Enrique Oriol

¿Qué es Angular Universal?

Published on 4 diciembre, 2017

Angular Universal es la tecnología que te **permite ejecutar tu aplicación Angular desde el servidor**. El *Server Side Rendering* (SSR) te puede interesar por cuestiones de SEO, para compartir páginas en redes sociales (con su *thumbnail*), o para reducir el tiempo de carga inicial, por ejemplo. ¿Quieres saber más? Sigue leyendo…

Un poco de historia

Renderizar en servidor no es nada nuevo. De hecho, ha sido la tónica predominante hasta hace unos años. Desde PHP a Python, pasando por Java o .NET, el objetivo

Aprende a hacer apps móviles con **Ionic 2** Ver curso

página nueva. En esos tiempos Javascript tenía un uso limitado, para mejorar la UX (*drag & drop*, por ejemplo) y poco más.

Luego llegaron las *Client Side Apps*, donde lo único que enviaba el servidor era una vista `index.html` vacía, con dependencias JS. Cuando recibía el código JS, el navegador empezaba a ejecutar funciones que alteraban el DOM, construyendo la vista desde el cliente.

Las *Client Side Apps* ofrecían una versatilidad y dinamismo nunca visto hasta el momento, con navegación en cliente y concentrando la comunicación con servidor en llamadas a APIs REST. Tanto es así, que hoy en día se han convertido en la tónica habitual de internet (Angular, React...).

¿Tienen inconvenientes las *Client Side Apps*?

Pues sí. Nada es perfecto, y trabajar en cliente tiene sus desventajas. Principalmente dos:

- **Incremento en el tiempo de carga**: hay que descargar y parsear index.html, encontrar las dependencias JS, descargarlas y ejecutarlas para empezar a tener algo en la vista.
- **Problemas con SEO**: Los bots de los buscadores no tienen por qué ejecutar JS. Normalmente lo que encuentran en tu página es lo que les da el servidor, un DOM vacío.

Estos inconvenientes han llevado a la comunidad a diseñar una **estrategia híbrida (*Isomorphic Apps*)**, para solucionar estos problemas.

¿Qué son las aplicaciones isomórficas?

La gracia de usar SSR con Angular (igual que con otras librerías front como React), es que luego puedas "reengancharte" a esa página desde el cliente. De este modo, consigues lo mejor de los dos mundos:

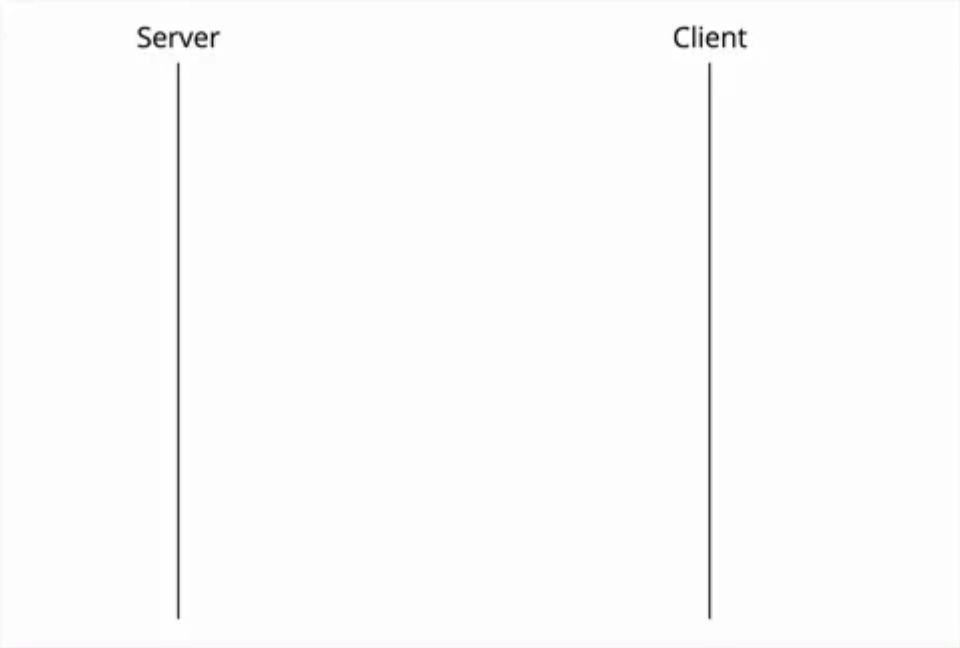
- Una carga inicial rápida gracias al SSR

Aprende a hacer apps móviles con **Ionic 2** **Ver curso** X

Esto es lo que se conoce como **aplicación isomórfica**: webapps que se renderizan en servidor para una ruta de entrada y se navegan en cliente para las interacciones posteriores.

