>
```

src/script.js

```
!function() {
  'use strict';

  var inputElt =
document.querySelector('#prenom');
  var outputElt =
document.querySelector('#output');

  inputElt.addEventListener('input', function()
{
    outputElt.innerHTML = inputElt.value;
  });
}();
```

uglify  
→

dist/script.js

```
!function(){"use strict";var
a=document.querySelector("#prenom"),b=document.qu
erySelector("#output");a.addEventListener("input"
,function(){b.innerHTML=a.value}})();
```

# Grunt - Exemple Copie + Obfuscation

---

## Gruntfile.js

```
/*global module:false*/
module.exports = function(grunt) {

  grunt.initConfig({
    copy: {
      dist: {
        src: 'index.html',
        dest: 'dist/index.html'
      }
    },
    uglify: {
      dist: {
        src: 'script.js',
        dest: 'dist/script.js'
      }
    }
  });

  grunt.loadNpmTasks('grunt-contrib-copy');
  grunt.loadNpmTasks('grunt-contrib-uglify');

  // Default task.
  grunt.registerTask('default', ['copy', 'uglify']);
};
```

## Package.json

```
{
  "devDependencies": {
    "grunt": "^0.4.5",
    "grunt-contrib-copy": "^0.8.0",
    "grunt-contrib-uglify": "^0.9.1"
  }
}
```

- Package.json créé avec :  
npm init  
npm install grunt --save-dev  
npm install grunt-contrib-copy --save-dev  
npm install grunt-contrib-uglify --save-dev

# Grunt - Plugins

---

- ▶ Liste des plugins pour grunt :  
<http://gruntjs.com/plugins>  
(4,403 plugins en juillet 2015)
- ▶ Les plugins contrib-\* sont ceux des développeurs de grunt.



# Grunt - Plugins jit-grunt

## ► jit-grunt :

Installation : `npm install jit-grunt --save-dev`

Simplifie le chargement de plugins

### Avant

```
/*global module:false*/  
module.exports = function(grunt) {  
  
  grunt.loadNpmTasks('grunt-contrib-clean');  
  grunt.loadNpmTasks('grunt-contrib-concat');  
  grunt.loadNpmTasks('grunt-contrib-copy');  
  grunt.loadNpmTasks('grunt-contrib-cssmin');  
  grunt.loadNpmTasks('grunt-contrib-jshint');  
  grunt.loadNpmTasks('grunt-contrib-less');  
  grunt.loadNpmTasks('grunt-contrib-uglify');  
  grunt.loadNpmTasks('grunt-contrib-watch');  
  grunt.loadNpmTasks('grunt-google-cdn');  
  grunt.loadNpmTasks('grunt-rev');  
  grunt.loadNpmTasks('grunt-spritesmith');  
  grunt.loadNpmTasks('grunt-usemin');  
  
  grunt.initConfig({  
    // ...  
  });  
  
  // Default task.  
  grunt.registerTask('default', [  
    // ...  
  ]);  
  
};
```

### Après

```
/*global module:false, require*/  
module.exports = function(grunt) {  
  'use strict';  
  
  require('jit-grunt')(grunt, {  
    useminPrepare: 'grunt-usemin',  
    cdnify: 'grunt-google-cdn',  
    sprite: 'grunt-spritesmith'  
  });  
  
  // Project configuration.  
  grunt.initConfig({  
    // ...  
  });  
  
  // Default task.  
  grunt.registerTask('default', [  
    // ...  
  ]);  
  
};
```

# Grunt - Plugins grunt-contrib-less

---

## ► grunt-contrib-less :

npm install grunt-contrib-less --save-dev

Compile des fichiers LESS en CSS

```
module.exports = function(grunt) {  
  
    // ...  
  
    grunt.initConfig({  
        less: {  
            dev: {  
                files: [{  
                    expand: true,  
                    cwd: 'less',  
                    src: ['*.less'],  
                    dest: 'css/',  
                    ext: '.css'  
                }]  
            },  
        },  
    });  
  
    // Default task.  
    grunt.registerTask('default', [  
        // ...  
    ]);  
  
};
```

## grunt less

Va compiler le contenu de /less vers /css

# Grunt - Plugins grunt-autoprefixer

---

## ► grunt-autoprefixer :

npm install grunt-autoprefixer --save-dev

Rajoute automatiquement les préfixes -moz, -webkit, -o, -ms en fonction des versions minimales des navigateurs à supporter

