



Conseil

Technologie

Innovation

JavaScript : un langage de script devenu industriel grâce aux frameworks

Romain Bohdanowicz

Twitter @bioub

<http://formation.tech/>

JavaScript : Genèse

- 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 : Helloworld 1996

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

      spanElt.firstChild.nodeValue = inputElt.value;
    }
  </script>
</head>
<body>
<div>
  Prénom :
  <input name="prenom" onkeyup="helloworld();">
</div>
<p>
  Bonjour <span> </span>
</p>
</body>
</html>
```

Prénom :

Bonjour Rom

JavaScript : Modules

- En JavaScript côté navigateur, chaque variable/fonction définie en dehors d'une fonction est globale, y compris dans les fichiers externes
- Pour isoler les variables/fonctions d'un fichier il faut les déclarer dans une fonction expression (fonction anonyme), c'est ce qu'on appelle un module
- Plusieurs conventions de modules existent :
Immediately-invoked function expression (IIFE), YUI, CommonJS, AMD, UMD, SystemJS, ECMAScript 6, AngularJS...
- Pour simplifier le chargement de ces modules on peut également utiliser un Loader :
RequireJS, curl.js, Node.js, SystemJS, webpack...

JavaScript : Helloworld 2006

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Helloworld</title>
  <script>
    (function() {
      'use strict';

      window.onload = function() {
        var inputElt = document.getElementById('saisie');
        var spanElt = document.getElementById('boite');

        inputElt.onkeyup = function () {
          spanElt.textContent = inputElt.value;
        };
      }();
    })();
  </script>
</head>
<body>
<div>
  Prénom : <input id="saisie">
</div>
<p>
  Bonjour <span id="boite"></span>
</p>
</body>
</html>

```

Prénom :

Bonjour Rom

JavaScript : Un développement multi-plateforme

- Navigateurs :
Chrome, Firefox, Internet Explorer, Safari, Opera...
- Mobiles :
Navigateurs mobiles, PhoneGap/Cordova (appli natives développées en JS)
- Serveur/Console :
Node.js, Meteor...
- Application de bureau :
Electron, NW.js
- Autres :
Extensions de navigateur, Montres connectées...

JavaScript : Première génération de frameworks

- A partir de 2005 :
Prototype, Dojo, jQuery, jQuery UI, Mootools, ExtJS, YUI...
- Objectifs :
 - Simplifier la manipulation du DOM
 - Encapsuler les problèmes de compatibilité inter-navigateur
 - Faciliter les échanges avec un serveur (AJAX)
 - Proposer des composants d'UI complexes (menus déroulants, alerts, fenêtres...)

JavaScript : Helloworld jQuery

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Helloworld</title>
</head>
<body>
<div>
  Prénom : <input class="firstName">
</div>
<p>
  Bonjour <span class="result"></span>
</p>
<script src="http://code.jquery.com/jquery-1.12.1.js"></script>
<script>
  (function() {
    'use strict';

    var $inputElt = $('.firstName');
    var $spanElt = $('.result');

