CommonJS

Commonjs— это добровольная рабочая группа, которая проектирует, прототипирует и стандартизирует различные JavaScript API. На сегодняшний день они ратифицировали стандарты для модулей и пакетов — CommonJS определяют простой API для написания модулей, которые могут быть использованы в браузере с помощью тега <script>, как с синхронной, так и с асинхронной загрузкой. Реализация паттерна «модуль» с помощью CommonJS выглядит очень просто, и это уверенный шаг на пути к модульной системе, предложенной в ES Harmony (следующей версии JavaScript).

В структурном плане, CommonJS-модуль представляет собой готовый к переиспользованию фрагмент JavaScript-кода, который экспортирует специальные объекты, доступные для использования в любом зависимом коде. CommonJS все чаще используется как стандартный формат JavaScript-модулей. Существует большое количество хороших уроков по написанию CommonJS-модулей, но обычно они описывают две главных идеи: объект exports, содержащий то, что модуль хочет сделать доступным для других частей системы, и функцию require, которая используется одними модулями для импорта объекта exports из других.

Пример обеспечения совместимости между AMD и обычным CommonJS с помощью

создания обертки над последним:

(function(define) {

define(function(require,exports) {

// module contents

var dep1 = require("dep1");

exports.someExportedFunction = function() {...};

//...

});

})(typeof define=="function"?define:function(factory){factory(require,exports)});

Есть много хороших JavaScript-библиотек, для загрузки модулей в формате CommonJS, но моим личным предпочтением является RequireJS. Полный учебник по RequireJS выходит за рамки этого руководства, но я могу порекомендовать вам почитать пост Джеймса Брука «ScriptJunkie». Кроме того, я знаю многих людей, которые предпочитают Yabble.

Из коробки, RequireJS уже содержит методы для простого создания статичных модулей с обертками. Благодаря этим методам становится действительно легко создавать модули с поддержкой асинхронной загрузки. RequireJS может легко загружать модули и их зависимости, выполняя тело модуля, сразу, как только это становится возможным.

Некоторые разработчики утверждают, что CommonJS-модули недостаточно удобны для применения в браузере, потому как без определенной помощи со стороны сервера, их нельзя загрузить с помощью тега script. Давайте представим, что есть некая библиотека для кодирования изображений в виде ASCII-изображений, которая экспортирует функцию encodeToASCII. Модуль использующий эту библиотеку будет выглядеть примерно так:

var encodeToASCII = require("encoder").encodeToASCII;

exports.encodeSomeSource = function() {

// Обработка изображения, затем вызов encodeToASCII

}

Этот код не будет работать с тегом script. Ему необходим определенный контекст. Я имею в виду наш метод encodeToASCII, который ссылается на несуществующие в контексте window методы require и exports. В такой ситуации нам пришлось бы писать require и exports для каждого отдельного модуля. Эту проблему легко решают клиентские библиотеки, которые загружают скрипты через XHR-запросы, а затем выполняют eval().

Попробуем переписать этот модуль, используя RequireJS:

define(function(require, exports, module) {

var encodeToASCII = require("encoder").encodeToASCII;

exports.encodeSomeSource = function() {

// Обработка изображения, затем вызов encodeToASCII

}

});

Для разработчиков, которые хотят пойти дальше простого использования JavaScript в своих проектах, CommonJS модули — прекрасная возможность начать движение в эту сторону, но придется потратить немного времени и познакомиться поближе с этим форматом.