```
module.exports = function(grunt) {  
  
  // ...  
  
  grunt.initConfig({  
    // ...  
    autoprefixer: {  
      options: {  
        browsers: ['last 2 versions', 'ie 8', 'ie 9']  
      },  
      dev: {  
        files: [{  
          expand: true,  
          cwd: 'css/',  
          src: '{,*/}*.css',  
          dest: 'css/'  
        }]  
      },  
    },  
  });  
  
  // Default task.  
  grunt.registerTask('default', [  
    // ...  
  ]);  
  
};
```

## grunt autoprefixer

Modifie les fichiers css en ajoutant les préfixes CSS

# Grunt - Plugins grunt-contrib-watch

---

## ► grunt-contrib-watch :

npm install grunt-contrib-watch --save-dev

Surveille les modifications sur des fichiers, exécute des taches en cas de changement

```
module.exports = function(grunt) {  
  
    // ...  
  
    grunt.initConfig({  
        // ...  
        watch: {  
            less: {  
                files: ['less/**/*.less'],  
                tasks: ['less:dev', 'autoprefixer:dev']  
            }  
        },  
    });  
  
    // Default task.  
    grunt.registerTask('default', [  
        // ...  
    ]);  
};
```

## grunt watch

Surveille les fichiers less, compile les fichiers CSS en ajoutant les préfixes

# Grunt - Plugins concat, uglify, cssmin

---

- ▶ **grunt-contrib-concat :**

`npm install grunt-contrib-concat --save-dev`

Concatène plusieurs fichiers en un. Utile pour optimiser les temps de chargement CSS/JS

- ▶ **grunt-contrib-uglify :**

`npm install grunt-contrib-uglify --save-dev`

Comprime les fichiers JS

- ▶ **grunt-contrib-cssmin :**

`npm install grunt-contrib-cssmin --save-dev`

Comprime les fichiers CSS

# Grunt - Plugins copy, clean

---

- ▶ **grunt-contrib-copy :**  
npm install grunt-contrib-copy --save-dev  
Copie des fichiers
- ▶ **grunt-contrib-clean :**  
npm install grunt-contrib-clean --save-dev  
Supprime des fichiers

# Grunt - Plugins grunt-usemin

## ► grunt-usemin:

npm install grunt-usemin --save-dev

Génère une configuration pour concat, uglify, cssmin à partir d'un fichier HTML

### index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title></title>

  <!-- build:css css/app.css -->
  <link rel="stylesheet" href="css/body.css">
  <link rel="stylesheet" href="css/button.css">
  <!-- endbuild -->
</head>
<body>

  <!-- build:js js/app.js -->
  <script src="js/create-button.js"></script>
  <script src="js/button-listener.js"></script>
  <!-- endbuild -->
</body>
</html>
```

### Gruntfile.js

```
/*global module, require*/
module.exports = function(grunt) {
  'use strict';

  // ...

  grunt.initConfig({
    // ...
    useminPrepare: {
      html: 'index.html'
    },

    usemin: {
      html: ['dist/{,*/}*.html'],
      css: ['dist/{,*/}*.css'],
      js: ['dist/{,*/}*.js'],
    },

  });

  // Default task.
  grunt.registerTask('default', [
    // ...
  ]);
};
```

# Grunt - Plugins grunt-usemin

## ► grunt-usemin:

npm install grunt-usemin --save-dev

Génère une configuration pour concat, uglify, cssmin à partir d'un fichier HTML

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title></title>

  <!-- build:css css/app.css -->
  <link rel="stylesheet" href="css/body.css">
  <link rel="stylesheet" href="css/button.css">
  <!-- endbuild -->
</head>
<body>

  <!-- build:js js/app.js -->
  <script src="js/create-button.js"></script>
  <script src="js/button-listener.js"></script>
  <!-- endbuild -->
</body>
</html>
```

Gruntfile.js

```
/*global module, require*/
module.exports = function(grunt) {
  'use strict';

  // ...

  grunt.initConfig({
    // ...
    useminPrepare: {
      html: 'index.html'
    },

    usemin: {
      html: ['dist/{,*/}*.html'],
      css: ['dist/{,*/}*.css'],
      js: ['dist/{,*/}*.js'],
    },

  });

