

Desarrollo web con Angular: aplicaciones innovadoras y escalables

IFC0219CL



Temario

1. Introducción a JavaScript y Angular 7

- Repaso de JS
- ¿Por qué Angular?
- Ventajas
- Características
- Instalación
- Creación de proyecto

2. ES8

- Clases
- Arrow Functions
- Ámbito
- Operador de programación
- Argumentos variables
- Características
- Valores por defecto de los parámetros
- Destructuring

3. TypeScript

- TSC
- Types
- Funciones
- Interfaces
- Decoradores

4. Componentes

- Metadatos de componentes
- Creación de un componente
- Data binding
- Anidado de componentes
- Pasando datos al componente
- Respondiendo a eventos
- Ciclo de Vida
- Estilos

5. Directivas y servicios

- Attribute Directives
- Structural Directives
- Procesado de recursos
- Clientes RESTful

6. Routing

- Introducción
- Imports del Router
- Configuración
- Router Outlet
- Router Links
- Rutas con parámetros
- Usando los parámetros del padre
- Router Guards

Temario

7. Formularios

- Introducción
- Formularios basados en plantillas
- Crear Formulario
- Crear componente del formulario
- Revisar app.module.ts
- Crear una plantilla HTML inicial
- Añadir selector con directivas
- Controlar estado y validez
- Enviar formulario con ngSubmit

8. Pipes

- DatePipe
- DecimalPipe
- CurrencyPipe
- LowerCasePipe y UpperCasePipe
- JSONpipe
- PercentPipe y SlicePipe

9. HTTP

- Introducción
- Implementación
- Obteniendo datos JSON
- Por qué implementar un servicio
- Verificación de tipo de respuesta
- Leyendo la respuesta completa
- Manejar errores
- Obtener los detalles del error
- RetryObservables y operadores
- Petición de datos no JSON
- Enviar datos al servidor
- Petición POST
- Petición DELETE
- Petición PUT
- Alternativa axios

10. Testing

- Testing
- Karma
- Testing a componentes

11. Animaciones

1. Translate
2. Rotate
3. Scale
4. Animaciones

Temario

12. Angular Material Design

- Introducción
- ¿Qué es Material Design?
- Instalación
- Componentes
- Buttons
- Navigation – Menú
- Menú Anidado
- Navigation – Sidenav
- Navigation – Toolbar
- Posicionar toolbar content
- Controles de formulario
- Layout – Grid
- Layout – Card
- Componente Mat-table

13. Creación de un proyecto simple con Angular 7

- Conexión a base de datos
- Alta, baja y modificación y consulta de datos

Introducción a Angular

¿Qué es Angular?

¿Qué es Angular?

- Framework JS

¿Qué es Angular?

- Framework JS
- SPA: Single Page Applications

¿Qué es Angular?

- Framework JS
- SPA: Single Page Applications
- TypeScript

¿Qué es Angular?

- Framework JS
- SPA: Single Page Applications
- TypeScript
- Código fuente y código compilado

¿Qué es Angular?

- Framework JS
- SPA: Single Page Applications
- TypeScript
- Código fuente y código compilado
- ¿Angular 2? ¿8? ¿AngularJS?

Entorno de desarrollo



TypeScript

Entorno de desarrollo

- IDE: [Visual Studio Code](#)
 - [Spanish Language Pack for Visual Studio Code](#)
 - [EditorConfig for VS Code](#)
 - [TSLint](#)
 - [Angular Language Service](#)
 - Configurar:
 - Format on Paste y Format on Save
 - "editor.codeActionsOnSave": {
 "source.fixAll.tslint": true
}
- [Git](#)
- [Node.js](#) y npm
- Extensión [Augury](#) para el navegador

Git



Comandos básicos

Comandos básicos

- Clonar un repositorio:

```
git clone URL
```


Comandos básicos

- Clonar un repositorio:

```
git clone URL
```

- Descargar última versión del repositorio:

```
git pull origin master
```

Configuración proxy

```
git config --global http.proxy http://username:password@host:port
```

```
git config --global https.proxy http://username:password@host:port
```

Node.js y npm



npm

npm

- Instalar última versión después de instalar Node.js (configurar proxy si es necesario): `npm install -g npm`

npm

- Instalar última versión después de instalar Node.js (configurar proxy si es necesario): `npm install -g npm`
- Repositorio de módulos distribuibles

npm

- Instalar última versión después de instalar Node.js (configurar proxy si es necesario): `npm install -g npm`
- Repositorio de módulos distribuibles
- Módulos globales y módulos locales

npm

- Instalar última versión después de instalar Node.js (configurar proxy si es necesario): `npm install -g npm`
- Repositorio de módulos distribuibles
- Módulos globales y módulos locales
- La carpeta `node_modules`

npm

- Instalar última versión después de instalar Node.js (configurar proxy si es necesario): `npm install -g npm`
- Repositorio de módulos distribuibles
- Módulos globales y módulos locales
- La carpeta `node_modules`
- El archivo `package.json`:

npm

- Instalar última versión después de instalar Node.js (configurar proxy si es necesario): `npm install -g npm`
- Repositorio de módulos distribuibles
- Módulos globales y módulos locales
- La carpeta `node_modules`
- El archivo `package.json`:
 - Registro de dependencias

npm

- Instalar última versión después de instalar Node.js (configurar proxy si es necesario): `npm install -g npm`
- Repositorio de módulos distribuibles
- Módulos globales y módulos locales
- La carpeta `node_modules`
- El archivo `package.json`:
 - Registro de dependencias
 - Dependencias de desarrollo y de producción

npm

- Instalar última versión después de instalar Node.js (configurar proxy si es necesario): `npm install -g npm`
- Repositorio de módulos distribuibles
- Módulos globales y módulos locales
- La carpeta `node_modules`
- El archivo `package.json`:
 - Registro de dependencias
 - Dependencias de desarrollo y de producción
 - Versiones (SEMVER)

Comandos npm

Comandos npm

- Instalar un paquete globalmente:
npm install -g paquete

Comandos npm

- Instalar un paquete globalmente:
npm install -g paquete
- Instalar un paquete de producción:
npm install paquete

Comandos npm

- Instalar un paquete globalmente:
npm install -g paquete
- Instalar un paquete de producción:
npm install paquete
- Instalar un paquete de desarrollo:
npm install paquete --save-dev

Comandos npm

- Instalar un paquete globalmente:
npm install -g paquete
- Instalar un paquete de producción:
npm install paquete
- Instalar un paquete de desarrollo:
npm install paquete --save-dev
- Instalar todas las dependencias:
npm install

Comandos npm

- Instalar un paquete globalmente:
npm install -g paquete
- Instalar un paquete de producción:
npm install paquete
- Instalar un paquete de desarrollo:
npm install paquete --save-dev
- Instalar todas las dependencias:
npm install
- Instalar las dependencias de producción:
npm install --production

Comandos npm

- Instalar un paquete globalmente:
npm install -g paquete
- Instalar un paquete de producción:
npm install paquete
- Instalar un paquete de desarrollo:
npm install paquete --save-dev
- Instalar todas las dependencias:
npm install
- Instalar las dependencias de producción:
npm install --production
- Listar paquetes instalados:
npm list --depth=0 (locales)
npm list -g --depth=0 (globales)

angular-cli

angular-cli

- Instalación global:
npm install -g @angular/cli

Configuración proxy

```
npm config set proxy http://username:password@host:port
```

```
npm config set https-proxy http://username:password@host:port
```

JavaScript



JavaScript

JavaScript

- Interpretado, compilado y ejecutado en el navegador

JavaScript

- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS

JavaScript

- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS
- Estandarización: **ECMAScript**

JavaScript

- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS
- Estandarización: **ECMAScript**
- La versión **ES6** o **ES2015**

JavaScript

- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS
- Estandarización: **ECMAScript**
- La versión **ES6** o **ES2015**
- Transpiladores: Babel, TypeScript



TypeScript

JavaScript

- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS
- Estandarización: **ECMAScript**
- La versión **ES6** o **ES2015**
- Transpiladores: Babel, TypeScript



Organización del código JavaScript

Organización del código JavaScript

- Ejemplo de uso clásico de JS: utilizar un plugin de jQuery en nuestra web, o implementar alguna interacción con el usuario

Organización del código JavaScript

- Ejemplo de uso clásico de JS: utilizar un plugin de jQuery en nuestra web, o implementar alguna interacción con el usuario
- Pocas líneas de código, todas en un mismo archivo

Organización del código JavaScript

Organización del código JavaScript

```
<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/tabs.js"></script>
</head>
```

Organización del código JavaScript

```
(function($) {  
  
    $(document).ready(function() {  
  
        // Al hacer clic en una pestaña  
        $(".tab a").on("click", function(e) {  
            // Anulamos el link  
            e.preventDefault();  
  
            // Ocultamos todos los bloques de contenido  
            // y mostramos sólo el que se ha elegido  
            var content_id = $(this).attr("href");  
            $(".tab-content").hide();  
            $(content_id).show();  
  
            // Desmarcamos la pestaña que estuviera activa  
            // y marcamos la clicada como activa  
            $(".tab.active").removeClass("active");  
            $(this).closest(".tab").addClass("active");  
        })  
    })  
  
})(jQuery);
```

24 líneas

Organización del código JavaScript

Organización del código JavaScript

```
<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/ui.js"></script>
</head>
```

