

RequireJS

RequireJS

- Cargador de ficheros y módulos
- Asíncrono
- Optimizado para el navegador

RequireJS

Permite:

- Cargar los recursos bajo demanda
- Cargar las dependencias automáticamente
- Organizar el código con más orden
- Optimizar, si fuera necesario, a un solo .js

Introduccion

- requirejs/ejercicio1/
- Empezando con una estructura tal que así:


```
.  
├── index.html  
└── js  
    ├── lib  
    │   └── jquery.min.js  
    └── main.js
```

Introducción

index.html

```
<!doctype html>
<html lang="">
  <head>
    <meta charset="utf-8">
    <title>RequireJS: ejemplo 1</title>
  </head>

  <body>
    <h1> RequireJS: ejemplo 1 </h1>
    <script src="js/main.js"></script>
  </body>
</html>
```



Introducción

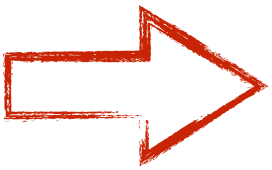
Para hacer la carga con RequireJS:

1. En el HTML, cargar solamente require.js
2. data-main apuntando al script a cargar

Introducción

```
<!doctype html>
<html lang="">
  <head>
    <meta charset="utf-8">
    <title>RequireJS: ejemplo 1</title>
  </head>

  <body>
    <h1> RequireJS: ejemplo 1 </h1>
    <script data-main="js/main.js"
      src="js/require.js">
    </script>
  </body>
</html>
```



Introducción

`main.js` ahora es el *punto de entrada*. Desde aquí:

- Podemos configurar RequireJS
- Cargamos los *módulos* que necesitamos

Introducción

Un módulo es:

- Un fichero **.js**
- Cuyo código está envuelto en una llamada a **define**
- Que exporta su valor de retorno

Introducción

```
define(function() {  
    console.log("Hola, soy un modulo!");  
    return 42;  
});
```

Introducción

`require([modulo, ...], callback)`

- Carga los módulos especificados
- Ejecuta el callback cuando estén todos listos
- El callback recibe un parámetro por módulo (su valor de retorno)

Introducción

js/main.js

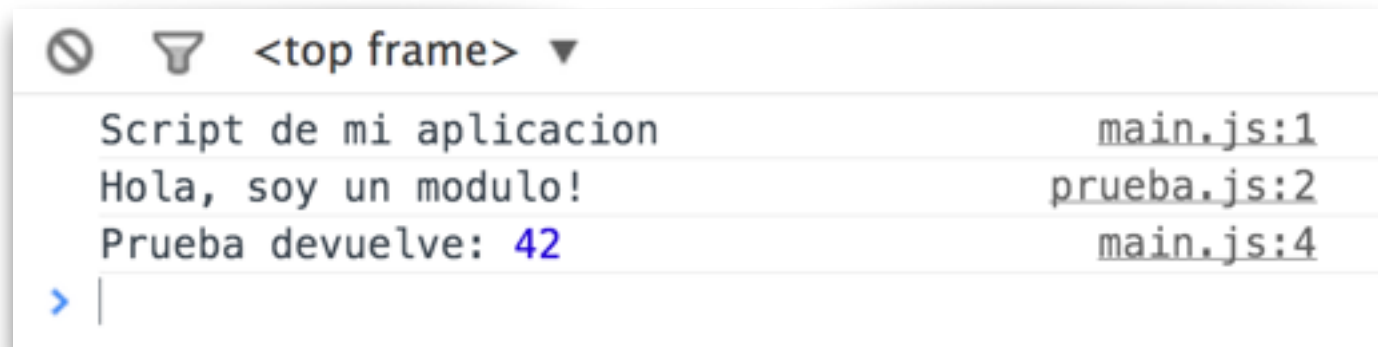
```
console.log("Script de mi aplicacion");

require(["prueba"], function(prueba) {
  console.log("Prueba devuelve:", prueba);
});
```

js/prueba.js

```
define(function() {
  console.log("Hola, soy un modulo!");
  return 42;
});
```

Introducción



The screenshot shows a browser's developer console with the following content:

- At the top, there is a toolbar with a disabled icon (a circle with a slash), a filter icon (a funnel), and a dropdown menu showing "<top frame>" with a downward arrow.
- Below the toolbar, there are three log entries:
 - The first entry has the text "Script de mi aplicacion" and the source "main.js:1".
 - The second entry has the text "Hola, soy un modulo!" and the source "prueba.js:2".
 - The third entry has the text "Prueba devuelve: 42" and the source "main.js:4". The number "42" is highlighted in blue.
- At the bottom, there is a prompt character ">" followed by a vertical cursor bar, indicating the console is ready for input.