  // Default task.
  grunt.registerTask('default', [
    // ...
  ]);
};
```



# Grunt - Plugins grunt-usemin

## config générée

```
{
  "concat": {
    "generated": {
      "files": [{
        "dest": ".tmp/concat/css/app.css",
        "src": ["css/body.css", "css/button.css"]
      }, {
        "dest": ".tmp/concat/js/app.js",
        "src": ["js/create-button.js", "js/button-
listener.js"]
      }]
    },
    "uglify": {
      "generated": {
        "files": [{
          "dest": "dist/js/app.js",
          "src": [".tmp/concat/js/app.js"]
        }]
      }
    },
    "cssmin": {
      "generated": {
        "files": [{
          "dest": "dist/css/app.css",
          "src": [".tmp/concat/css/app.css"]
        }]
      }
    }
  }
}
```

## index.html généré

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title></title>

  <link rel="stylesheet" href="css/app.css">
</head>
<body>

<script src="js/app.js"></script>
</body>
</html>
```

## app.css généré

```
body{background:beige}button{width:50px;height:50px}
```

## app.js généré

```
!function(){"use strict";var
a=document.createElement("button");a.innerHTML=0,a.id="monBouton",document.body.appendChild(a)}(),!function()
{"use strict";var
a=document.querySelector("#monBouton");a.addEventListener("click",function(){this.innerHTML++})}();
```

# Grunt - Autres plugins

---

- ▶ **contrib-connect :**  
serveur web
- ▶ **karma :**  
lancer des tests
- ▶ **concurrent :**  
exécuter des tâches en parallèle
- ▶ **sass :**  
compile des fichiers SASS en CSS
- ▶ **contrib-imagemin :**  
compresser des images
- ▶ **contrib-htmlmin :**  
minifier le HTML
- ▶ **newer :**  
ne lancer les tâches que sur les nouveaux fichiers
- ▶ **rev :**  
génère un nom de fichier avec hash pour le cache (avec usemin)
- ▶ **contrib-jshint, jscs :**  
vérifie les conventions sur les fichiers JS
- ▶ **google-cdn :**  
remplace les fichiers locaux par des CDN
- ▶ **spritesmith :**  
génère des fichiers Sprite CSS

# Grunt - Assistant

---

- ▶ Grunt Init

Assistant de création de projet grunt

- ▶ Installation

npm install -g grunt-init

- ▶ Création du projet

grunt-init gruntfile

```
Please answer the following:
[?] Is the DOM involved in ANY way? (Y/n) Y
[?] Will files be concatenated or minified? (Y/n) Y
[?] Will you have a package.json file? (Y/n) Y
[?] Do you need to make any changes to the above before continuing? (y/N) N

Writing Gruntfile.js...OK
Writing package.json...OK

Initialized from template "gruntfile".
```

- ▶ Créer son propre assistant/plugin :

<https://github.com/gruntjs/grunt-init-gruntplugin>

# Grunt - Autres

---

## ► Gulp

Equivalent de grunt, repose sur les streams Node.js (utilise la RAM plutôt que les fichiers).

Deviens très populaire, 1645 plugins contre 4403 pour grunt (juillet 2015)

## ► Broccoli

484 plugins

## ► Brunch

262 plugins

## ► Prepros / CodeKit

<https://prepros.io>

<https://incident57.com/codekit/>

## gulpfile.js

```
var gulp = require('gulp');
var uglify = require('gulp-uglify');

gulp.task('scripts', function() {
  // Minify and copy all JavaScript (except vendor
  // scripts)
  gulp.src(['client/js/**/*.js', '!client/js/vendor/
  **'])
    .pipe(uglify())
    .pipe(gulp.dest('build/js'));

  // Copy vendor files
  gulp.src('client/js/vendor/**')
    .pipe(gulp.dest('build/js/vendor'));
});

// The default task (called when you run `gulp`)
gulp.task('default', function() {
  gulp.run('scripts');

  // Watch files and run tasks if they change
  gulp.watch('client/js/**', function(event) {
    gulp.run('scripts');
  });
});
```

# Tests Automatisés

---

# Tests automatisés - Introduction

---

- ▶ Avec les tests automatisés, les scénarios de test sont codés et peuvent être rejoués régulièrement.
  