Organización del código JavaScript

```
(function($) {  
  
    $(document).ready(function() {  
        $(document).on('click', '.tab_new', offerGroupSwitchTab);  
        $(document).on('click', '.navigationServices-li', jumpTo);  
        $('.load-more_new').on('click', loadMore).each(function() {  
            $(this).data('main', $(this).text());  
        });  
    });  
})  
  
var loadMore = function(e) {  
    e.preventDefault();  
    var $list = $(this).prev('.promos-list_new');  
    var button_text = $(this).data('main');  
    var button_alt_text = $(this).data('alt');  
    if ($(window).width() > 992) {  
        var hidden_classes = ".hidden";  
        var $hidden = $list.find(hidden_classes);  
        var n_show = 3;  
    } else if ($(window).width() > 768) {  
        var hidden_classes = ".hidden, .hidden-sm";  
        var $hidden = $list.find(hidden_classes);  
        var n_show = 2;  
    } else {  
        var hidden_classes = ".hidden, .hidden-sm, .hidden-xs";  
        var $hidden = $list.find(hidden_classes);  
        var n_show = 1;  
    }  
}
```

75 líneas

Organización del código JavaScript

Organización del código JavaScript

- Programar toda la UI de una página

Organización del código JavaScript

- Programar toda la UI de una página

```
(function() {
  CodeMirror.defineMode("javascript", function(config, parserConfig) {
    var indentUnit = config.indentUnit;
    var jsonMode = parserConfig.json;

    // Tokenizer
    var keywords = function(){
      function kw(type) {return {type: type, style: "keyword"};}
      var A = kw("keyword a"), B = kw("keyword b"), C = kw("keyword c");
      var operator = kw("operator"), atom = {type: "atom", style: "atom"};
      return {
        "if": A, "while": A, "with": A, "else": B, "do": B, "try": B, "finally": B,
        "return": C, "break": C, "continue": C, "new": C, "delete": C, "throw": C,
        "var": kw("var"), "const": kw("var"), "let": kw("var"),
        "function": kw("function"), "catch": kw("catch"),
        "for": kw("for"), "switch": kw("switch"), "case": kw("case"), "default": kw("default"),
        "in": operator, "typeof": operator, "instanceof": operator,
        "true": atom, "false": atom, "null": atom, "undefined": atom, "NaN": atom, "Inf": atom
      };
    };

    var isOperatorChar = /[+\-*/%=<>!?|]/;

```

Organización del código JavaScript

- Programar toda la UI de una página

```
(function() {  
  CodeMirror.defineMode("javascript", function(config, parserConfig)  
    var indentUnit = config.indentUnit;  
    var jsonMode = parserConfig.json;  
  
    // Tokenizer  
  
    var keywords = function(){  
      function kw(type) {return {type: type, style: "keyword"};}  
      var A = kw("keyword a"), B = kw("keyword b"), C = kw("keyword c");  
      var operator = kw("operator"), atom = {type: "atom", style: "atom"};  
      return {  
        "if": A, "while": A, "with": A, "else": B, "do": B, "try": B, "finally": B,  
        "return": C, "break": C, "continue": C, "new": C, "delete": C, "throw": C,  
        "var": kw("var"), "const": kw("var"), "let": kw("var"),  
        "function": kw("function"), "catch": kw("catch"),  
        "for": kw("for"), "switch": kw("switch"), "case": kw("case"), "default": kw("de  
        "in": operator, "typeof": operator, "instanceof": operator,  
        "true": atom, "false": atom, "null": atom, "undefined": atom, "NaN": atom, "Inf      };  
    }();  
  
    var isOperatorChar = /[+\-*&%=<>!?|]/;
```

1445
líneas

Organización del código JavaScript

Organización del código JavaScript

- ¿2000 líneas en un solo archivo?

Organización del código JavaScript

- ¿2000 líneas en un solo archivo?

Ventajas	Inconvenientes

Organización del código JavaScript

- ¿2000 líneas en un solo archivo?

Ventajas	Inconvenientes
	<ul style="list-style-type: none">• Difícil de leer/entender

Organización del código JavaScript

- ¿2000 líneas en un solo archivo?

Ventajas	Inconvenientes
	<ul style="list-style-type: none">• Difícil de leer/entender• Difícil de mantener

Organización del código JavaScript

- ¿2000 líneas en un solo archivo?

Ventajas	Inconvenientes
	<ul style="list-style-type: none">• Difícil de leer/entender• Difícil de mantener• Poca reusabilidad

Organización del código JavaScript

- ¿2000 líneas en un solo archivo?

Ventajas	Inconvenientes
	<ul style="list-style-type: none">• Difícil de leer/entender• Difícil de mantener• Poca reusabilidad• Difícil encontrar código no usado

Organización del código JavaScript

- ¿2000 líneas en un solo archivo?

Ventajas	Inconvenientes
	<ul style="list-style-type: none">• Difícil de leer/entender• Difícil de mantener• Poca reusabilidad• Difícil encontrar código no usado• Colisiones de nombres

Organización del código JavaScript

- ¿2000 líneas en un solo archivo?

Ventajas	Inconvenientes
<ul style="list-style-type: none">• Una sola petición HTTP	<ul style="list-style-type: none">• Difícil de leer/entender• Difícil de mantener• Poca reusabilidad• Difícil encontrar código no usado• Colisiones de nombres

Organización del código JavaScript

Organización del código JavaScript

- Optimización: dividir el código en varios archivos/módulos

Organización del código JavaScript

- Optimización: dividir el código en varios archivos/módulos

```
├─ js
  └─ modules
    └─ JS banners.js
    └─ JS carousel.js
    └─ JS gallery.js
    └─ JS lightbox.js
    └─ JS navigation.js
    └─ JS scroll.js
    └─ JS slideshow.js
    └─ JS tabs.js
```

Organización del código JavaScript

- Optimización: dividir el código en varios archivos/módulos

```
<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/modules/tabs.js"></script>
  <script src="js/modules/banners.js"></script>
  <script src="js/modules/lightbox.js"></script>
  <script src="js/modules/scroll.js"></script>
  <script src="js/modules/carousel.js"></script>
  <script src="js/modules/slideshow.js"></script>
  <script src="js/modules/gallery.js"></script>
  <script src="js/modules/navigation.js"></script>
</head>
```

```
├─ js
  └─ modules
    JS banners.js
    JS carousel.js
    JS gallery.js
    JS lightbox.js
    JS navigation.js
    JS scroll.js
    JS slideshow.js
    JS tabs.js
```


Organización del código JavaScript

```
<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/modules/tabs.js"></script>
  <script src="js/modules/banners.js"></script>
  <script src="js/modules/lightbox.js"></script>
  <script src="js/modules/scroll.js"></script>
  <script src="js/modules/carousel.js"></script>
  <script src="js/modules/slideshow.js"></script>
  <script src="js/modules/gallery.js"></script>
  <script src="js/modules/navigation.js"></script>
</head>
```

```
├ js
├ modules
  JS banners.js
  JS carousel.js
  JS gallery.js
  JS lightbox.js
  JS navigation.js
  JS scroll.js
  JS slideshow.js
  JS tabs.js
```

Ventajas

Inconvenientes

```
<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/modules/tabs.js"></script>
  <script src="js/modules/banners.js"></script>
  <script src="js/modules/lightbox.js"></script>
  <script src="js/modules/scroll.js"></script>
  <script src="js/modules/carousel.js"></script>
  <script src="js/modules/slideshow.js"></script>
  <script src="js/modules/gallery.js"></script>
  <script src="js/modules/navigation.js"></script>
</head>
```

```
├ js
├ modules
  JS banners.js
  JS carousel.js
  JS gallery.js
  JS lightbox.js
  JS navigation.js
  JS scroll.js
  JS slideshow.js
  JS tabs.js
```

Ventajas

Inconvenientes

- Difícil encontrar código no usado (menos difícil que antes)

```
<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/modules/tabs.js"></script>
  <script src="js/modules/banners.js"></script>
  <script src="js/modules/lightbox.js"></script>
  <script src="js/modules/scroll.js"></script>
  <script src="js/modules/carousel.js"></script>
  <script src="js/modules/slideshow.js"></script>
  <script src="js/modules/gallery.js"></script>
  <script src="js/modules/navigation.js"></script>
</head>
```

```
├ js
├ modules
  JS banners.js
  JS carousel.js
  JS gallery.js
  JS lightbox.js
  JS navigation.js
  JS scroll.js
  JS slideshow.js
  JS tabs.js
```

Ventajas

Inconvenientes

- Difícil encontrar código no usado (menos difícil que antes)
- Colisiones de nombres

```
<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/modules/tabs.js"></script>
  <script src="js/modules/banners.js"></script>
  <script src="js/modules/lightbox.js"></script>
  <script src="js/modules/scroll.js"></script>
  <script src="js/modules/carousel.js"></script>
  <script src="js/modules/slideshow.js"></script>
  <script src="js/modules/gallery.js"></script>
  <script src="js/modules/navigation.js"></script>
</head>
```

```
├ js
├ modules
  JS banners.js
  JS carousel.js
  JS gallery.js
  JS lightbox.js
  JS navigation.js
  JS scroll.js
  JS slideshow.js
  JS tabs.js
```

Ventajas

Inconvenientes

- Difícil encontrar código no usado (menos difícil que antes)
- Colisiones de nombres
- Muchas peticiones HTTP

```
<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/modules/tabs.js"></script>
  <script src="js/modules/banners.js"></script>
  <script src="js/modules/lightbox.js"></script>
  <script src="js/modules/scroll.js"></script>
  <script src="js/modules/carousel.js"></script>
  <script src="js/modules/slideshow.js"></script>
  <script src="js/modules/gallery.js"></script>
  <script src="js/modules/navigation.js"></script>
</head>
```

```
├─ js
  └─ modules
    JS banners.js
    JS carousel.js
    JS gallery.js
    JS lightbox.js
    JS navigation.js
    JS scroll.js
    JS slideshow.js
    JS tabs.js
```