> En las apps isomórficas, el servidor renderiza la ruta de entrada y la navegación posterior se hace en cliente.

A continuación puedes ver una animación con el funcionamiento básico de una web isomórfica.



Como ves, el servidor renderiza la página inicial y se la envía al cliente. El cliente parsea el html recibido en busca de recursos para pedirlos al servidor (incluyendo el JS necesario para lanzar la app en cliente).

El cliente muestra la vista entregada por el servidor y a la que recibe los datos para generar la app desde Javascript (como hacen Angular/React habitualmente), empieza a cargarla en *background*.

Cuando la app está lista, reemplaza la vista entregada por servidor por la vista generada en cliente. Esto último se conoce como proceso de **rehidratación**.

Aprende a hacer apps móviles con **Ionic 2** **Ver curso**

Angular Universal es un proyecto que existe desde Angular 2. Igual me dirás… ¿por qué has esperado hasta Angular 5 para contármelo?

Hasta hace poco, Angular Universal estaba en una fase un poco experimental y entre otras cosas, la rehidratación de la página no estaba muy bien resuelta. El estado de la vista SSR no se traspasaba a la vista cliente y en determinadas condiciones eso producía un ligero parpadeo de la vista.

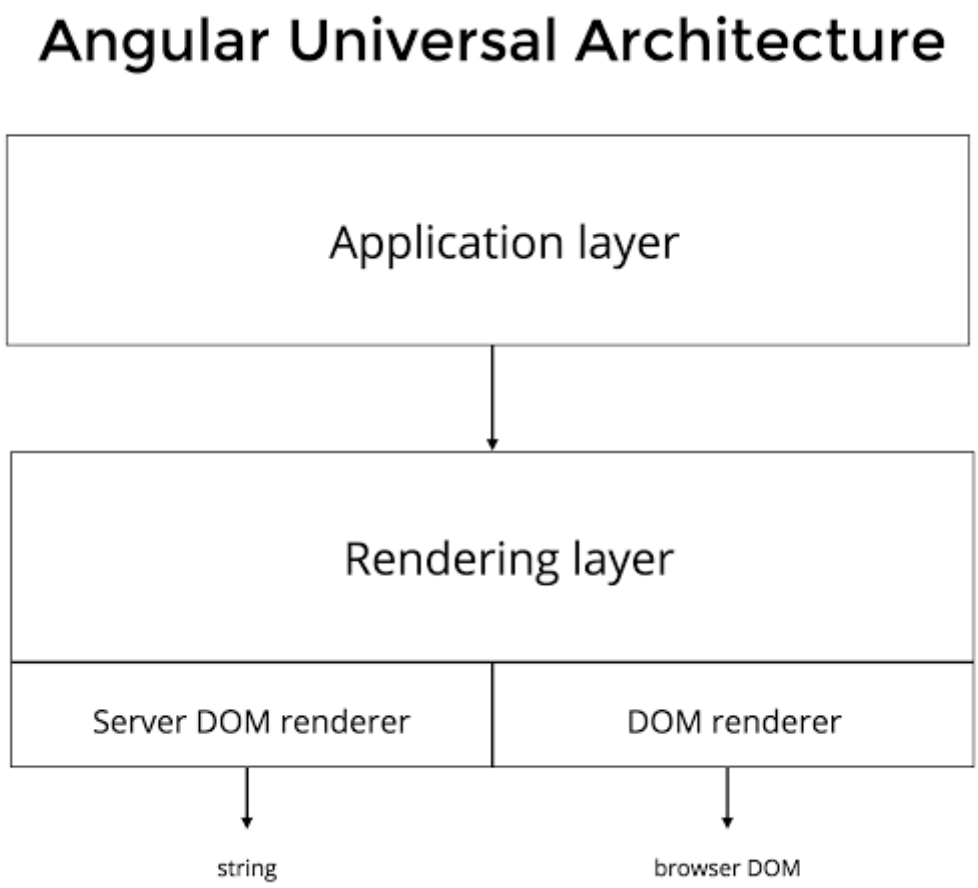
Todos estos problemas han quedado resueltos con Angular 5, gracias al servicio **TransferState** que permite pasar los datos de la web renderizada en servidor al cliente.