- ▶ 3 types de test :
  - Test unitaire  
Permet de tester les briques d'une application (classes)
  
  - Test d'intégration  
Teste que les briques fonctionnent correctement ensemble
  
  - Test fonctionnel  
Vérifie l'application du point de vue du client

# Tests automatisés - Karma



- ▶ **Lanceur de test**

Permet de lancer vos tests simultanément dans Chrome, Firefox, Internet Explorer...

- ▶ **Installation**

npm install -g karma-cli

npm install karma --save-dev

- ▶ **Configuration du projet**

karma init

- ▶ **Lancement des tests**

karma start

```
Air-de-Romain:Jasmine romain$ karma init

Which testing framework do you want to use ?
Press tab to list possible options. Enter to move to the next question.
> jasmine

Do you want to use Require.js ?
This will add Require.js plugin.
Press tab to list possible options. Enter to move to the next question.
> no

Do you want to capture any browsers automatically ?
Press tab to list possible options. Enter empty string to move to the next question.
> Chrome
> Safari
>

What is the location of your source and test files ?
You can use glob patterns, eg. "js/*.js" or "test/**/*.js".
Enter empty string to move to the next question.
> 
```

```
Air-de-Romain:Jasmine romain$ karma start
02 09 2015 21:30:11.510:INFO [karma]: Karma v0.13.9 server started at http://localhost:9876/
02 09 2015 21:30:11.518:INFO [launcher]: Starting browser Chrome
02 09 2015 21:30:11.526:INFO [launcher]: Starting browser Safari
02 09 2015 21:30:12.723:INFO [Safari 8.0.7 (Mac OS X 10.10.4)]: Connected on socket HE38slHTBKXL5t5yAAAA with id 54715269
Safari 8.0.7 (Mac OS X 10.10.4): Executed 1 of 1 SUCCESS (0.038 secs / 0.003 secs)
Safari 8.0.7 (Mac OS X 10.10.4): Executed 1 of 1 SUCCESS (0.038 secs / 0.003 secs)
Chrome 45.0.2454 (Mac OS X 10.10.4): Executed 1 of 1 SUCCESS (0.04 secs / 0.008 secs)
TOTAL: 2 SUCCESS
```

# Tests automatisés - QUnit

---

- ▶ Créé en 2008 par les développeurs de jQuery
- ▶ Type xUnit (JUnit, PHPUnit...) : basés sur des assertions
- ▶ Plutôt destiné à du code client
- ▶ Installation
  - npm install --save-dev qunitjs
  - bower install --save-dev qunit
- ▶ Lancement des tests
  - Ouverture du fichier .html
  - grunt-contrib-qunit
  - karma-qunit