Ventajas

Inconvenientes

- Difícil encontrar código no usado (menos difícil que antes)
- Colisiones de nombres
- Muchas peticiones HTTP
- El orden importa: dependencias

```
<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/modules/tabs.js"></script>
  <script src="js/modules/banners.js"></script>
  <script src="js/modules/lightbox.js"></script>
  <script src="js/modules/scroll.js"></script>
  <script src="js/modules/carousel.js"></script>
  <script src="js/modules/slideshow.js"></script>
  <script src="js/modules/gallery.js"></script>
  <script src="js/modules/navigation.js"></script>
</head>
```

```
├─ js
  └─ modules
    JS banners.js
    JS carousel.js
    JS gallery.js
    JS lightbox.js
    JS navigation.js
    JS scroll.js
    JS slideshow.js
    JS tabs.js
```

Ventajas

- Legible e inteligible

Inconvenientes

- Difícil encontrar código no usado (menos difícil que antes)
- Colisiones de nombres
- Muchas peticiones HTTP
- El orden importa: dependencias

```
<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/modules/tabs.js"></script>
  <script src="js/modules/banners.js"></script>
  <script src="js/modules/lightbox.js"></script>
  <script src="js/modules/scroll.js"></script>
  <script src="js/modules/carousel.js"></script>
  <script src="js/modules/slideshow.js"></script>
  <script src="js/modules/gallery.js"></script>
  <script src="js/modules/navigation.js"></script>
</head>
```

```
├ js
├ modules
  JS banners.js
  JS carousel.js
  JS gallery.js
  JS lightbox.js
  JS navigation.js
  JS scroll.js
  JS slideshow.js
  JS tabs.js
```

Ventajas

- Legible e inteligible
- Fácil de mantener

Inconvenientes

- Difícil encontrar código no usado (menos difícil que antes)
- Colisiones de nombres
- Muchas peticiones HTTP
- El orden importa: dependencias


```

<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/modules/tabs.js"></script>
  <script src="js/modules/banners.js"></script>
  <script src="js/modules/lightbox.js"></script>
  <script src="js/modules/scroll.js"></script>
  <script src="js/modules/carousel.js"></script>
  <script src="js/modules/slideshow.js"></script>
  <script src="js/modules/gallery.js"></script>
  <script src="js/modules/navigation.js"></script>
</head>

```

```

├─ js
  └─ modules
    JS banners.js
    JS carousel.js
    JS gallery.js
    JS lightbox.js
    JS navigation.js
    JS scroll.js
    JS slideshow.js
    JS tabs.js

```

Ventajas

- Legible e inteligible
- Fácil de mantener
- Reutilizable

Inconvenientes

- Difícil encontrar código no usado (menos difícil que antes)
- Colisiones de nombres
- Muchas peticiones HTTP
- El orden importa: dependencias

```
<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/modules/tabs.js"></script>
  <script src="js/modules/banners.js"></script>
  <script src="js/modules/lightbox.js"></script>
  <script src="js/modules/scroll.js"></script>
  <script src="js/modules/carousel.js"></script>
  <script src="js/modules/slideshow.js"></script>
  <script src="js/modules/gallery.js"></script>
  <script src="js/modules/navigation.js"></script>
</head>
```

```
├─ js
  └─ modules
    JS banners.js
    JS carousel.js
    JS gallery.js
    JS lightbox.js
    JS navigation.js
    JS scroll.js
    JS slideshow.js
    JS tabs.js
```

Ventajas

- Legible e inteligible
- Fácil de mantener
- Reutilizable
- Cargamos sólo lo que necesitamos

Inconvenientes

- Difícil encontrar código no usado (menos difícil que antes)
- Colisiones de nombres
- Muchas peticiones HTTP
- El orden importa: dependencias

Organización del código JavaScript

Organización del código JavaScript

- Dependencias: es difícil asegurar el orden, y no es posible tener dependencias circulares

```

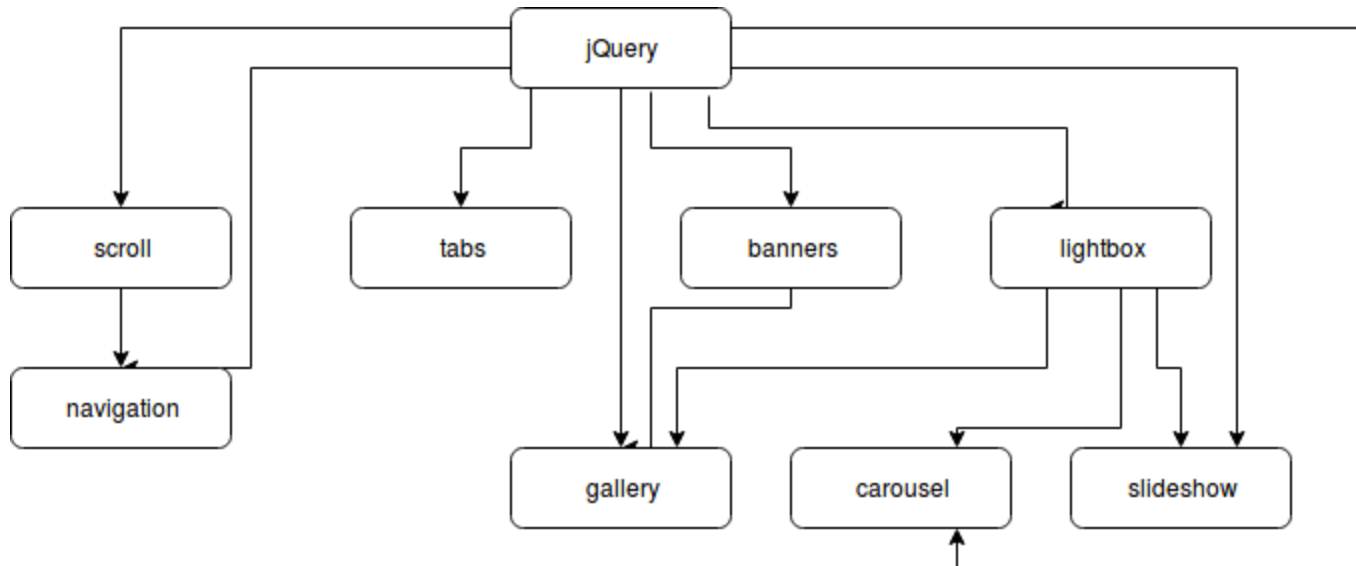
<head>
  <meta charset="UTF-8">
  <title>Mi web</title>
  <script src="vendor/jquery/jquery.min.js"></script>
  <script src="js/modules/tabs.js"></script>
  <script src="js/modules/banners.js"></script>
  <script src="js/modules/lightbox.js"></script>
  <script src="js/modules/scroll.js"></script>
  <script src="js/modules/carousel.js"></script>
  <script src="js/modules/slideshow.js"></script>
  <script src="js/modules/gallery.js"></script>
  <script src="js/modules/navigation.js"></script>
</head>

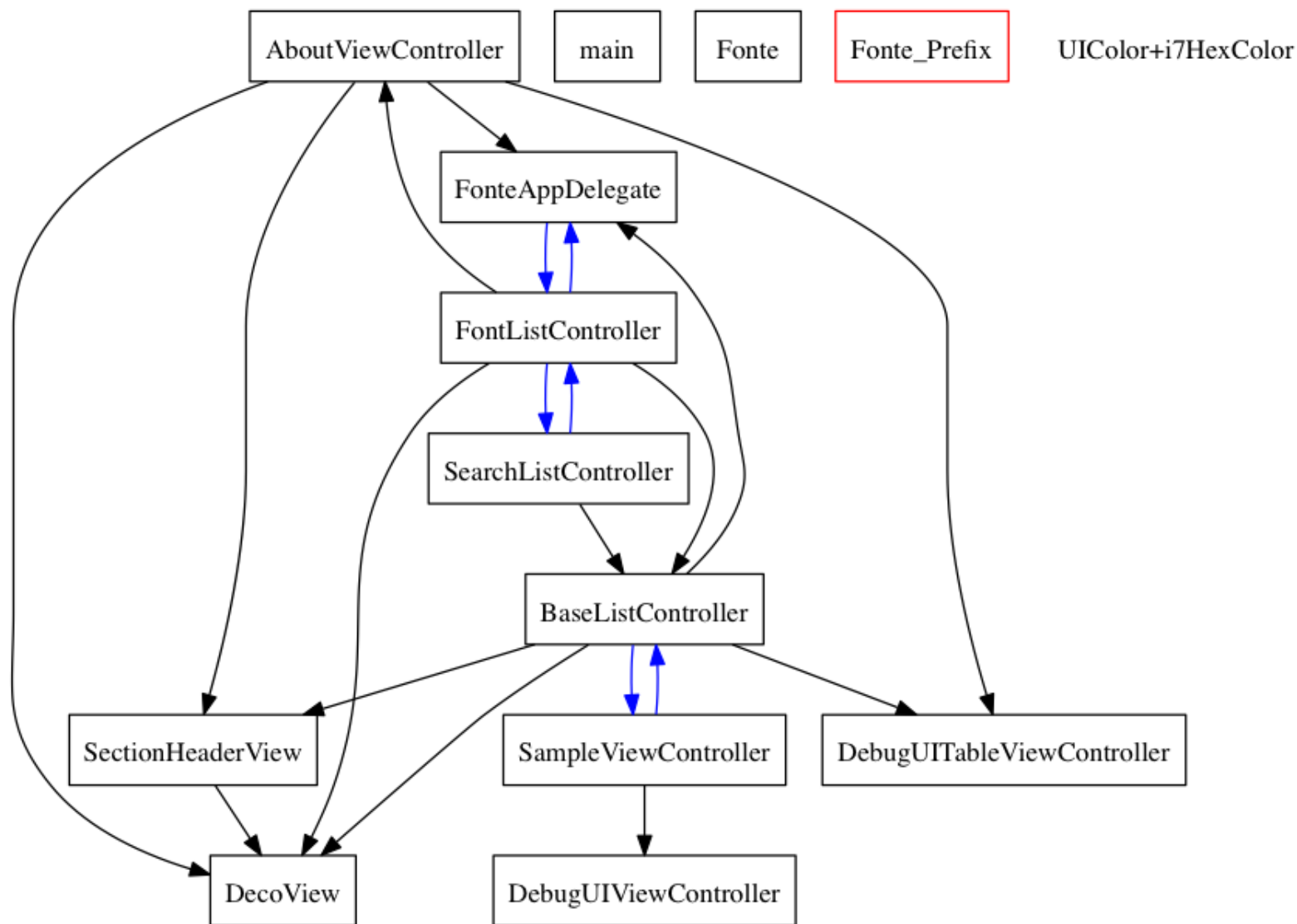
```

```

├─ js
  └─ modules
    JS banners.js
    JS carousel.js
    JS gallery.js
    JS lightbox.js
    JS navigation.js
    JS scroll.js
    JS slideshow.js
    JS tabs.js

