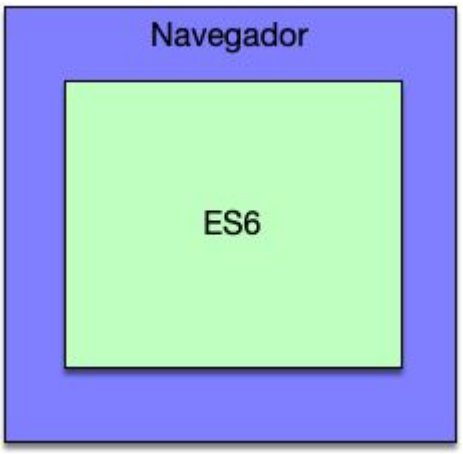
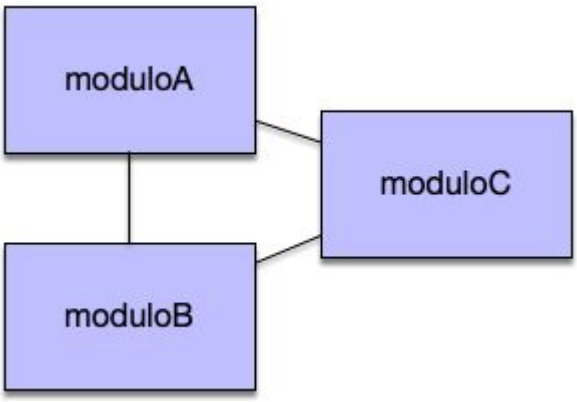
El concepto de ES6 Bundle (envoltorio) es bastante conocido. ¿A qué hace referencia?. Hace referencia a cuando nosotros tenemos un bloque de código construido con JavaScript ES6 y queremos desplegarlo en un navegador para que se ejecute. Es decir muchas veces nosotros queremos hacer uso de las habilidades de ES6 (JavaScript 2015) pero que luego nuestro código se ejecute de forma directa en un navegador. ¿ Es posible ejecutar código JavaScript ES6 en un navegador ? .



La realidad es que si solo tendríamos que ir a la página de CanIUse y ver cual es el soporte para JavaScript ES6. Sin embargo una de las características que siempre genera problemas es la gestión de módulos . Recordemos que JavaScript ES6 nos permite organizar el código de forma modular como si fueran packages de Java o NameSpaces de .NET.



¿Cómo podemos hacer que la gestión de estos módulos se ejecute de forma transparente en un navegador? Para ello necesitamos hacer uso de un module bundle . Existen muchos pero Rollup.js esta orientado a gestionar código construido en ES6 que es lo que a nosotros nos interesa. Vamos a construir un ejemplo sencillo , para ello lo primero que voy a construirme es una mini aplicación de Node.js que nos devuelve un listado de personas. El primer paso es generar un package.json usando:

npm init –yes

Esto nos generará un package.json por defecto

```
{
  "name": "servidor",
  "version": "1.0.0",
  "description": "",
  "main": "servidor.js",
  "dependencies": {
    "express": "^4.16.4",
    "whatwg-fetch": "^3.0.0"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Vamos ahora a instalar express.js para disponer de un servidor con un framework MVC.

npm install express –save-dev

Una vez instalado express.js ,pasamos a construir el código del lado servidor.

```
var express = require('express');
var app = express();
var lista=[];
lista.push({nombre:"pedro",edad:20});
lista.push({nombre:"gema",edad:30});

app.use(express.static('../dist'))

app.get('/personas', function (req, res) {
  res.send(lista);
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

Este servidor le arrancamos con la instrucción:

node servidor.js

accedemos al puerto 3000 y nos devuelve una lista de datos:

```
[{"nombre":"pedro","edad":20},{"nombre":"gema","edad":30}]
```

Demonos cuenta que ademas estamos publicando la carpeta /dist para que nos podamos descargar cualquier fichero de ella al navegador.