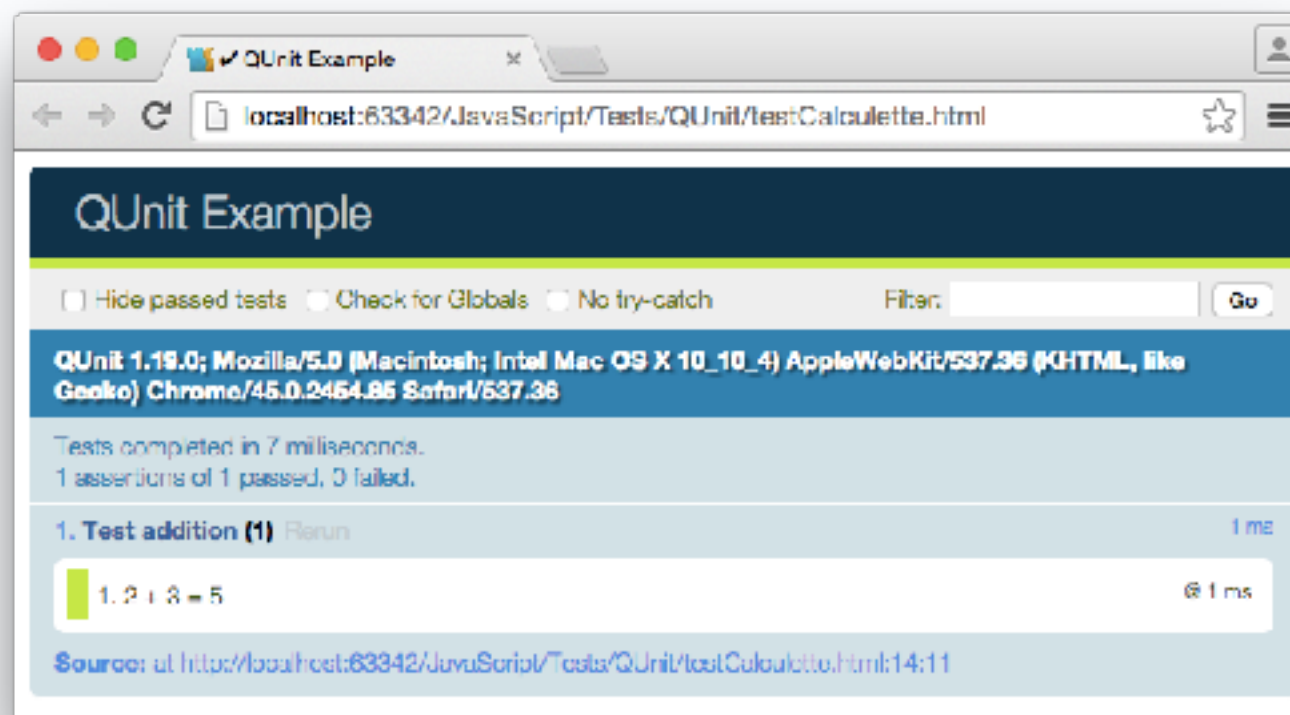




# Tests automatisés - QUnit

```
<!-- runner.html -->
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>QUnit Example</title>
  <link rel="stylesheet" href="node_modules/qunitjs/qunit/qunit.css">
</head>
<body>
<div id="qunit"></div>
<div id="qunit-fixture"></div>
<script src="calcullette.js"></script>
<script src="node_modules/qunitjs/qunit/qunit.js"></script>
<script src="calcullette-test.js"></script>
</body>
</html>
```

```
// calcullette-test.js
QUnit.test("Test addition", function(assert) {
  assert.equal(calcullette.ajouter(2, 3), 5, "2 + 3 = 5");
});
```



# Tests automatisés - Jasmine

---



- ▶ Créé en 2010
- ▶ Type BDD (Behavior-Driven Development)
- ▶ Fonctionne pour le browser ou node.js
- ▶ Installation et lancement des tests (node)  
`npm install -g jasmine`  
`jasmine init`  
`jasmine`
- ▶ Installation et lancement des tests (browser)  
`npm install --save-dev jasmine-core`  
`SpecRunner.html`  
`karma`

# Tests automatisés - Jasmine

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Jasmine Spec Runner v2.3.4</title>

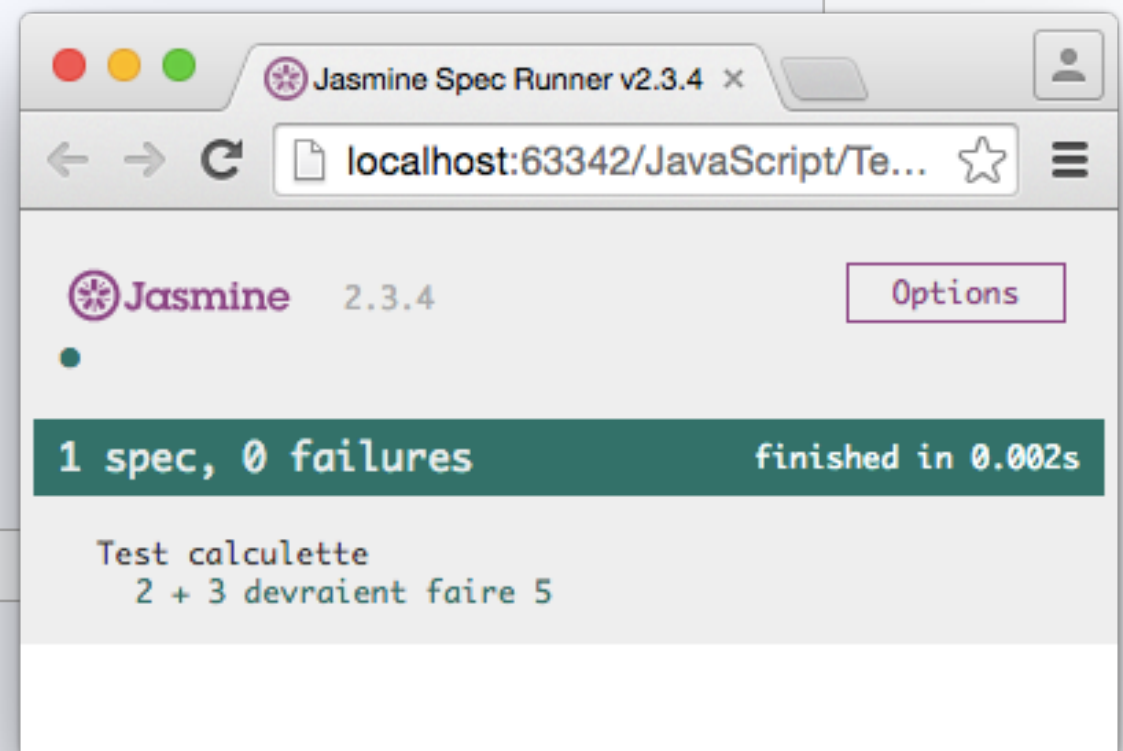
  <link rel="shortcut icon" type="image/png" href="node_modules/jasmine-core/images/
jasmine_favicon.png">
  <link rel="stylesheet" href="node_modules/jasmine-core/lib/jasmine-core/jasmine.css">