    $inputElt.on('input', function() {
      $spanElt.html($inputElt.val());
    });
  }());
</script>
</body>
</html>

```

Prénom :

Bonjour Rom

JavaScript : HTML5

- Evolution de la norme HTML
 - Normer l'existant
 - Nouvelles balises issues de demandes utilisateurs
 - Normer la correction d'erreur
- Terme marketing regroupant :
 - Norme HTML
 - Nouveautés CSS3
 - JavaScript
 - Nouveaux APIs du Navigateur (Geolocation, Storage, Workers, WebGL...)

JavaScript : Helloworld 2016

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

    var inputElt = document.body.querySelector('.firstName');
    var spanElt = document.body.querySelector('.result');

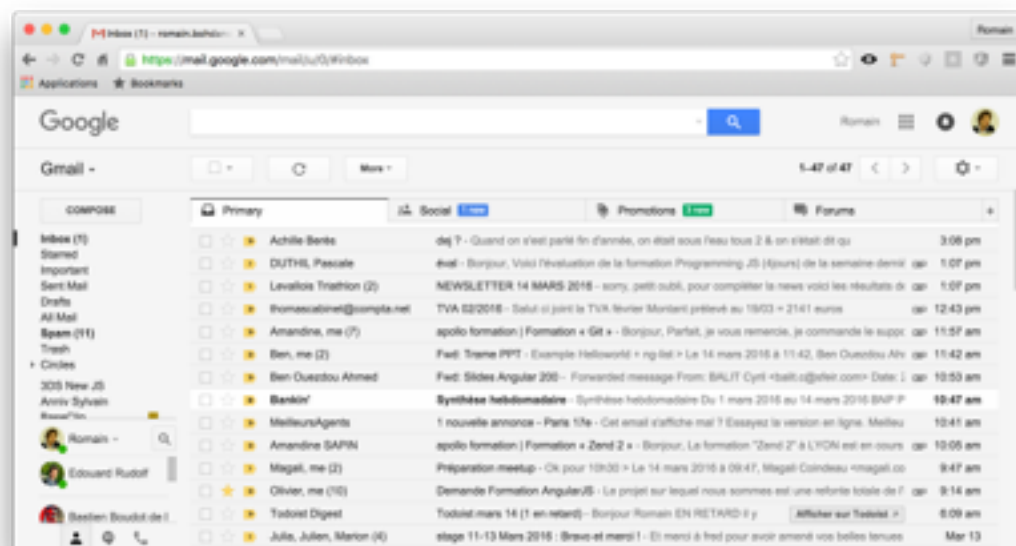
    inputElt.addEventListener('input', function() {
      spanElt.innerHTML = inputElt.value;
    });
  })();
</script>
</body>
</html>
```

Prénom :

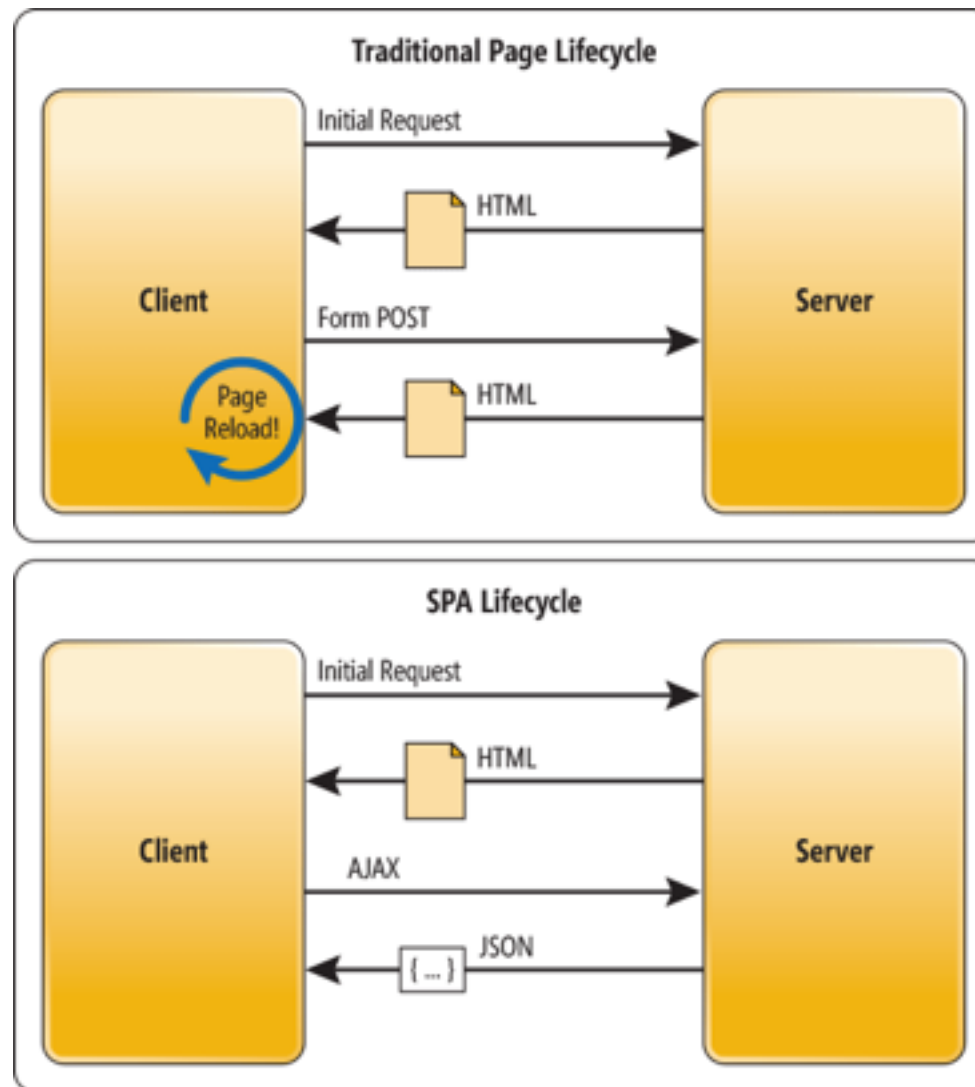
Bonjour Rom

JavaScript : Single Page Application (SPA)

