



formation.tech

Modules/Loaders/Bundlers JavaScript



formation.tech

Modules JavaScript

Modules JavaScript - Introduction



- JavaScript à sa conception
 - Objectif : créer des interactions côté client, après chargement de la page
 - Exemples de l'époque :
 - Menu en rollover (image ou couleur de fond qui change au survol)
 - Validation de formulaire
- JavaScript aujourd'hui
 - Applications front-end, back-end, en ligne de commande, de bureau, mobiles...
 - Applications pouvant contenir plusieurs centaines de milliers de lignes de codes (Front-end de Facebook > 1 000 000 LOC)
 - Il faut faciliter le travail collaboratif, en plusieurs fichiers et en limitant les risques de conflit

Modules JavaScript - Introduction

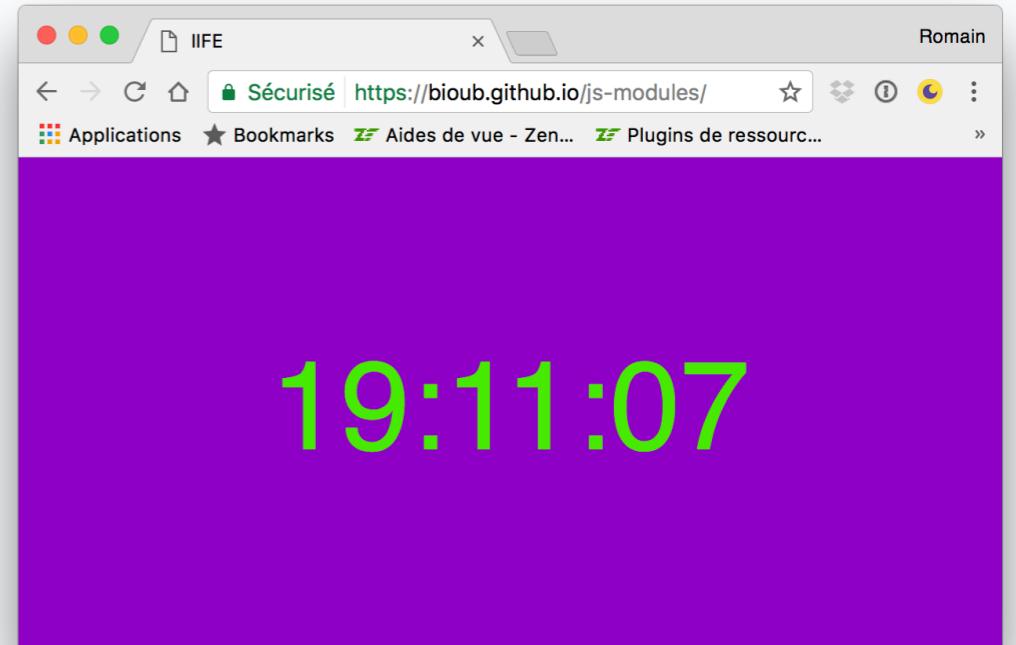
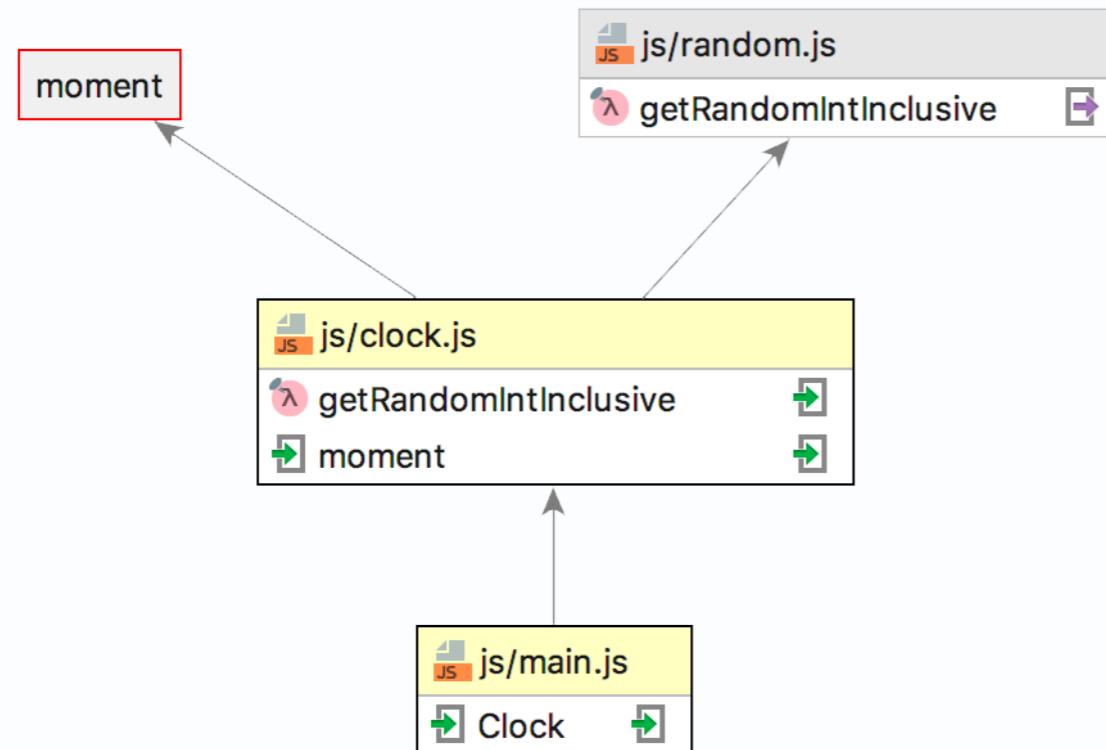