Dependencias

Dependencias

Un módulo puede importar otros módulos

- Pasando un array de dependencias a define
- Primero se cargan las dependencias
- Después se ejecuta el callback
- Recibiendo como parámetros el valor de retorno de los módulos
- La función de un módulo se ejecuta, como máximo, una vez (se cachea el resultado)

Dependencias

- requirejs/ejercicio2/

```
.  
├── index.html  
└── js  
    ├── main.js  
    ├── modulo1.js  
    ├── modulo2.js  
    ├── require.js  
    └── submodulo.js
```


Dependencias

```
* main.js ejecutando                main.js:3  
*** submodulo ejecutando            submodulo.js:3  
** modulo1 ejecutando               modulo1.js:3  
** modulo2 ejecutando               modulo2.js:3  
m1: 21                             main.js:6  
m2: 84                             main.js:7
```

> |

Configurar RequireJS

Configurar RequireJS

```
require.config({ option: value, ...});
```

- Se suele llamar desde el punto de entrada (**data-main**) antes de cargar ningún módulo
- No se puede invocar desde dentro de un módulo

Configurar RequireJS

baseUrl

- Ruta base para la búsqueda de módulos
- No se aplica a las rutas que:
 - Empiecen con una barra /
 - Incluyan un protocolo
 - Acaben en .js

Configurar RequireJS

```
require.config({  
  baseUrl: "assets/js"  
});
```

```
require(["modulo1", "modulo2"], function(m1, m2) {  
  console.log("m1:", m1);  
  console.log("m2:", m2);  
});
```



/assets/js/modulo1.js

Configurar RequireJS

paths

- Alias de rutas
- Relativas a **baseUrl**
 - A menos que empiecen por /
 - O contengan un protocolo
- Para ficheros o directorios
- Se nombran sin extensión

Configurar RequireJS

```
require.config({
  paths: {
    "jquery": "../assets/js/libs/jquery",
    "views": "../assets/js/views"
  }
});

require(["jquery", "views/user"], function($, userView) {
  console.log($);
  console.log(userView);
});
```

Configurar RequireJS

map

- Crear un alias de ruta para los modulos que satisfagan cierto prefijo
- Util para mantener diferentes versiones de una dependencia

Configurar RequireJS

```
require.config({  
  map: {  
    "*" : {  
      "jquery" : "lib/jquery2"  
    },  
    "old/module" : {  
      "jquery" : "lib/jquery1.8"  
    }  
  }  
});
```

Configurar RequireJS

waitSeconds

- Timeout para la carga de scripts, en segundos
- Ponerlo a 0 deshabilita el timeout
- Por defecto, 7 segundos

Configurar RequireJS

config

- Pasar parámetros de configuración a un módulo
- Para acceder a estos datos hay que cargar el módulo especial “**module**”

Configurar RequireJS

main.js

```
require.config({  
  config: {  
    "modulo1": { clave: "valor" },  
    "modulo2": { color: {r: 0, g: 0, b: 0} }  
  }  
});
```

modulo2.js

```
define([ "module" ], function(module) {  
  "use strict";  
  console.log(module.config().color);  
});
```

Shims

Shims

Integración de componentes que:

- No siguen el estándar AMD
- Exportan variables globales
- Generalmente, librerías

Shims

```
require.config({
  shim: {
    "underscore": {
      exports: "_"
    },
    "backbone": {
      deps: [ "jquery", "underscore" ],
      exports: "Backbone"
    }
  }
});
```

Shims

- **exports**: la variable global que exporta el script
- **deps**: dependencias a cargar
- **init**: función a ejecutar cuando el script se termine de cargar
 - Si esta función devuelve algún valor, ese valor se considera el contenido del módulo

Shims

Una vez definido, un shim se carga como cualquier otro módulo.

```
define([ "backbone" ], function(Backbone) {  
    "use strict";  
    var Test = Backbone.Model.extend({});  
});
```

Sugar

Sugar

Cuando un módulo tiene muchas dependencias, la definición se vuelve muy confusa

```
define(  
  ["jquery", "jqm", "backbone", "R", "underscore", "views/  
index", "models", "rbbone", "storage_manager",  
"viewcontrollers", "views/index", "views/replenish_fresh/  
index", "views/stock_check/index", "views/manual_request/  
index", "views/new_designs/index"],  
  function($, jqm, Backbone, R, _, views, models, bbone,  
StorageManager, viewcontrollers, commonViews, rfViews,  
scViews, mrViews, ndViews) {  
  
    // ...  
  
  });
```