- Site web « classique » :
Plusieurs pages servies par le serveur
- Single Page Application :
Une seule page servie par le serveur, plusieurs vues possibles et accès au données du serveur via un API REST

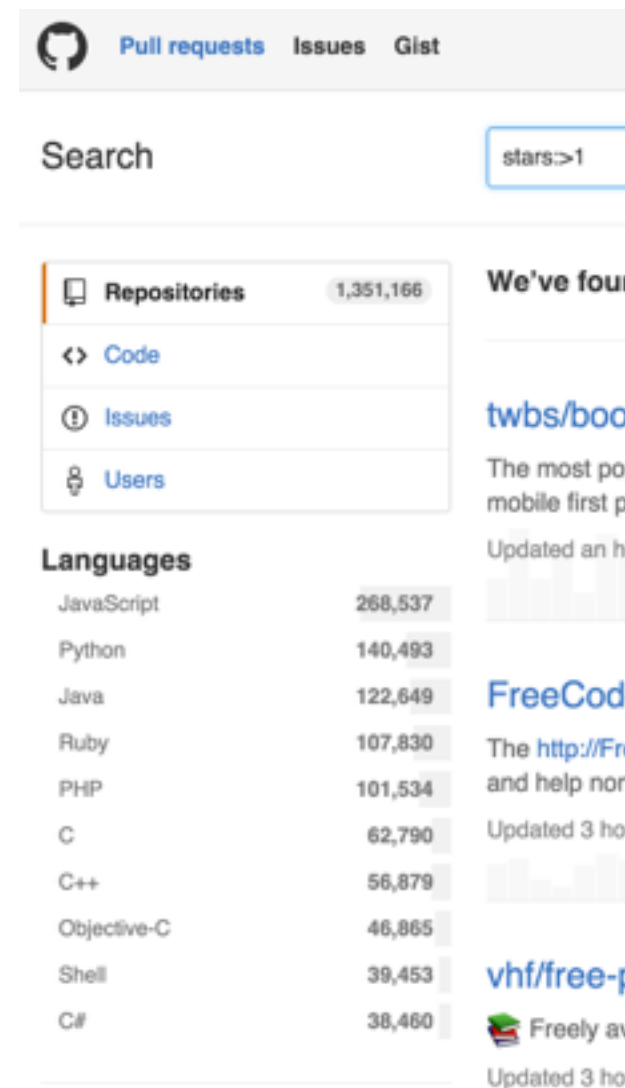


JavaScript : Single Page Application (SPA)



JavaScript : Gestionnaires de dépendances

- JavaScript : Langage le plus populaire sur GitHub
- Projets très fractionnés
- Nécessité d'un gestionnaire pour télécharger nos dépendances dans les versions demandées (équivalent à Maven en Java)
- Gestionnaires de dépendances Javascript : npm, bower, jspm...
- Ex : `bower install angular --save`



JavaScript : 2nd génération de frameworks

- A partir de 2009 :
Backbone.js, AngularJS, Ember.js, Knockout
- Objectifs :
 - Architecturer les applications (MVC, MVVM...)
 - Gestion des URL (Routers)
 - Simplifier l'UI (Templates, Data Binding)

JavaScript : Helloworld AngularJS

Prénom :

Bonjour Rom

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Helloworld</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.0/
angular.min.js"></script>
</head>
<body ng-app>
  <div>
    Prénom : <input ng-model="prenom">
  </div>
  <p>
    Bonjour <span ng-bind="prenom"></span>
  </p>
</body>
</html>
```

JavaScript : Objet

- Pas de classes en JavaScript
- Objets prédéfinis
- Object Literal / JSON
- Fonctions constructeurs / Programmation orientée Prototype

JavaScript : Helloworld Objet