- Objectifs d'un module JavaScript
 - Créer une portée au niveau du fichier
 - Permettre l'export et l'import d'identifiants (variables, fonctions...) entre ces fichiers qui auront désormais leur propre portée
- Principaux systèmes existants
 - IIFE / Function Wrapper
 - CommonJS (fonction synchrone)
 - AMD
 - UMD
 - ES6/ES2015/ESM (statiques mots clés import / export)
 - ES2020 : import() (fonction asynchrone)
 - ES2022 : import.meta
 - ES2025 : import attributes (with)

Modules JavaScript - Introduction



- Exemple utilisé pour la suite



- Le point d'entrée de l'application est le fichier `main.js`, qui dépend de `Clock` défini dans le fichiers `clock.js`, qui dépend lui même de `getRandomIntInclusive` du fichier `random.js` et `moment` définit dans le projet Open Source `moment.js`
- Exemples : <https://github.com/bioub/js-modules>
- Démo : <https://bioub.github.io/js-modules/>

Modules JavaScript - IIFE



- Immediately-invoked function expression (IIFE)

```
// random.js
(function () {
  'use strict';

  var getRandom = function() {
    return Math.random();
  };

  var getRandomIntInclusive = function(min, max) {
    min = Math.ceil(min);
    max = Math.floor(max);
    return Math.floor(getRandom() * (max - min + 1)) + min;
  };

  globalThis.getRandomIntInclusive = getRandomIntInclusive;
}());
```

- Une function expression appelée immédiatement
 - Limite la portée des identifiants (getRandom, getRandomIntInclusive)
 - Permet de renommer localement des dépendances (this → global)



Modules JavaScript - IIFE

▸ Import / Export en IIFE

```
// clock.js
(function(global, moment, random) {
  'use strict';

  var Clock = function(parentElt) {
    this.parentElt = parentElt;
  };

  Clock.prototype.update = function() {
    var n = random(0, 255);
    document.body.style.backgroundColor = 'rgb('+n+', '+n+', '+n+')';
    this.parentElt.innerHTML = moment().format('HH:mm:ss');
  };

  Clock.prototype.start = function() {
    this.update();
    setInterval(this.update.bind(this), 1000);
  };
  global.Clock = Clock;
})(this, moment, getRandomIntInclusive);
```

- Pour importer on utilise des variables globales, éventuellement en paramètres d'entrée de la fonction pour pouvoir les renommer localement (getRandomIntInclusion → random)
- Pour exporter on crée des variables globales en étendant l'objet global