```





Organización del código JavaScript: módulos

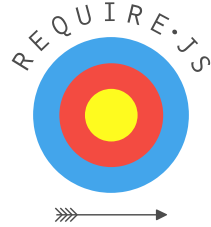
Organización del código JavaScript: módulos

- **Module loaders:** ellos gestionan las dependencias y cargan los módulos (RequireJS, SystemJS)



SystemJS

Organización del código JavaScript: módulos

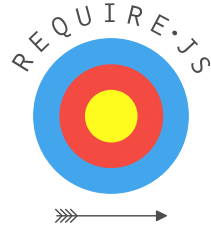


SystemJS

Ventajas

Inconvenientes

Organización del código JavaScript: módulos



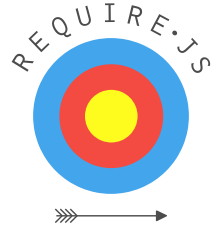
SystemJS

Ventajas

Inconvenientes

- Difícil encontrar código no usado (menos difícil que antes)
- Muchas peticiones HTTP

Organización del código JavaScript: módulos



SystemJS

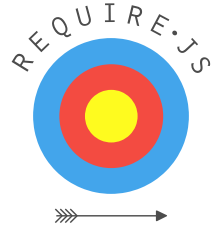
Ventajas

- Legible e inteligible
- Fácil de mantener
- Reutilizable
- Cargamos sólo lo que necesitamos

Inconvenientes

- Difícil encontrar código no usado (menos difícil que antes)
- Muchas peticiones HTTP

Organización del código JavaScript: módulos



SystemJS

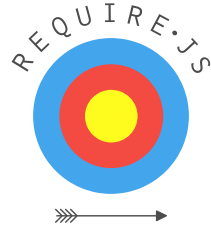
Ventajas

- Legible e inteligible
- Fácil de mantener
- Reutilizable
- Cargamos sólo lo que necesitamos
- Gestión automática de dependencias

Inconvenientes

- Difícil encontrar código no usado (menos difícil que antes)
- Muchas peticiones HTTP

Organización del código JavaScript: módulos



SystemJS

Ventajas

- Legible e inteligible
- Fácil de mantener
- Reutilizable
- Cargamos sólo lo que necesitamos
- Gestión automática de dependencias
- Encapsulación

Inconvenientes

- Difícil encontrar código no usado (menos difícil que antes)
- Muchas peticiones HTTP

Organización del código JavaScript: módulos

Organización del código JavaScript: módulos

- **Module bundlers:** además de lo anterior, generan un solo código encadenado y minificado (Browserify, webpack, Parcel)



Organización del código JavaScript: módulos



Ventajas

Organización del código JavaScript: módulos



Ventajas

- Legible e inteligible
- Fácil de mantener
- Reutilizable
- Cargamos sólo lo que necesitamos
- Gestión automática de dependencias
- Encapsulación

Organización del código JavaScript: módulos



Ventajas

- Legible e inteligible
- Fácil de mantener
- Reutilizable
- Cargamos sólo lo que necesitamos
- Gestión automática de dependencias
- Encapsulación
- Una o muy pocas conexiones HTTP

Organización del código JavaScript: módulos



Ventajas

- Legible e inteligible
- Fácil de mantener
- Reutilizable
- Cargamos sólo lo que necesitamos
- Gestión automática de dependencias
- Encapsulación
- Una o muy pocas conexiones HTTP
- Eliminación de código no usado (*tree shaking*)

Organización del código JavaScript: módulos

Organización del código JavaScript: módulos

- ¿Puedo escribir mis módulos como yo quiera? ¿hay un estándar?

Organización del código JavaScript: módulos

- ¿Puedo escribir mis módulos como yo quiera? ¿hay un estándar?
- AMD: Asynchronous Module Definition

Organización del código JavaScript: módulos

- ¿Puedo escribir mis módulos como yo quiera? ¿hay un estándar?
- AMD: Asynchronous Module Definition

```
define(['myModule', 'myOtherModule'], function(myModule, myOtherModule) {  
  
    return {  
        hello: function() {  
            console.log('hello');  
        },  
        goodbye: function() {  
            console.log('goodbye');  
        }  
    };  
});
```

Organización del código JavaScript: módulos

- ¿Puedo escribir mis módulos como yo quiera? ¿hay un estándar?
- AMD: Asynchronous Module Definition
- CommonJS

Organización del código JavaScript: módulos

- ¿Puedo escribir mis módulos como yo quiera? ¿hay un estándar?
- AMD: Asynchronous Module Definition
- CommonJS

```
var myModuleA = require('myModuleA');

function myModuleB() {
  this.hello = function() {
    return 'hello!';
  }

  this.goodbye = function() {
    return 'goodbye!';
  }
}

module.exports = myModuleB;
```

Organización del código JavaScript: módulos

- ¿Puedo escribir mis módulos como yo quiera? ¿hay un estándar?
- AMD: Asynchronous Module Definition
- CommonJS
- UMD: Universal Module Definition

Organización del código JavaScript: módulos

- ¿Puedo escribir mis módulos como yo quiera? ¿hay un estándar?
- AMD: Asynchronous Module Definition
- CommonJS
- UMD: Universal Module Definition

```
(function (root, factory) {  
  if (typeof define === 'function' && define.amd) {  
    // AMD  
    define(['myModule', 'myOtherModule'], factory);  
  } else if (typeof exports === 'object') {  
    // CommonJS  
    module.exports = factory(require('myModule'), require('myOtherModule'));  
  } else {  
    // Browser globals (Note: root is window)  
    root.returnExports = factory(root.myModule, root.myOtherModule);  
  }  
})(this, function (myModule, myOtherModule) {  
  // Methods  
  function notHelloOrGoodbye(){}; // A private method
```

Organización del código JavaScript: módulos

- ¿Puedo escribir mis módulos como yo quiera? ¿hay un estándar?
- AMD: Asynchronous Module Definition
- CommonJS
- UMD: Universal Module Definition
- ES6 Modules

Organización del código JavaScript: módulos

- ¿Puedo escribir mis módulos como yo quiera? ¿hay un estándar?
- AMD: Asynchronous Module Definition
- CommonJS
- UMD: Universal Module Definition
- ES6 Modules

```
import { method1 } from './moduleA.js';  
  
method1("hello");  
  
export let method2 = function() {  
  console.log("Method 2");  
}
```

Organización del código JavaScript: módulos

Organización del código JavaScript: módulos

- ¿AMD, CommonJS, UMD, ES6?

Organización del código JavaScript: módulos

- ¿AMD, CommonJS, UMD, ES6?
- Compatibilidad de los módulos ES6 en navegadores

Organización del código JavaScript: módulos

- ¿AMD, CommonJS, UMD, ES6?
- Compatibilidad de los módulos ES6 en navegadores
- ¡Webpack!

Organización del código JavaScript: módulos

- ¿AMD, CommonJS, UMD, ES6?
- Compatibilidad de los módulos ES6 en navegadores
- ¡Webpack!
- TypeScript usa la sintaxis ES6

