Modules JavaScript

Modules JavaScript - Introduction

- ## JavaScript inventé en 1995 par Netscape

  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

  - Permet la création d'applications front-end, back-end, en ligne de commande, de bureau, mobiles...
  - Ces applications peuvent 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 - 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 button = new Y.MonBouton({
                value: 'Cliquez ici'
            });

            button.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

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

  - Les modules commons 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`

- ## Transormation en code client :
  browserify main.js > calculette-browser.js

```html
<!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 définie 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;
});
```

- Require JS définie 2 fonctions globales require et define. define permet de définir

```
<!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/calculette.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 './calculette';

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

répertoire dist/ :

babel src --out-dir dist/

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 - 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