Modules JavaScript - IIFE



- ▶ Limiter les risques de conflits en IIFE

Exemple : API Maps de Google

```
<!DOCTYPE html>
<html>
<head></head>
<body>
<div id="map"></div>
<script>
  function initMap() {
    var uluru = {lat: -25.363, lng: 131.044};
    var map = new google.maps.Map(document.getElementById('map'), {
      zoom: 4,
      center: uluru
    });
    var marker = new google.maps.Marker({
      position: uluru,
      map: map
    });
  }
</script>
<script async defer src="https://maps.googleapis.com/maps/api/js?callback=initMap"></script>
</body>
</html>
```

Modules JavaScript - IIFE



- ▶ Limiter les risques de conflits en IIFE

```
(function(global, google) {
    'use strict';

    var GMap = function (parentElt, options) {
        // ...
    };

    var GMarker = function (parentElt, options) {
        // ...
    };

    global.google = google || {};
    google.maps = google.maps || {};
    google.maps.Map = GMap;
    google.maps.Marker = GMarker;

})(global, google));
```

- Pour limiter les conflits de nom on simule des namespaces.
- Il ne faut créer qu'une seule variable globale du nom de la société, nom du projet... (google dans l'exemple)



Modules JavaScript - IIFE

- ▶ Utilisation dans le Navigateur

```
<!DOCTYPE html>
<html lang="en">
<head></head>
<body>
  <div class="clock"></div>
  <script src="node_modules/moment/moment.js"></script>
  <script src="js/random.js"></script>
  <script src="js/clock.js"></script>
  <script src="js/main.js"></script>
</body>
</html>
```

- Avec les modules IIFE c'est au développeur de créer les balises script
- Les imports/exports se faisant via des variables globales, il faut maintenir ces balises dans l'ordre des dépendances
- ▶ Utilisation dans Node.js
 - Node.js incluant son propre système de module, il n'est pas recommandé d'utiliser des modules IIFE

Modules JavaScript - CommonJS



- ▶ **CommonJS (parfois CJS)**

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://wiki.commonjs.org/wiki/Modules/1.1.1>

- ▶ **Création d'un module**

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

- ▶ **Utilisation**

```
// main.js
var calc = require('./calculette');

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



Modules JavaScript - CommonJS

▸ Export

```
// (function (exports, require, module, __filename, __dirname) {
'use strict';

var getRandom = function() {
    return Math.random();
};

exports.getRandomIntInclusive = function(min, max) {
    min = Math.ceil(min);
    max = Math.floor(max);
    return Math.floor(getRandom() * (max - min + 1)) + min;
};
// });
});
```

- Le code s'exécute dans une fonction, pas besoin de la créer (ici en commentaire)
- Cette fonction contient un paramètre exports, qui est un objet que l'on pourra récupérer à l'import (il n'y a plus qu'à l'étendre)

Modules JavaScript - CommonJS



▸ Import

```
// (function (exports, require, module, __filename, __dirname) {  
'use strict';  
  
var moment = require('moment');  
var getRandomIntInclusive = require('./random').getRandomIntInclusive;  
  
var Clock = function(parentElt) {  
  this.parentElt = parentElt;  
};  
  
// ...  
// });
```

▸ A l'import on utilise la fonction require()

- Si le fichier est local on commencera toujours par ./ ou ../
- Sinon on fait référence à une installation via npm ou bien un fichier se trouvant dans le binaire de Node (fs, http, readline...)
- La fonction require est synchrone et peut s'utiliser dans un if ou une boucle

Modules JavaScript - CommonJS



- Export (autre chose qu'un objet)

```
// (function (exports, require, module, __filename, __dirname) {  
'use strict';  
  
var Clock = function(parentElt) {  
    this.parentElt = parentElt;  
};  
  
// ...  
  
module.exports = Clock;  
  