Organización del código JavaScript: módulos

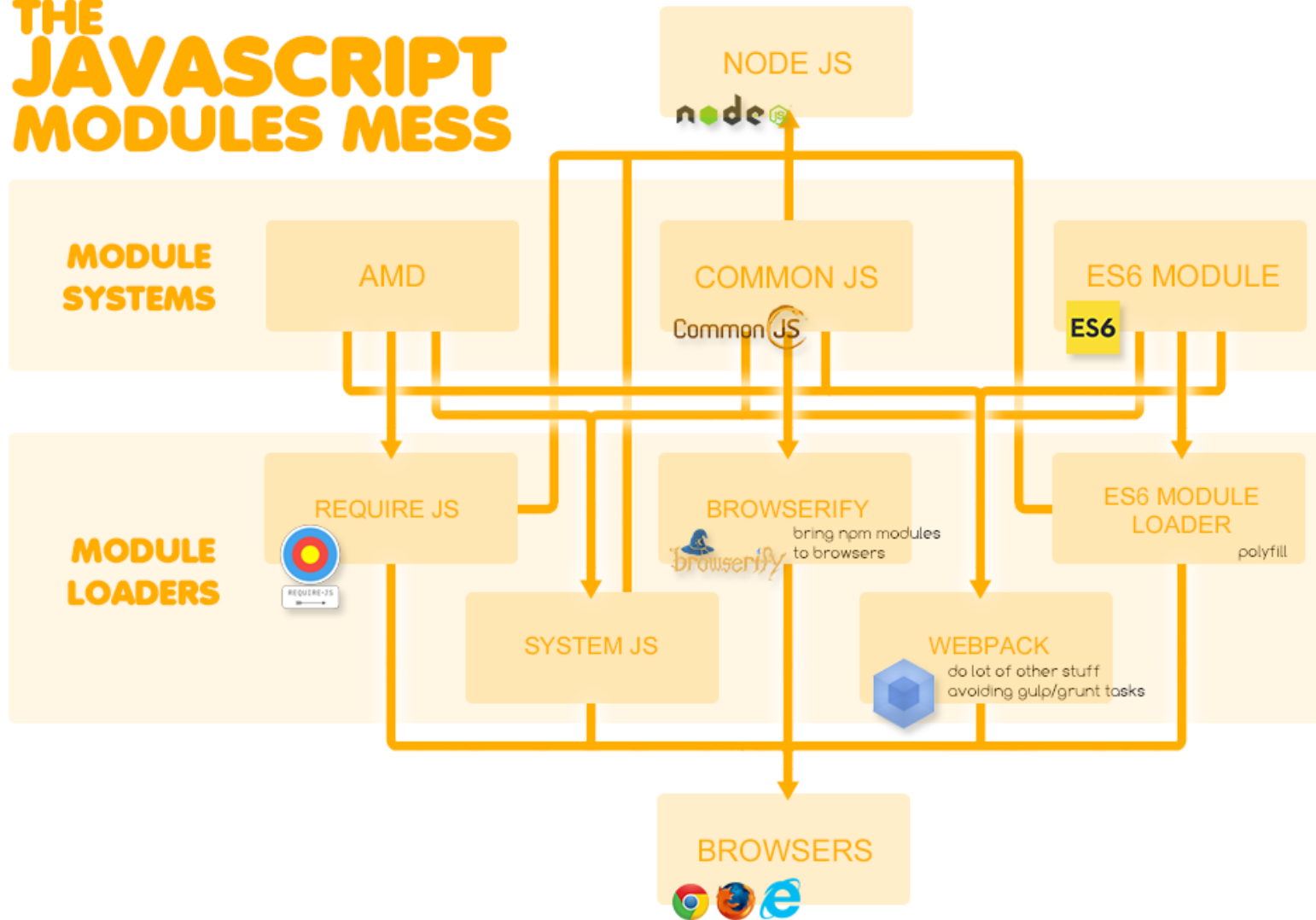
- ¿AMD, CommonJS, UMD, ES6?
- Compatibilidad de los módulos ES6 en navegadores
- ¡Webpack!
- TypeScript usa la sintaxis ES6
- TS -> ES5 -> webpack -> bundle -> browser =

Organización del código JavaScript: módulos

- ¿AMD, CommonJS, UMD, ES6?
- Compatibilidad de los módulos ES6 en navegadores
- ¡Webpack!
- TypeScript usa la sintaxis ES6
- TS -> ES5 -> webpack -> bundle -> browser = **Angular CLI**



THE JAVASCRIPT MODULES MESS



ES6

ES6

- let y const

ES6

- let y const

```
let a = 3;

let a = 10; // Error
var a = 12; // Error

const b = 10;

b = 3; // Error

const obj = {
  x: 10,
  y: 12
}

obj.x = 15; // OK

obj = { // Error
  x: 15,
  y: 12
}
```


ES6

- let y const
- Template literals

ES6

- let y const
- Template literals

```
let nombre = "Antonio";

let cuadrado = function(x) {
  return x * x;
}

let n = Math.floor(Math.random() * 10);

let saludo1 = "Hola, " + nombre + ". El cuadrado de " + n + " es " + cuadrado(n) + ".";
let saludo2 = `Hola, ${nombre}. El cuadrado de ${n} es ${cuadrado(n)}.`;
```

ES6

- let y const
- Template literals
- for ... of

ES6

```
let nombres = ["Patricia", "Zacarías", "Miguel", "Maite"]

for (let i in nombres) {
  console.log(nombres[i]);
}

for (let nombre of nombres) {
  console.log(nombre);
}

let obj = {
  x: 3,
  y: 4
}

for (let i in obj) {
  console.log(obj[i]);
}

let nombre = "Antonio Jesús";

for (let c of nombre) {
  console.log(c);
}
```

ES6

- let y const
- Template literals
- for ... of
- Funciones

ES6

- let y const
- Template literals
- for ... of
- Funciones
 - Parámetros por defecto

ES6

```
function potencia(x, y = 2) {  
    return Math.pow(x, y);  
}  
  
console.log(`10 elevado a 8 es ${potencia(10, 8)}`)  
console.log(`El cuadrado de 5 es ${potencia(5)}`);
```

ES6

- let y const
- Template literals
- for ... of
- Funciones
 - Parámetros por defecto
 - Función arrow:
(parámetros) => expresión_devuelta;

ES6

```
const potencia = function (x, y = 2) {  
    return Math.pow(x, y);  
}  
  
const potencia = (x, y = 2) => Math.pow(x, y);  
setTimeout(() => console.log("pausa"), 2000);
```

ES6

ES6

- Operador spread

ES6

- Operador spread
 - Parámetros en funciones

ES6

- Operador spread
 - Parámetros en funciones
 - Enviar varios parámetros a partir de un array

ES6

- Operador spread
 - Parámetros en funciones
 - Enviar varios parámetros a partir de un array
 - push y unshift

ES6

- Operador spread
 - Parámetros en funciones
 - Enviar varios parámetros a partir de un array
 - push y unshift
 - Intercalar un array dentro de otro

ES6

- Operador spread
 - Parámetros en funciones
 - Enviar varios parámetros a partir de un array
 - push y unshift
 - Intercalar un array dentro de otro
 - Copiar un array en otro

ES6

- Operador spread
 - Parámetros en funciones
 - Enviar varios parámetros a partir de un array
 - push y unshift
 - Intercalar un array dentro de otro
 - Copiar un array en otro
 - Copiar un objeto en otro

ES6

```
// function(a, b, c)
let nums = [1, 3, 6];
function sumar(a, b, c) {
  console.log(a + b + c);
}
sumar(...nums);

// function(n parámetros)
let a = 3;
let b = 7;
let c = 8;

function sumar(...nums) {
  let suma = 0;
  for (n of nums) {
    suma += n;
  }
  console.log("La suma es " + suma);
}
sumar(a, b, c);
```

ES6

ES6

- Destructuring

ES6

- Destructuring
 - Asignar desde un array

ES6

```
function medidasMueble() {  
  // ...  
  
  return [100, 70, 20];  
}  
  
let [ancho, alto, profundo] = medidasMueble();  
  
console.log(ancho, alto, profundo);  
  
// 100, 70, 20
```

ES6

- Destructuring
 - Asignar desde un array
 - Asignar desde un objeto

ES6

```
function getRGB(colorHex) {  
  // ...  
  
  return {  
    alias: 'deeppink',  
    red: 255,  
    green: 20,  
    blue: 147,  
    alpha: 0.8  
  }  
}  
  
let { red, green, blue } = getRGB("#ff1493");  
  
console.log(red, green, blue);  
  
// 255, 20, 147
```


ES6

```
let personas = [{
  nombre: "Luis",
  apellido: "Herrera",
  edad: 23
},
{
  nombre: "Marta",
  apellido: "Nieto",
  edad: 29
}];

for (let {nombre, edad} of personas) {
  console.log(`Me llamo ${nombre} y tengo ${edad} años`)
}

// Me llamo Luis y tengo 23 años
// Me llamo Marta y tengo 29 años
```

ES6

- Destructuring
 - Asignar desde un array
 - Asignar desde un objeto
 - Renombrar variables

ES6

```
let notas = {  
  mat: 8,  
  fis: 6,  
  dib: 5,  
  tec: 6  
}  
  
let { mat: matematicas, fis: fisica, dib: dibujo, tec: tecnologia } = notas  
  
console.log(matematicas, fisica, dibujo, tecnologia);  
  
// 8, 6, 5, 6
```

ES6

- Destructuring
 - Asignar desde un array
 - Asignar desde un objeto
 - Renombrar variables
 - Valores por defecto

ES6

```
let persona = {  
  nombre: "Luis",  
  edad: 23  
}  
  
let { nombre, edad, estado = "soltero" } = persona  
  
console.log(nombre, edad, estado);  
  
// Luis, 23, soltero
```

ES6

- Destructuring
 - Asignar desde un array
 - Asignar desde un objeto
 - Renombrar variables
 - Valores por defecto
 - Nested destructuring

ES6

```
persona = {  
  nombre: "Francisco José",  
  apellidos: "González Primo",  
  edad: 40,  
  residencia: {  
    ciudad: "Cornellà de Llobregat",  
    provincia: "Barcelona",  
    direccion: {  
      calle: "Londres",  
      numero: 67,  
      piso: ["2º", "B"]  
    }  
  }  
}  
  
{ nombre, apellidos, residencia: { ciudad, direccion: { calle, numero, piso: [ planta, puerta ] } }  
  
console.log(nombre, apellidos, ciudad, calle, numero, planta, puerta);  
  
Francisco José González Primo Cornellà de Llobregat Londres 67 2º B
```

ES6

- Destructuring
 - Asignar desde un array
 - Asignar desde un objeto
 - Renombrar variables
 - Valores por defecto
 - Nested destructuring
 - Intercambiar variables

ES6

```
let a = 10;  
let b = 20;  
  
[a, b] = [b, a];  
  
console.log(a, b);  
  
// 20, 10
```

ES6

- Destructuring
 - Asignar desde un array
 - Asignar desde un objeto
 - Renombrar variables
 - Valores por defecto
 - Nested destructuring
 - Intercambiar variables
 - Argumentos en las funciones