  <script src="node_modules/jasmine-core/lib/jasmine-core/jasmine.js"></script>
  <script src="node_modules/jasmine-core/lib/jasmine-core/jasmine-html.js"></script>
  <script src="node_modules/jasmine-core/lib/jasmine-core/boot.js"></script>

  <!-- include source files here... -->
  <script src="calculette.js"></script>

  <!-- include spec files here... -->
  <script src="spec/CalculetteSpec.js"></script>
</head>
<body>
</body>
</html>
```

```
describe("Test calculette", function() {
  it("2 + 3 devraient faire 5", function() {
    expect(calculette.ajouter(2, 3)).toEqual(5);
  });
});
```



# Tests automatisés - Mocha

---



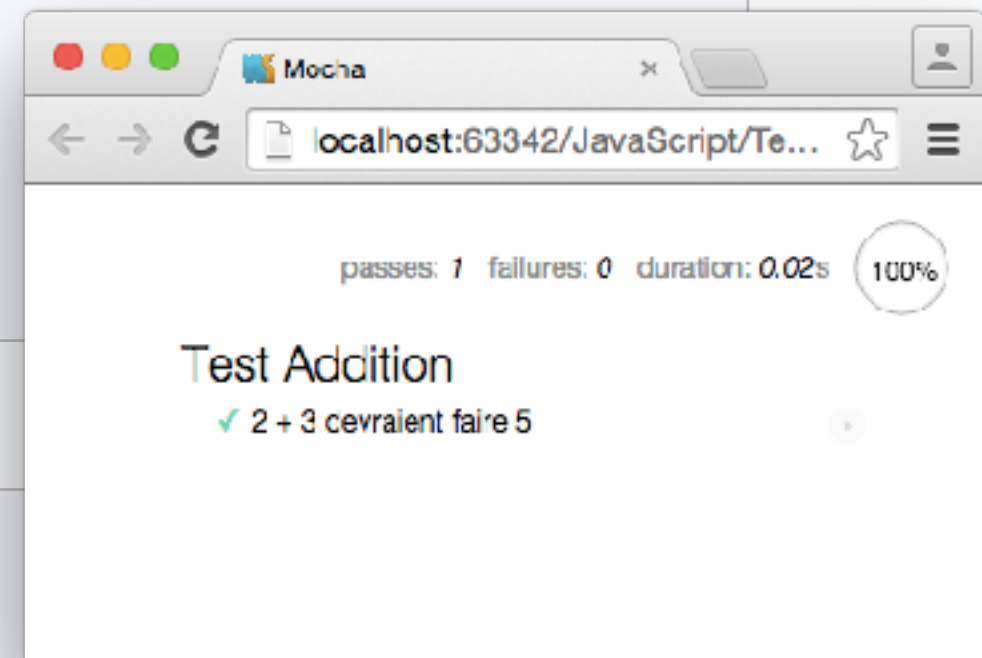
- ▶ Créé en 2011
- ▶ Type assert ou BDD (le framework est flexible)
- ▶ Fonctionne pour le browser ou node.js
- ▶ Installation et lancement des tests (node)  
`npm install -g mocha`  
`mocha`
- ▶ Installation et lancement des tests (browser)  
`npm install -g mocha`  
`mocha init`  
`npm install chai`  
`runner.html`  
`karma`

# Tests automatisés - Mocha

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mocha</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="mocha.css" />
  </head>
  <body>
    <div id="mocha"></div>
    <script src="mocha.js"></script>
    <script src="node_modules/chai/chai.js"></script>
    <script>mocha.setup('bdd');</script>
    <script src="src/calcullette.js"></script>
    <script src="test/calcullette-test.js"></script>
    <script>
      mocha.run();
    </script>
  </body>
</html>
```

```
var assert = chai.assert;

describe('Test Addition', function() {
  it('2 + 3 devraient faire 5', function () {
    assert.equal(5, calcullette.ajouter(2, 3));
  });
});
```