// });
```

- Lorsque l'on exporte autre chose qu'un objet, on peut écraser exports en écrivant module.exports (ici on exporte une fonction constructeur)
- Une bonne pratique pourrait être d'importer en début de fichier et d'exporter toujours en fin de fichier (en pratique c'est rarement le cas)

Modules JavaScript - CommonJS



- Utilisation dans le Navigateur
 - Il faut passer par une bibliothèque externe :
 - Soit un bundler comme browserify (historique) ou webpack (moderne)
 - Soit un loader comme SystemJS
 - Les fichiers décrivent eux-mêmes leurs dépendances, il suffit d'indiquer un point d'entrée dans l'application (ou plusieurs)
- Utilisation dans Node.js
 - Node.js supporte par défaut les modules CommonJS

Modules JavaScript - AMD



- **Asynchronous Module Definition**

CommonJS ne permettant de charger des modules côté client sans transformation préalable. Des développeurs ont imaginé la syntaxe AMD.

- **Fonctionnement**

L'utilisation de modules AMD se fait via 2 fonctions globales : require() et define(). define() permet de définir un module et ses dépendances, require définit un point d'entrée dans l'application.



Modules JavaScript - AMD

- Export

```
// random.js
define(function() {
  'use strict';

  return {
    getRandom: function () {
      return Math.random();
    },
    // ...
    getRandomIntInclusive: function (min, max) {
      min = Math.ceil(min);
      max = Math.floor(max);
      return Math.floor(this.getRandom() * (max - min + 1)) + min;
    }
  };
});
```

- define

- La définition d'un module se passe dans le callback de la fonction define
- Pour exporter on utilise la valeur de retour
- Si plusieurs valeurs à exporter il suffit de retourner un tableau ou un objet

Modules JavaScript - AMD



▸ Import

```
// clock.js
define(['moment', './random'], function(moment, random) {
  'use strict';

  var getRandomIntInclusive = random.getRandomIntInclusive;

  var Clock = function(parentElt) {
    this.parentElt = parentElt;
  };

  // ...

  return Clock;
});
```

- Pour importer on passera un tableau en premier paramètre du define
- Le callback sera appelé avec les retours des modules importés, dans le même ordre

Modules JavaScript - AMD



- Point d'entrée

```
// main.js
require(['./clock'], function(Clock) {
  'use strict';

  var clockElt = document.querySelector('.clock');
  var clock = new Clock(clockElt);
  clock.start();
});
```

- Require

- La fonction require permet de définir un point d'entrée dans l'application
- C'est elle qui va inclure les balises scripts sur la page
- Idéalement elle ne devrait être utilisée qu'un seul fois par page
- Les scripts chargent de manière asynchrone et s'exécute lorsqu'il n'ont pas de dépendances ou que toutes leurs dépendances ont été résolues

Modules JavaScript - AMD



- Utilisation dans le Navigateur
 - Il faut passer par une bibliothèque externe :
 - Soit un loader comme curl, Require.js ou plus récemment SystemJS
 - Soit un bundler comme webpack qui va faire une transformation
- Utilisation dans Node.js
 - Non pertinent

Modules JavaScript - UMD



- Universal Module Definition (UMD)
 - Un module universel, qui est à la fois AMD, CommonJS et parfois IIFE
<https://github.com/umdjs/umd>
 - Peut s'utiliser avec tous les loaders et les bundlers existants (et sans rien si IIFE)
 - Pertinent pour les projets où l'on ne peut pas prédire la techno qui sera utilisée pour charger le module
 - Exemple de module UMD :
 - jQuery
 - Lodash → « The Lodash library exported as a UMD module. »
<https://github.com/lodash/lodash>
 - Angular → @angular/core/bundles/core.umd.js



Modules JavaScript - UMD

- Exemple de module UMD

```
(function (root, factory) {
  if (typeof exports === 'object') {
    // CommonJS
    module.exports = factory(require('moment'), require('./random'));
  } else if (typeof define === 'function' && define.amd) {
    // AMD
    define(['moment', './random'], function (moment, random) {
      return factory(moment, random);
    });
  } else {
    // IIFE (global var)
    root.Clock = factory(root.moment, root.random);
  }
})(this, function (moment, random) {
  'use strict';

  var getRandomIntInclusive = random.getRandomIntInclusive;

  var Clock = function(parentElt) {
    this.parentElt = parentElt;
  };

  // ...