```
// helloworld-objet.js
(function(global) {
  'use strict';

  function Helloworld(source, dest) {
    if (arguments.length < 2) {
      throw new Error('Not enough arguments');
    }

    this.inputElt = document.body.querySelector(source);
    this.spanElt = document.body.querySelector(dest);
  }

  Helloworld.prototype.start = function() {
    var that = this;
    this.inputElt.addEventListener('input', function() {
      that.spanElt.innerHTML = that.inputElt.value;
    });
  };

  global.Helloworld = Helloworld;
})(this);
```

JavaScript : Helloworld Objet

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Helloworld</title>
</head>
<body>
<div>
  Prénom : <input class="firstName">
</div>
<p>
  Bonjour <span class="result"></span>
</p>
<script src="helloworld-objet.js"></script>
<script>
  (function() {
    'use strict';

    var hello = new Helloworld('.firstName', '.result');
    hello.start();
  }());
</script>
</body>
</html>
```

Prénom :

Bonjour Rom

JavaScript : ECMAScript 2015 / ECMAScript 6

- Evolution du langage en 2015
- Peu de navigateurs supportent la norme en 2016
<http://kangax.github.io/compat-table/es6/>
- Du code ECMAScript 6 peut se transformer en ECMAScript 5 avec un transpileur (Traceur, Babel, TypeScript...)
- Nouveautés : <http://es6-features.org/>
 - Classes
 - Modules
 - Arrow functions (lambdas)
 - Constantes
 - Portée de bloc
 - ...

JavaScript : Helloworld ECMAScript 6

```
// helloworld-es6.js
class Helloworld {
  constructor(source, dest) {
    if (arguments.length < 2) {
      throw new Error('Not enough arguments');
    }

    this.inputElt = document.body.querySelector(source);
    this.spanElt = document.body.querySelector(dest);
  }

  start() {
    this.inputElt.addEventListener('input', () => {
      this.spanElt.innerHTML = this.inputElt.value;
    });
  }
}

export default Helloworld;
```

```
// main.js
import Helloworld from "helloworld-es6";

let hello = new Helloworld('.firstName', '.result');
hello.start();
```