Sugar

RequireJS nos ofrece una sintaxis alternativa:

```
define(function(require) {  
    var Backbone = require("backbone"),  
        $ = require("jquery"),  
        subm = require("submodulo");  
  
    // ...  
});
```

Sugar

O la versión extendida, que sigue el estándar CommonJS:

```
define(function(require, exports, module) {  
  var subm = require("submodulo");  
  exports.value = subm / 42;  
  // module.exports = subm / 42;  
});
```

Sugar

Parámetro **module**:

- **id**: Identificador del modulo (“modulo2”)
- **uri**: URI del fichero (“js/modulo2.js”)
- **exports**: El valor exportado del módulo, por defecto un objeto vacío.
- **config**: Función que devuelve los parámetros de configuración que hayamos especificado con `require.config` (por defecto, un objeto vacío).

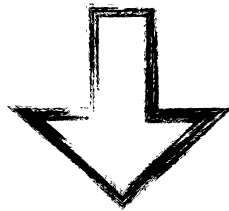
Sugar

Pero:

- Para hacer esto posible, RequireJS parsea el texto de la función y sustituye las llamadas a **require(...)** por dependencias.
- No funciona en ciertas versiones de Opera antiguas (por ej, el navegador de la PS3).
- No podemos hacer carga condicional.

Sugar

```
define(function(require) {  
  var subm = require("submodulo");  
  // ...  
});
```



```
define(["require", "submodulo"], function(require) {  
  var subm = require("submodulo");  
  // ...  
})
```


Sugar

No se puede hacer carga condicional

```
define(function(require) {  
  // Esto no funciona!!!  
  var a = (Math.random() > 0.5) ? require("b1")  
                                     : require("b2");  
  // ...  
});
```

Plugins

Plugins

RequireJS ofrece 4 plugins:

- **text.js** para cargar ficheros de texto
- **domReady.js** para ejecutar cuando la pagina haya cargado
- **i18n.js** para gestionar traducciones
- **cs.js** para cargar codigo coffeescript
- Muchos mas en <https://github.com/jrburke/requirejs/wiki/Plugins>

Plugins

text.js

```
define(  
  [ "text!templates/user.html" ],  
  function(userTemplate) {  
    "use strict";  
    console.log(userTemplate);  
  } );
```

Plugins

domReady.js

```
define([ "domReady!" ], function(doc) {  
    "use strict";  
    console.log( "Todo listo!" );  
});
```

Plugins

`text.js`

- Añadir traducciones gradualmente
- Siempre dentro de la carpeta **nls/**
- Idioma determinado por el navegador o con **requirejs.config**

Plugins


/modulo/nls/colors.js

```
define({  
  "root": {  
    "red": "red",  
    "blue": "blue",  
    "green": "green"  
  }  
});
```

Plugins

Para añadir otro idioma, modificamos el fichero..

```
define({  
  "root": {  
    "red": "red",  
    "blue": "blue",  
    "green": "green"  
  },  
  "es-es": true  
});
```



Plugins

Y creamos el objeto con las traducciones en la ruta apropiada: `/modulo/nls/es-es/colors.js`

```
define( {  
    "red": "rojo",  
    "blue": "azul",  
    "green": "verde"  
} );
```

Plugins

Para utilizar las traducciones:

```
define(  
  [ "i18n!modulo/nls/colors" ],  
  function(colors) {  
    "use strict";  
    console.log(colors.red);  
  } );
```

Plugins

Por defecto, el idioma se saca del navegador. Para forzar otro idioma:

```
requirejs.config({  
  config: {  
    i18n: {  
      locale: "es-es"  
    }  
  }  
});
```

Optimización

Optimización

RequireJS nos permite calcular todas las dependencias y empaquetarlas en un solo fichero.

- Aumentar la velocidad de carga
- Aprovechar el cache del navegador
- Aplicaciones con conectividad limitada

Optimización

r.js

- Un script node
- Muchísimas opciones de configuración
- <http://requirejs.org/docs/optimization.html>
- Para instalarlo: `npm install -g requirejs`

Optimización

El caso mas común: optimizar las dependencias de un fichero:

```
r.js -o name=main out=main-build.js baseUrl=.
```