ES6

```
function area({radio = 0, base = 0, altura = 0, tipo = 'circulo'} = {}  
  console.log(radio, base, altura, tipo);  
}  
area({ tipo: 'rectangulo', base: 10, altura: 20 });  
// 0, 10, 20, "rectangulo"  
  
area();  
// 0, 0, 0, "circulo"
```

ES6

ES6

- Clases

ES6

- Clases
 - Propiedades y métodos

ES6

```
class A {  
  
    constructor(z) {  
        this.x = 3;  
        this.y = 10;  
        this.z = z;  
    }  
  
    suma() {  
        return this.x + this.y + this.z;  
    }  
}  
  
let a = new A(20);  
  
console.log(a.suma());
```

ES6

- Clases
 - Propiedades y métodos
 - Getters y setters

ES6

```
class A {  
  
    constructor(z) {  
        this.x = 3;  
        this.y = 10;  
        this.z = z;  
    }  
  
    suma() {  
        return this.x + this.y + this.z;  
    }  
  
    set zeta(z) {  
        this.z = z * 2;  
    }  
  
    get zeta() {  
        return this.z / 2;  
    }  
}  
  
let a = new A(20);  
  
a.zeta = 15;  
  
console.log(a.zeta);
```

ES6

- Clases
 - Propiedades y métodos
 - Getters y setters
 - Métodos estáticos

ES6

```
class A {  
  
    constructor(z) {  
        this.x = 3;  
        this.y = 10;  
        this.z = z;  
    }  
  
    static getPI() {  
        return 3.14159;  
    }  
  
    suma() {  
        return this.x + this.y + this.z;  
    }  
  
    set zeta(z) {  
        this.z = z * 2;  
    }  
  
    get zeta() {  
        return this.z / 2;  
    }  
}  
  
let a = new A(20);  
  
a.zeta = 15;  
  
console.log(a.zeta);  
  
console.log(A.getPI());
```

ES6

- Clases
 - Propiedades y métodos
 - Getters y setters
 - Métodos estáticos
 - Herencia con extends y super()

ES6

```
class A {  
  
    constructor(z) {  
        this.x = 3;  
        this.y = 10;  
        this.z = z;  
    }  
  
    static getPI() {  
        return 3.14159;  
    }  
  
    suma() {  
        return this.x + this.y + this.z;  
    }  
  
    set zeta(z) {  
        this.z = z * 2;  
    }  
  
    get zeta() {  
        return this.z / 2;  
    }  
}  
  
class B extends A {  
    constructor() {  
        super(100);  
        this.x = 20;  
    }  
  
    suma() {  
        return this.x + this.z;  
    }  
}
```

ES6

ES6

- Módulos

- import

```
import { literal } from 'ruta_modulo';
```

```
import literal from 'ruta_modulo';
```

```
import * as literal from 'ruta_modulo';
```

```
import 'ruta_modulo';
```

- export

```
export let a = 3;
```

```
export let class Clase {
```

```
    ...
```

```
}
```

```
export default {
```

```
    key: value
```

```
}
```

ES6

ES6

- Módulos
 - import dinámicos

```
import('ruta_modulo').then(  
  modulo => ...  
);
```

Programación funcional con arrays

- Métodos:
 - map

Programación funcional con arrays

```
let nombres = ["juan", "luisa", "amparo", "arturo"];  
nombres = nombres.map(nombre => nombre.toUpperCase());  
console.log(nombres);
```

Programación funcional con arrays

- Métodos:
 - map
 - filter

Programación funcional con arrays

```
let personas = [  
  {  
    nombre: "juan",  
    edad: 15  
  },  
  {  
    nombre: "luisa",  
    edad: 35  
  },  
  {  
    nombre: "amparo",  
    edad: 17  
  },  
  {  
    nombre: "arturo",  
    edad: 32  
  }  
];  
  
let mayoresEdad = personas.filter(persona => persona.edad >= 18);  
  
console.log(mayoresEdad);
```

Programación funcional con arrays

- Métodos:
 - map
 - filter
 - reduce

Programación funcional con arrays

```
let nums = [2, 4, 10, 15, 12];

let suma = nums.reduce((x, y) => x + y);

let objs = [
  {
    x: 3,
    y: 2
  },
  {
    x: 8,
    y: 10
  },
  {
    x: 10,
    y: 15
  }
]

let sumaX = objs.reduce((x, o2) => x + o2.x, 0); // Método 1
let sumaX = objs.map(o => o.x).reduce((x, y) => x + y); // Método 2
```

Programación funcional con arrays

- Métodos:
 - map
 - filter
 - reduce
 - find

Programación funcional con arrays

```
let notas = [  
  {  
    nombre: "juan",  
    nota: 6  
  },  
  {  
    nombre: "luisa",  
    nota: 8  
  },  
  {  
    nombre: "amparo",  
    nota: 4  
  },  
  {  
    nombre: "arturo",  
    nota: 3  
  }  
];  
  
let notaArturo = notas.find(n => n.nombre === "arturo");
```

Programación funcional con arrays

- Métodos:
 - map
 - filter
 - reduce
 - find
- Encadenamiento

Programación funcional con arrays

```
let notas = [  
  {  
    nombre: "juan",  
    nota: 6  
  },  
  {  
    nombre: "luisa",  
    nota: 8  
  },  
  {  
    nombre: "amparo",  
    nota: 4  
  },  
  {  
    nombre: "arturo",  
    nota: 3  
  }  
];  
  
let notasAprobados = notas.filter(n => n.nota >= 5).map(n => n.nota)  
console.log(notasAprobados);
```

TypeScript



TypeScript

TypeScript

- Superconjunto de JavaScript

TypeScript

- Superconjunto de JavaScript
- Transpila a ES5 (o a otra versión)

TypeScript

- Superconjunto de JavaScript
- Transpila a ES5 (o a otra versión)

```
TS scripts.ts  x  ...  JS scripts.js  x
1  let a: number = 4;
2  let b: number = 10;
3
4  let suma = (x: number, y: number, z: number = 3): number => x + y + z;
5
1  var a = 4;
2  var b = 10;
3
4  var suma = function (x, y, z) {
5      if (z === void 0) { z = 3; }
6      return x + y + z;
7  };
8
```


TypeScript

- Superconjunto de JavaScript
- Transpila a ES5 (o a otra versión)

TypeScript

- Superconjunto de JavaScript
- Transpila a ES5 (o a otra versión)
- Tipado

TypeScript

- Superconjunto de JavaScript
- Transpila a ES5 (o a otra versión)
- Tipado
- Errores en tiempo de compilación

TypeScript

```
let a: number;  
let b: number;  
let resultado: string;
```

```
function sumar(x: number, y: number): string {  
  let suma: number = x + y;  
  return `El resultado de ${x} + ${y} es ${suma}`;  
}
```

[ts] No se puede asignar un argumento de tipo ""Hola"" al parámetro de tipo "number".

```
sumar('Hola', 'Adiós');  
sumar(a, b);
```

TypeScript

- Superconjunto de JavaScript
- Transpila a ES5 (o a otra versión)
- Tipado
- Errores en tiempo de compilación
- tsconfig.json

TypeScript

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "es2015",
    "moduleResolution": "node",
    "sourceMap": true,
    "outDir": "./public/js/",
  }
}
```

tsconfig.json

TypeScript - Tipos

TypeScript - Tipos

- Tipos básicos:

TypeScript - Tipos

- Tipos básicos:
 - number

TypeScript - Tipos

- Tipos básicos:
 - number
 - string

TypeScript - Tipos

- Tipos básicos:
 - number
 - string
 - boolean

TypeScript - Tipos

- Tipos básicos:
 - number
 - string
 - boolean
 - Array

TypeScript - Tipos

- Tipos básicos:
 - number
 - string
 - boolean
 - Array
 - any

TypeScript - Tipos

- Tipos básicos:
 - number
 - string
 - boolean
 - Array
 - any
 - void

TypeScript - Tipos

```
let peso: number;
peso = 89.5;

let saludo: string;
saludo = 'Vais a petarlo con TypeScript';

let esVerano: boolean;
esVerano = false;

let nums: Array<number>;
nums = [10, 55, -3, 4.14];

let nombres: string[];
nombres = ['Juan', 'Paqui', 'Lorenzo', 'Alicia'];

let cosas: any[];
cosas = [10, 'Teruel', -5, true, [0, -10, 15], false];

function imprimeSaludo(s: string): void {
  console.log(s);
}
imprimeSaludo('Buenas tardes');
```

TypeScript - Tipos

- Tipos básicos:
 - number
 - string
 - boolean
 - Array
 - any
 - void
- Enum

TypeScript - Tipos

```
enum FormasPago {
  TPV,
  PayPal,
  transferencia
}
let pago: FormasPago;

pago = FormasPago.PayPal;
procesarPago(pago);

function procesarPago(formaPago: FormasPago): void {
  switch (formaPago) {
    case FormasPago.TPV:
      // ...
      break;
    case FormasPago.PayPal:
      // ...
      break;
    case FormasPago.transferencia:
      // ...
      break;
  }
}
```

TypeScript - Tipos

- Tipos básicos:
 - number
 - string
 - boolean
 - Array
 - any
 - void
- Enum
- Union types

TypeScript - Tipos

```
let numeros: Array<number | string>;
numeros = ['3', 6, '15.8', 0];

function procesar(a: string | number): void {
  if (typeof a === 'string') {
    console.log(a.toUpperCase());
  } else {
    console.log(a.toFixed(2));
  }
}
```

TypeScript - Tipos

- Tipos básicos:
 - number
 - string
 - boolean
 - Array
 - any
 - void
- Enum
- Union types
- Genéricos

TypeScript - Tipos