JavaScript : Helloworld ECMAScript 6

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Helloworld</title>
</head>
<body>
<div>
  Prénom : <input class="firstName">
</div>
<p>
  Bonjour <span class="result"></span>
</p>
<script src="jspm_packages/system.js"></script>
<script src="config.js"></script>
<script>
  System.import('main.js');
</script>
</body>
</html>
```

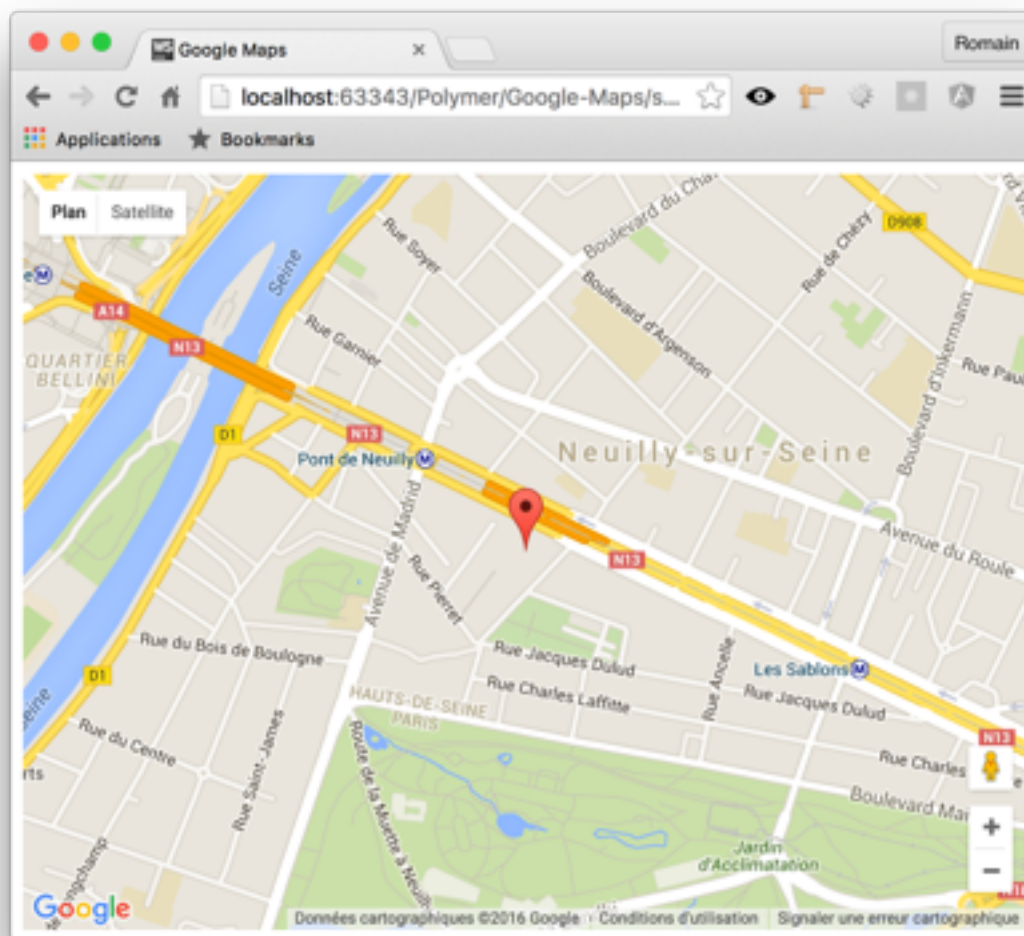
Prénom :

Bonjour Rom

JavaScript : Vers un web orienté composants

- A partir de 2013 :
Polymer, React, Angular 2
- Objectifs :
 - Créer ses propres balises masquant la complexité du JavaScript ou du CSS
 - Réutiliser ses composants d'UI plus facilement

JavaScript : Polymer



JavaScript : Sans Polymer

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Google Maps</title>
  <style>
    #map {
      width: 640px;
      height: 480px;
    }
  </style>
</head>
<body>
  <div id="map"></div>
  <script src="https://maps.googleapis.com/maps/api/js?callback=initMap" async defer></script>
  <script>
    (function(global) {
      'use strict';

      global.initMap = function() {
        var mapContainer = document.querySelector('#map');

        var coords = {lat: 48.88304609999999, lng: 2.2633637};

        var map = new google.maps.Map(mapContainer, {center: coords, zoom: 15});

        var marker = new google.maps.Marker({position: coords, map: map});

      };

    })(this));
  </script>
</body>
</html>

```


JavaScript : Avec Polymer

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Google Maps</title>
  <script src="bower_components/webcomponentsjs/webcomponents-lite.min.js"></script>
  <link rel="import" href="bower_components/google-map/google-map.html">
  <style>
    google-map {
      width: 640px;
      height: 480px;
    }
  </style>
</head>
<body>
  <google-map latitude="48.88304609999999" longitude="2.2633637" zoom="15">
    <google-map-marker latitude="48.88304609999999" longitude="2.2633637"></google-map-marker>
  </google-map>
</body>
</html>
```

JavaScript : Helloworld Polymer

```
<!-- components/bioub-helloworld.html -->
<link rel="import" href="../../bower_components/polymer/polymer.html">

<dom-module id="bioub-helloworld">
  <template>
    <p>Bonjour <span>{{qui}}</span></p>
  </template>
</dom-module>

<script>
  Polymer({
    is: 'bioub-helloworld',
    properties: {
      qui: {
        type: String,
        value: ''
      }
    }
  });
</script>
```

JavaScript : Helloworld Polymer

```
<!-- components/bioub-app.html -->
<link rel="import" href="../../bower_components/polymer/polymer.html">
<link rel="import" href="bioub-helloworld.html">

<dom-module id="bioub-helloapp">
  <template>
    <div>Prénom : <input value="{{prenom::input}}"></div>
    <bioub-helloworld qui="{{prenom}}"></bioub-helloworld>
  </template>
</dom-module>

<script>
  Polymer({
    is: 'bioub-helloapp'
  });
</script>
```

JavaScript : Helloworld Polymer

Prénom :

Bonjour Rom

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Helloworld</title>
  <script src="bower_components/webcomponentsjs/webcomponents-lite.min.js"></script>
  <link rel="import" href="components/bioub-helloapp.html">
</head>
<body>
  <bioub-helloapp></bioub-helloapp>
</body>
</html>
```