Avanzado

Paquetes

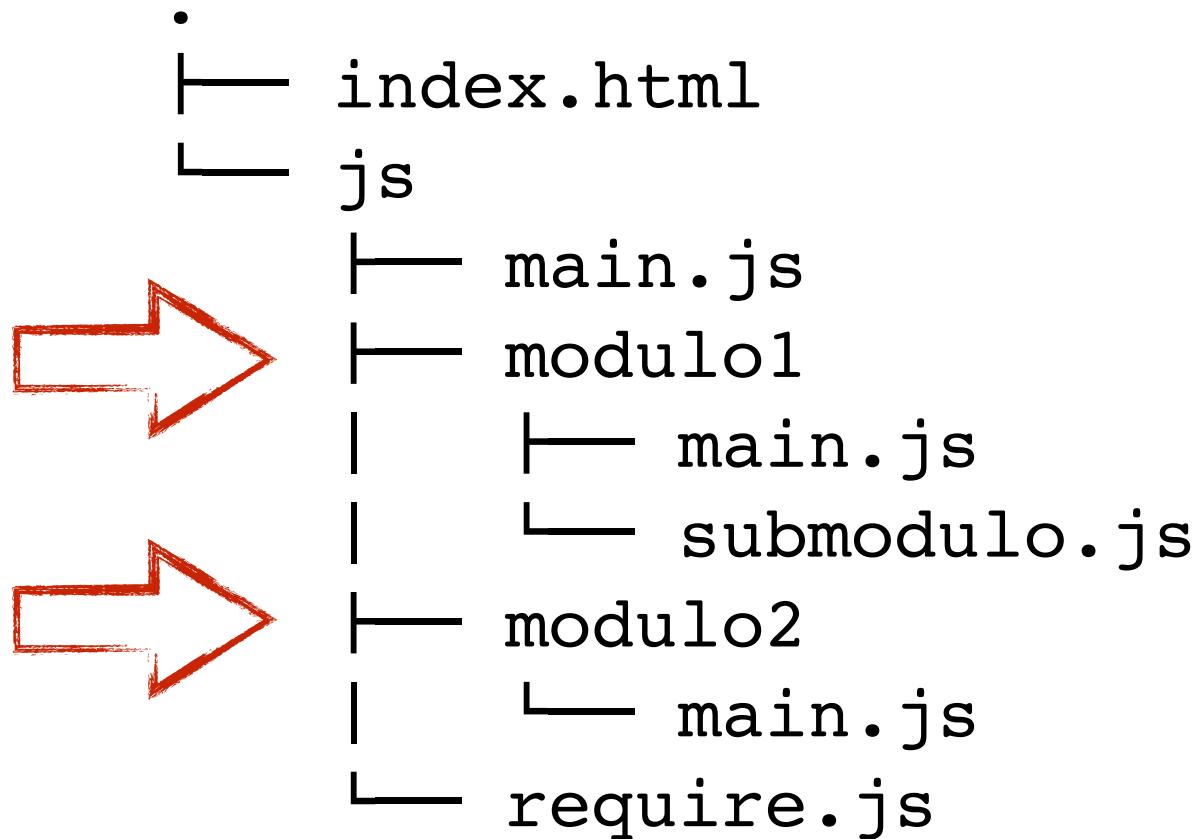
RequireJS permite cargar módulos que sigan la estructura de directorios de los paquetes

CommonJS

- Pero los ficheros han de seguir el estándar AMD y `user define(...)`

Paquetes

requirejs/ejercicio3/



Paquetes

requirejs/ejercicio3/main.js

```
requirejs.config({  
    "packages": [ "modulo1", "modulo2" ]  
});
```

```
console.log("* main.js ejecutando");
```

```
require([ "modulo1", "modulo2" ], function(m1, m2) {  
    console.log("m1:", m1);  
    console.log("m2:", m2);  
});
```

Gestión de Errores

RequireJS nos ofrece tres maneras de gestionar errores de carga:

- Errbacks específicos para cada caso
- Rutas alternativas
- **`requirejs.onError`**

Gestión de Errores

Errbacks específicos:

```
require([ "mi-modulo" ], function(miModulo) {  
    // ....  
}, function (err) {  
    // err.requireModules es una lista  
    // de los módulos que dieron error  
});
```

Gestión de Errores

Rutas alternativas

```
requirejs.config({  
  paths: {  
    jquery: [  
      "http://my.cdn.com/jquery.min",  
      "lib/jquery"  
    ]  
  }  
});
```

Gestión de Errores

Cualquier error no gestionado por errbacks locales, lo maneja `requirejs.onError`

```
requirejs.onError = function (err) {  
    console.log(err.requireType);  
    if (err.requireType === 'timeout') {  
        console.log('modules: ' + err.requireModules);  
    }  
  
    throw err;  
};
```