```
function verDoble<T>(elem: T): T[] {  
  let elemDoble: T[] = [elem, elem];  
  return elemDoble;  
}
```

TypeScript - Tipos

- Tipos básicos:
 - number
 - string
 - boolean
 - Array
 - any
 - void
- Enum
- Union types
- Genéricos
- Type assertion

TypeScript - Tipos

```
const inputText = <HTMLInputElement>document.getElementById("nombre");  
inputText.select();
```

TypeScript - Funciones

TypeScript - Funciones

- Sin flexibilidad en el número de parámetros

TypeScript - Funciones

```
function sumar(a: number, b: number): number
    return a + b;
}

sumar(); // Error
sumar(3); // Error
sumar(10, 2); // OK
sumar(4, -3, 10, 8) // Error
```

TypeScript - Funciones

- Sin flexibilidad en el número de parámetros
- Parámetros opcionales

TypeScript - Funciones

```
function sumar(a: number, b: number, c?: number): number {
  if (c) {
    return a + b + c;
  } else {
    return a + b;
  }
}

sumar(10, 2);
sumar(10, 2, 15);
```

TypeScript - Funciones

- Sin flexibilidad en el número de parámetros
- Parámetros opcionales
- Sobrecarga

TypeScript - Funciones

```
function nChars(a: number): string;
function nChars(a: string): number;
function nChars(a: string | number): number | string {
  if (typeof a === 'number') {
    return '¡Es un número!';
  } else if (typeof a === 'string') {
    return a.length;
  }
}

type RGB = [number, number, number];

function convierteColor(color: string): RGB;
function convierteColor(color: RGB): string;
function convierteColor(color: string | RGB): string | RGB {
  if (typeof color === 'string') {
    return [0, 128, 0];
  } else {
    return '#006600';
  }
}

const colorRGB = convierteColor('#006600');
const colorHEX = convierteColor([0, 128, 0]);
```

TypeScript - Funciones

- Sin flexibilidad en el número de parámetros
- Parámetros opcionales
- Sobrecarga
- Function types

TypeScript - Funciones

```
function transformaNumero(x: number, callback: (n: number) => void) {
    callback(x);
}

let a = 10;

transformaNumero(a, m => console.log(m * 2));
```


TypeScript - Módulos

TypeScript - Módulos

- Sintaxis ES6:
 - `import { literal } from 'ruta_modulo';`
`import literal from 'ruta_modulo';`
`import * as literal from 'ruta_modulo';`
`import 'ruta_modulo';`
 - `export let a = 3;`
`export class Clase {`
 ...
}
 - `export default {`
 key: value
}

TypeScript - Módulos

TypeScript - Módulos

- Módulos
 - import dinámicos

```
import('ruta_modulo').then(  
  modulo => ...  
);
```

TypeScript - Módulos

TypeScript - Módulos

- Sintaxis ES6

TypeScript - Módulos

- Sintaxis ES6
- Se omite la extensión .ts

TypeScript - Módulos

- Sintaxis ES6
- Se omite la extensión .ts
- Importar de paquetes npm: nombre del paquete
`import { } from 'paquete';`

TypeScript - Módulos

- Sintaxis ES6
- Se omite la extensión .ts
- Importar de paquetes npm: nombre del paquete
`import { } from 'paquete';`
- Importar de nuestros módulos: rutas relativas
`import { } from './modulo';`

TypeScript - Clases

TypeScript - Clases

- Propiedades fuera del constructor

TypeScript - Clases

```
class Factura {  
  numero: string;  
  base: number;  
  tipoIva: number;  
  
  constructor(numero: string, base: number, tipoIva: number = 21)  
  {  
    this.numero = numero;  
    this.base = base;  
    this.tipoIva = tipoIva;  
  }  
}
```

TypeScript - Clases

- Propiedades fuera del constructor
- Visibilidad de los miembros

TypeScript - Clases

- Propiedades fuera del constructor
- Visibilidad de los miembros
- Getters y setters

TypeScript - Clases

- Propiedades fuera del constructor
- Visibilidad de los miembros
- Getters y setters
- Modificador readonly

TypeScript - Clases

- Propiedades fuera del constructor
- Visibilidad de los miembros
- Getters y setters
- Modificador readonly
- Propiedades estáticas

TypeScript - Clases

```
class Factura {
  private static caracteresSerie = 2;
  public num: string;
  public serie: string;
  public base: number;
  private readonly intTipoIva: number;

  constructor(base: number, tipoIva: number = 21) {
    this.base = base;
    this.intTipoIva = tipoIva;
  }

  get numero(): string {
    return this.serie + this.num;
  }

  set numero(n: string) {
    this.serie = n.slice(0, Factura.caracteresSerie - 1);
    this.num = n.slice(Factura.caracteresSerie);
  }
}

let f = new Factura(100);

f.numero = 'AB600';
console.log(f.numero);
```

TypeScript - Clases

- Propiedades fuera del constructor
- Visibilidad de los miembros
- Getters y setters
- Modificador readonly
- Propiedades estáticas
- Métodos abstractos

TypeScript - Clases

```
abstract class Vehiculo {  
    public manual: boolean;  
  
    constructor(public ruedas: number, public motor: Motor)  
        this.manual = this.motor === Motor.ninguno;  
    }  
  
    public abstract arrancar(): void;  
}  
  
class Bici extends Vehiculo {  
    public arrancar(): void {  
        console.log('Me pongo de pie y pedaleo');  
    }  
}
```

TypeScript - Clases

- Propiedades fuera del constructor
- Visibilidad de los miembros
- Getters y setters
- Modificador readonly
- Propiedades estáticas
- Métodos abstractos
- Interfaces

TypeScript - Clases

```
interface Arrancable {
  arrancar(): void;
  apagar(): void;
}

abstract class Vehiculo {

  public manual: boolean;

  constructor(public ruedas: number, public motor: Motor) {
    this.manual = this.motor === Motor.ninguno;
  }
}

class Bici extends Vehiculo implements Arrancable {
  public arrancar(): void {
    console.log('Me pongo de pie y pedaleo');
  }
  public apagar(): void {
    console.log('Me bajo de la bici');
  }
}
```

TypeScript - Clases

```
interface Cliente {
  id: number;
  login: string;
  nombre: string;
  tipo: TiposCliente;
  fechaAlta: Date;
}

function getClientes(): Cliente[] {
  let clientes: Cliente[] = conectaBD('clientes');
  return clientes;
}
```

TypeScript - Decoradores

TypeScript - Decoradores

- @

TypeScript - Decoradores

- @
- Asignar metadatos

TypeScript - Decoradores

- @
- Asignar metadatos
- Muy utilizados en Angular

TypeScript - Decoradores

```
import { Component, Input } from '@angular/core'