  return Clock;
}));
```

Modules JavaScript - UMD



- Utilisation dans le Navigateur :
 - Sans rien si UMD inclus IIFE (pas toujours le cas)
 - Avec un bundler : browserify ou webpack
 - Avec un loader : curl, Require.js, SystemJS
- Utilisation dans Node.js
 - Avec la fonction require

Modules JavaScript - ECMAScript Modules



- ES6 introduit la notion de module au niveau du langage (ECMAScript Modules ou ESM)
- Le système est statique, les imports doivent être présent en début de fichier
- Ne permet donc pas de charger dynamiquement le contenu d'un tableau ou d'effectuer un import conditionnel comme avec CommonJS

Modules JavaScript - ECMAScript Modules



- Exporter
 - Les modules ES permettent d'exporter des valeurs multiples (plutôt que de les regrouper dans un objet)

```
export function getRandom() {
    return Math.random();
}

export function getRandomArbitrary(min, max) {
    return Math.random() * (max - min) + min;
}

export function getRandomInt(min, max) {
    min = Math.ceil(min);
    max = Math.floor(max);
    return Math.floor(Math.random() * (max - min)) + min;
}

export function getRandomIntInclusive(min, max) {
    min = Math.ceil(min);
    max = Math.floor(max);
    return Math.floor(Math.random() * (max - min + 1)) + min;
}
```

Modules JavaScript - ECMAScript Modules



- On peut également exporter en fin de fichier pour mieux lire les dépendances

```
function getRandom() {
    return Math.random();
}

function getRandomArbitrary(min, max) {
    return Math.random() * (max - min) + min;
}

function getRandomInt(min, max) {
    min = Math.ceil(min);
    max = Math.floor(max);
    return Math.floor(Math.random() * (max - min)) + min;
}

function getRandomIntInclusive(min, max) {
    min = Math.ceil(min);
    max = Math.floor(max);
    return Math.floor(Math.random() * (max - min + 1)) + min;
}

export {
    getRandom,
    getRandomArbitrary,
    getRandomInt,
    getRandomIntInclusive,
}
```

Modules JavaScript - ECMAScript Modules



- Export par défaut
 - On peut également exporter une valeur par défaut (l'import est particulier)
 - Eviter d'exporter des fonctions multiples dans un objet par défaut

```
export default class Clock {  
    constructor(parentElt) {  
        this.parentElt = parentElt;  
    }  
  
    update() {  
        const r = getRandomIntInclusive(0, 255);  
        const g = getRandomIntInclusive(0, 255);  
        const b = getRandomIntInclusive(0, 255);  
        const now = new Date();  
  
        document.body.style.backgroundColor = `rgb(${r}, ${g}, ${b})`;  
        document.body.style.color = `rgb(${255 - r}, ${255 - g}, ${255 - b})`;  
        this.parentElt.innerHTML = now.toLocaleTimeString();  
    }  
  
    start() {  
        this.update();  
        setInterval(this.update.bind(this), 1000);  
    }  
}
```



Modules JavaScript - ECMAScript Modules

- › Pour les imports par défaut il faut retirer les accolades
- › L'import peut être renommé directement

```
import Horloge from './clock';

let clockElt = document.querySelector('.clock');
let clock = new Horloge(clockElt);
clock.start();
```

Modules JavaScript - ECMAScript Modules



- On peut renommer les imports (en cas de conflit par exemple)

```
import { format } from 'date-fns/esm';
import { getRandomIntInclusive as rand } from './random';

export default class Clock {
  constructor(parentElt) {
    this.parentElt = parentElt;
  }

  update() {
    const r = rand(0, 255);
    const g = rand(0, 255);
    const b = rand(0, 255);
    const now = new Date();

    document.body.style.backgroundColor = `rgb(${r}, ${g}, ${b})`;
    document.body.style.color = `rgb(${255 - r}, ${255 - g}, ${255 - b})`;
    this.parentElt.innerHTML = now.toLocaleTimeString();
  }