# Scaffolding

---

# Scaffolding - Yeoman

---

- ▶ **Scaffolding**  
Permet d'initialiser des projets déjà structurés
- ▶ **Yeoman**  
<http://yeoman.io>
- ▶ **Installation**  
npm install -g yo
- ▶ **Générateurs**  
2132 (juillet 2015)  
<http://yeoman.io/generators/>
- ▶ **Exemple : webapp**  
npm install -g generator-webapp



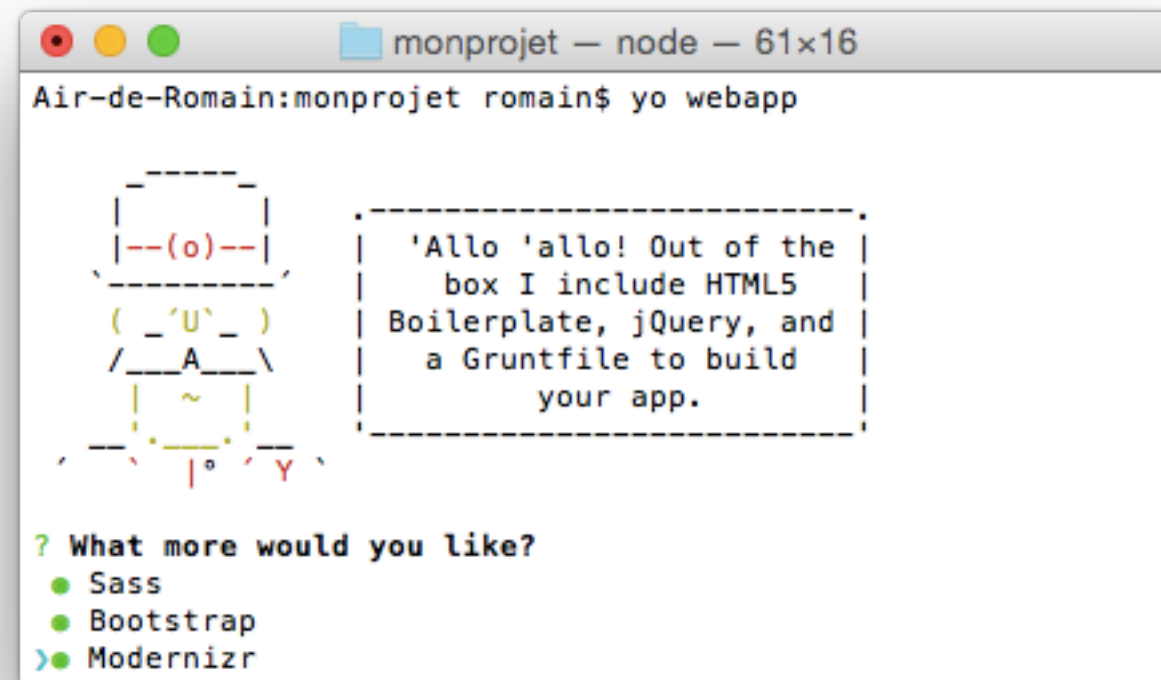
# Scaffolding - Yeoman

---

## ► Création d'un projet

`mkdir mon-projet && cd mon-projet`

`yo webapp`



```
monprojet — node — 61x16
Air-de-Romain:monprojet romain$ yo webapp

  _ _ _
 |--(o)--|
 |  _U_  |
 |  A  ~  |
 |  _ _  |
 |  _ _  |
 |  _ _  |

'Allo 'allo! Out of the
  box I include HTML5
  Boilerplate, jQuery, and
  a Gruntfile to build
  your app.

? What more would you like?
  ● Sass
  ● Bootstrap
  >● Modernizr
```