ES6 Bundle

Es momento de construir nuestro código de JavaScript ES6 para ser capaces de conectarnos a este servicio.

```
export class ServicioAjax {


      constructor(url) {

        this.url=url;

      }

      buscarPersonas() {

      return fetch(this.url).then(function(response) {
        return response.json();
      });

      }

}
```

En este caso hemos creado un servicio REST para obtener datos con Ajax usando fetch api , es momento de usar un programa principal e importar el módulo que hemos creado.

```
import {ServicioAjax} from "./moduloajax.js";

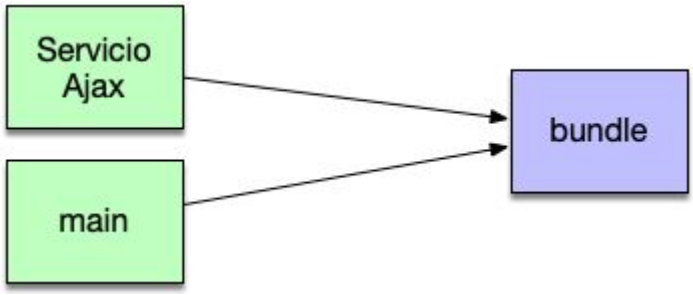

let servicio= new ServicioAjax("personas");
```

```
servicio.buscarPersonas().then((personas)=> {

      console.log(personas);

});
```

Estamos ya en una situación en la que tenemos dos ficheros de Javascript con JavaScript ES6.



Es momento de usar Rollup.js y empaquetarlo para que un navegador pueda permitir su ejecución como bundle (empaquetado) . Para ello construimos un sencillo fichero js de configuración (rollup-config.js)

```
module.exports = {
  input: 'src/001.js',
  output: {
    file: 'bundle.js',
    format: 'iife'
  }
};
```

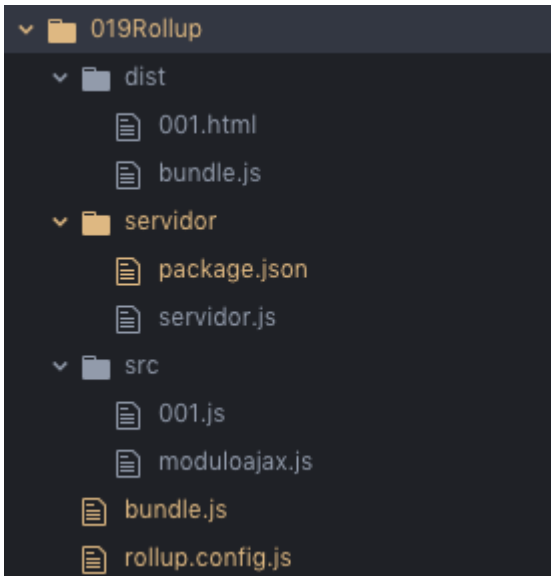
Este fichero

Ejecutamos desde linea de consola rollup -c y eso nos generará un bundle.

```
iMac-de-cecilio-2:019Rollup cecilioalvarezcaules$ rollup -c

src/001.js → bundle.js...
created bundle.js in 22ms
iMac-de-cecilio-2:019Rollup cecilioalvarezcaules$
```

La estructura de carpetas finales es la siguiente:

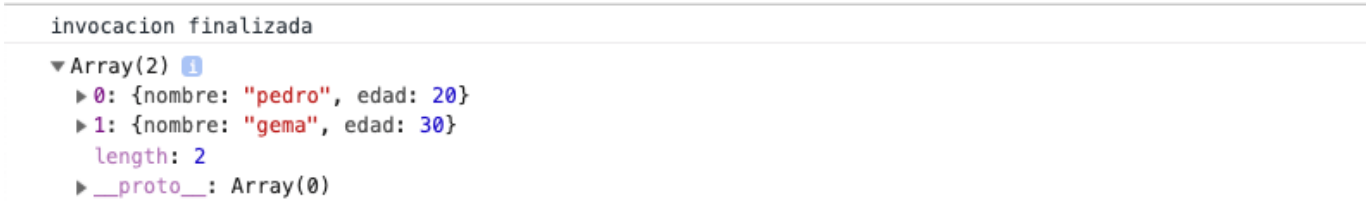


Una vez tenemos todo esto construido nos servirá con una página html que cargue en bundle desde nuestro servidor de node.

```
<html>
<head>
<script type="text/javascript" src="bundle.js">
</script>
</head>
```

```
<body>
</body>
</html>
```

Cargamos el fichero html desde un navegador y veremos los datos en la consola:



Acabamos de generar un bundle de JavaScript ES6.

1. JSP JavaScript Template Literal
2. WebPack y la gestión de dependencias en JavaScript
3. JavaScript High Order Functions y su manejo