  start() {
    this.update();
    setInterval(this.update.bind(this), 1000);
  }
}
```

Modules JavaScript - ECMAScript Modules



- Utilisation dans le Navigateur :
 - Sans rien dans une balise `<script type="module">`
 - Avec un bundler : webpack ou Rollup
 - Avec un loader : SystemJS
- Utilisation dans Node.js
 - Avec le flag `--experimental-modules` à partir de Node.js 10
 - Nativement depuis Node.js 13.2
 - Les fichiers doivent avoir l'extension `.mjs` ou il doit y avoir une clé "type": "module" dans le fichier `package.json`

Modules JavaScript - Dynamic Imports



- ES2020 prévoit un import dynamique (fonctions asynchrones retournant une promesse)
- webpack est déjà compatible (le code est mis dans un bundle différent, chargé au moment de l'import)

```
document.addEventListener('click', () => {
  import('./calc').then((module) => {
    console.log(module.add(1, 2));
  });
});
```

Modules JavaScript - Dynamic Imports



- Utilisation dans le Navigateur :
 - Nativement
 - Avec un bundler : webpack ou Rollup
- Utilisation dans Node.js
 - Avec webpack



formation.tech

Loaders JavaScript

Loaders JavaScript - Require.js



- **Require.js**

Bibliothèque permettant le chargement de modules AMD.

- **Plugins**

Quelques plugins existent pour charger par exemple des fichiers CSS, des fichiers de traduction, etc...

- **r.js**

Il est possible d'utiliser un builder pour créer un seul fichier de production, le chargement se ferait sinon en « escalier ».





Loaders JavaScript - Require.js

▶ Chargement en escalier

Name	Status	Type	Initiator	Size	Time	Waterfall	2.00 s	3.00 s	4.00 s	5.00 s	6.00 s	7.00 s	8.00 s	9.00 s	10.00 s	▲1
index-dev.html	200	document	Other	737 B	2.01 s											
normalize.css	200	stylesheet	index-dev.html	8.1 KB	2.42 s											
body.css	200	stylesheet	index-dev.html	473 B	2.04 s											
clock.css	200	stylesheet	index-dev.html	334 B	2.07 s											
require.js	200	script	index-dev.html	84.8 KB	3.97 s											
config.js	200	script	index-dev.html	399 B	2.10 s											
main.js	200	script	index-dev.html	463 B	2.16 s											
clock.js	200	script	require.js:1961	1023 B	2.01 s											
moment.js	200	script	require.js:1961	128 KB	4.60 s											
random.js	200	script	require.js:1961	997 B	2.05 s											

Loaders JavaScript - SystemJS



- **SystemJS**

SystemJS 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>

- **jspm**

Afin de faciliter le chargement de modules installés via des gestionnaires de dépendances, jspm permet le chargement de packages npm ou bien de

- **SystemJS Builder**

Afin de faciliter le chargement de modules installés via des gestionnaires de dépendance



formation.tech

Bundlers JavaScript

Bundlers JavaScript - 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>
```

Bundlers JavaScript - webpack



- **webpack**

webpack est un bundler universel, il sait charger n'importe quel type de modules, AMD, CommonJS, ES6

- Des loaders supplémentaires permettent de charger des fichiers CSS / LESS / SCSS / i18n / ...
- Depuis sa version 2, webpack supporte nativement les modules ES6, alors qu'il fallait utiliser un transpileur comme Babel dans la version 1.
- Le projet open-source le plus financé avec Vue.js (+ 300k\$ par an), son développeur contribue à webpack à temps plein.

Bundlers JavaScript - webpack



```
const HtmlWebpackPlugin = require('html-webpack-plugin');
const CleanWebpackPlugin = require('clean-webpack-plugin');

module.exports = {
  entry: './src/js/index.js',
  output: {
    path: __dirname + '/dist',
    filename: 'bundle.[hash].js'
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /(node_modules|bower_components)/,
        use: {
          loader: 'babel-loader',
          options: {
            presets: [[ "env", {
              "targets": {
                "browsers": ["> 1% in FR"]
              }
            }]]
          }
        }
      }
    ]
  },
  plugins: [
    new CleanWebpackPlugin(['dist']),
    new HtmlWebpackPlugin({
      template: './src/index.html'
    })
  ]
};
```