@Component({
  selector: 'app-factura',
  templateUrl: './factura.component.html',
  styleUrls: ['./factura.component.css']
})
export class FacturaComponent {
  @Input()
  facturaId: number;
}
```

Angular



Primeros pasos

Primeros pasos

- Generar la app:
ng new <nombre-app> --prefix <prefijo>

Primeros pasos

- Generar la app:
ng new <nombre-app> --prefix <prefijo>
- Ejecutar la app y verla en el navegador:
ng serve -o

Primeros pasos

- Generar la app:
ng new <nombre-app> --prefix <prefijo>
- Ejecutar la app y verla en el navegador:
ng serve -o
- Entornos dev y prod

Primeros pasos

- Generar la app:
ng new <nombre-app> --prefix <prefijo>
- Ejecutar la app y verla en el navegador:
ng serve -o
- Entornos dev y prod
- Archivos de configuración
- Módulos: contenedores lógicos
- Componentes
 - Divisiones de la UI
 - Tienen clase y template

Primeros pasos

- Generar la app:
ng new <nombre-app> --prefix <prefijo>
- Ejecutar la app y verla en el navegador:
ng serve -o
- Entornos dev y prod
- Archivos de configuración
- Módulos: contenedores lógicos
- Componentes
 - Divisiones de la UI
 - Tienen clase y template
- Creando piezas:
ng generate <tipo-pieza> <nombre-pieza> [params]

Esqueleto de una pieza en Angular

Esqueleto de una pieza en Angular

- clase =>

Esqueleto de una pieza en Angular

- clase =>
- => clase exportada =>

Esqueleto de una pieza en Angular

- clase =>
- => clase exportada =>
- => clase decorada =>

Esqueleto de una pieza en Angular

- clase =>
- => clase exportada =>
- => clase decorada =>
- => dependencias

Examinando un módulo

Examinando un módulo

- Metadata

Examinando un módulo

- Metadata
 - declarations:
 - componentes, directivas y pipes del módulo

Examinando un módulo

- Metadata
 - declarations:
 - componentes, directivas y pipes del módulo
 - imports:
 - otros módulos cuyos componentes, directivas o pipes exportados queremos usar
 - exports:
 - componentes, directivas o pipes que exponemos para que los usen otros módulos
 - providers:
 - objetos inyectables que están disponibles para el inyector del módulo

Examinando un módulo

- Metadata
 - declarations:
 - componentes, directivas y pipes del módulo
 - imports:
 - otros módulos cuyos componentes, directivas o pipes exportados queremos usar
 - exports:
 - componentes, directivas o pipes que exponemos para que los usen otros módulos
 - providers:
 - objetos inyectables que están disponibles para el inyector del módulo
 - bootstrap:
 - componente(s) inicial de la app

Examinando un componente

Examinando un componente

- Metadata

Examinando un componente

- Metadata
 - selector:
Selector CSS que se corresponde con una etiqueta HTML

Examinando un componente

- Metadata
 - selector:
Selector CSS que se corresponde con una etiqueta HTML
 - template / templateUrl:
String con el HTML / fichero con el HTML

Examinando un componente

- Metadata
 - selector:
Selector CSS que se corresponde con una etiqueta HTML
 - template / templateUrl:
String con el HTML / fichero con el HTML
 - styles / styleUrls:
Strings con los estilos / ficheros con los estilos

Examinando un componente

- Metadata
 - selector:
Selector CSS que se corresponde con una etiqueta HTML
 - template / templateUrl:
String con el HTML / fichero con el HTML
 - styles / styleUrls:
Strings con los estilos / ficheros con los estilos
- ngOnInit
Componente inicializado (con su vista renderizada y sus valores cargados), se usa para los procesos iniciales (no usar el constructor).

Examinando un componente

- Metadata
 - selector:
Selector CSS que se corresponde con una etiqueta HTML
 - template / templateUrl:
String con el HTML / fichero con el HTML
 - styles / styleUrls:
Strings con los estilos / ficheros con los estilos
- ngOnInit
Componente inicializado (con su vista renderizada y sus valores cargados), se usa para los procesos iniciales (no usar el constructor).
- ngOnDestroy

Examinando un template

Examinando un template

- Custom elements

Examinando un template

- Custom elements
- Data binding

Examinando un template

- Custom elements
- Data binding
- Interpolation

Examinando un template

- Custom elements
- Data binding
- Interpolation
- Property binding

Examinando un template

- Custom elements
- Data binding
- Interpolation
- Property binding
- Class & style binding

Examinando un template

- Custom elements
- Data binding
- Interpolation
- Property binding
- Class & style binding
- Event binding

Examinando un template

- Custom elements
- Data binding
- Interpolation
- Property binding
- Class & style binding
- Event binding
- Two-way binding

Examinando un template

Examinando un template

- Directivas de atributo

Examinando un template

- Directivas de atributo
 - ngClass

Examinando un template

- Directivas de atributo
 - ngClass
 - ngStyle

Examinando un template

- Directivas de atributo
 - ngClass
 - ngStyle
- Directivas estructurales

Examinando un template

- Directivas de atributo
 - ngClass
 - ngStyle
- Directivas estructurales
 - ngIf

Examinando un template

- Directivas de atributo
 - ngClass
 - ngStyle
- Directivas estructurales
 - ngIf
 - ngFor

Examinando un template

- Directivas de atributo
 - ngClass
 - ngStyle
- Directivas estructurales
 - ngIf
 - ngFor
 - ngSwitch

Examinando un template

- Directivas de atributo
 - ngClass
 - ngStyle
- Directivas estructurales
 - ngIf
 - ngFor
 - ngSwitch
- Pipes

Examinando un template

- Directivas de atributo
 - ngClass
 - ngStyle
- Directivas estructurales
 - ngIf
 - ngFor
 - ngSwitch
- Pipes
 - @Pipe, PipeTransform

Examinando un template

- Directivas de atributo
 - ngClass
 - ngStyle
- Directivas estructurales
 - ngIf
 - ngFor
 - ngSwitch
- Pipes
 - @Pipe, PipeTransform
- Directivas propias

Examinando un template

- Directivas de atributo
 - ngClass
 - ngStyle
- Directivas estructurales
 - ngIf
 - ngFor
 - ngSwitch
- Pipes
 - @Pipe, PipeTransform
- Directivas propias
 - De atributo (ElementRef.nativeElement)

Examinando un template

- Directivas de atributo
 - ngClass
 - ngStyle
- Directivas estructurales
 - ngIf
 - ngFor
 - ngSwitch
- Pipes
 - @Pipe, PipeTransform
- Directivas propias
 - De atributo (ElementRef.nativeElement)
 - Estructurales (ViewContainerRef y TemplateRef)

Servicios

Servicios

- Dependency Injection:
Proveedores y jerarquía de inyectores

Servicios

- Dependency Injection:
Proveedores y jerarquía de inyectores
- Injectable()

Servicios

- Dependency Injection:
Proveedores y jerarquía de inyectores
- Injectable()
- Singleton: tiene como ámbito su inyector y todos sus inyectores hijos.

Formularios

Formularios

- [(ngModel)]: Two-way binding

Formularios

- [(ngModel)]: Two-way binding
- ngForm, ngModel y ngSubmit

Formularios

- [(ngModel)]: Two-way binding
- ngForm, ngModel y ngSubmit
- Variables de template con #

Formularios

- [(ngModel)]: Two-way binding
- ngForm, ngModel y ngSubmit
- Variables de template con #
- Validaciones: los diferentes estados

Formularios

- [(ngModel)]: Two-way binding
- ngForm, ngModel y ngSubmit
- Variables de template con #
- Validaciones: los diferentes estados
- Resetear los estados

Formularios

- [(ngModel)]: Two-way binding
- ngForm, ngModel y ngSubmit
- Variables de template con #
- Validaciones: los diferentes estados
- Resetear los estados
- Template driven y Reactive forms

Conexiones con el servidor

Conexiones con el servidor

- Asincronía

Conexiones con el servidor

- Asincronía
- Observables

Conexiones con el servidor

- Asincronía
- Observables
- Suscripciones

Conexiones con el servidor

- Asincronía
- Observables
- Suscripciones
- API REST

Conexiones con el servidor

- Asincronía
- Observables
- Suscripciones
- API REST
- El módulo HttpClientModule

Conexiones con el servidor

- Asincronía
- Observables
- Suscripciones
- API REST
- El módulo HttpClientModule
 - Módulo HttpClientModule y servicio HttpClient

Conexiones con el servidor

- Asincronía
- Observables
- Suscripciones
- API REST
- El módulo HttpClientModule
 - Módulo HttpClientModule y servicio HttpClient
 - Métodos del servicio HttpClient:
get(), post(), put(), patch(), delete()

Conexiones con el servidor

- Asincronía
- Observables
- Suscripciones
- API REST
- El módulo HttpClientModule
 - Módulo HttpClientModule y servicio HttpClient
 - Métodos del servicio HttpClient:
get(), post(), put(), patch(), delete()
 - Obteniendo la respuesta completa:
{ observe: 'response' }

Conexiones con el servidor

- Asincronía
- Observables
- Suscripciones
- API REST
- El módulo HttpClientModule
 - Módulo HttpClientModule y servicio HttpClient
 - Métodos del servicio HttpClient:
get(), post(), put(), patch(), delete()
 - Obteniendo la respuesta completa:
{ observe: 'response' }
 - Interceptors y autenticación

Navegación por la app

Navegación por la app

- El router

Navegación por la app

- El router
- El RouterOutlet

Navegación por la app

- El router
- El RouterOutlet
- Las rutas

Navegación por la app

Navegación por la app

- El router

Navegación por la app

- El router
- El RouterOutlet

Navegación por la app

- El router
- El RouterOutlet
- Las rutas
 - Página por defecto

Navegación por la app

- El router
- El RouterOutlet
- Las rutas
 - Página por defecto
 - 404

Navegación por la app

- El router
- El RouterOutlet
- Las rutas
 - Página por defecto
 - 404
 - Parámetros: los observables paramMap y data de ActivatedRoute

Navegación por la app

- El router
- El RouterOutlet
- Las rutas
 - Página por defecto
 - 404
 - Parámetros: los observables paramMap y data de ActivatedRoute
 - Guards y resolvers

Navegación por la app

- El router
- El RouterOutlet
- Las rutas
 - Página por defecto
 - 404
 - Parámetros: los observables paramMap y data de ActivatedRoute
 - Guards y resolvers
- Links de navegación: routerLink y routerLinkActive

Navegación por la app

- El router
- El RouterOutlet
- Las rutas
 - Página por defecto
 - 404
 - Parámetros: los observables paramMap y data de ActivatedRoute
 - Guards y resolvers
- Links de navegación: routerLink y routerLinkActive
- router.navigate()

Navegación por la app

- El router
- El RouterOutlet
- Las rutas
 - Página por defecto
 - 404
 - Parámetros: los observables paramMap y data de ActivatedRoute
 - Guards y resolvers
- Links de navegación: routerLink y routerLinkActive
- router.navigate()
- Lazy loading

Navegación por la app

- El router
- El RouterOutlet
- Las rutas
 - Página por defecto
 - 404
 - Parámetros: los observables paramMap y data de ActivatedRoute
 - Guards y resolvers
- Links de navegación: routerLink y routerLinkActive
- router.navigate()
- Lazy loading
- El servicio Title

Despliegue a producción

Despliegue a producción

- Pruebas con ng build

Despliegue a producción

- Pruebas con ng build
- ng build:

Despliegue a producción

- Pruebas con ng build
- ng build:
 - --prod: código optimizado para producción

Despliegue a producción

- Pruebas con ng build
- ng build:
 - --prod: código optimizado para producción
 - --base-href=: cambia el directorio base

Despliegue a producción

- Pruebas con ng build
- ng build:
 - --prod: código optimizado para producción
 - --base-href=: cambia el directorio base
 - --sourcemaps: genera los source maps

Despliegue a producción

- Pruebas con ng build
- ng build:
 - --prod: código optimizado para producción
 - --base-href=: cambia el directorio base
 - --sourcemaps: genera los source maps
- Entornos propios

Links

- [Documentación oficial de Angular](#)
- [Playground para Angular](#)
- [Documentación de TypeScript](#)
- [Playground para TypeScript](#)
- [Configuración del compilador TypeScript](#)
- [Documentación de Angular CLI](#)
- [Documentación sobre todas las API de JavaScript](#)
- [JSON Server API](#)
- [Tablas de compatibilidad en navegadores](#)
- [Angular en navegadores antiguos](#)

mario.gonzalez@flexired.es