Arquitectura

Para ejecutar la app en cliente necesitas tu código habitual de Angular. OK, pero… ¿y la parte servidor?

Angular Universal te da las herramientas para que casi no tengas que tocar tu código: La capa de aplicación se mantiene intacta y en la capa de renderizado se añade un segundo elemento que permite generar el DOM como un string para devolverlo como respuesta a la petición del navegador.

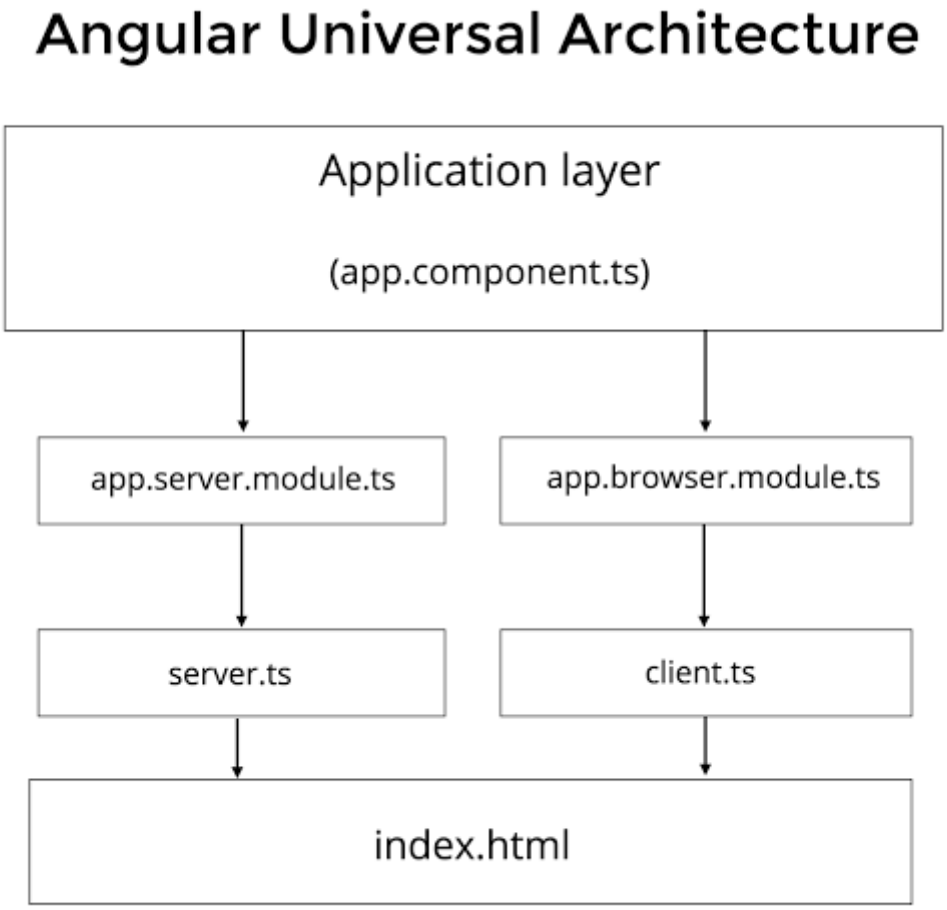
Aprende a hacer apps móviles con **Ionic 2** Ver curso



Quizá este diagrama es demasiado abstracto y lo que quieres saber es cómo te va a afectar realmente a nivel de código.

En ese caso, la respuesta es que a nivel de Angular, simplemente tendrás que añadir un segundo punto de entrada (con su correspondiente módulo), que es el que se ejecutará desde el servidor. El siguiente diagrama es más esclarecedor.

Aprende a hacer apps móviles con **Ionic 2** Ver curso



Por supuesto, esto te lleva a otro tema. Si tienes 2 puntos de entrada, tendrás que generar 2 bundles distintos a nivel de webpack, uno para ejecutar en el servidor, y otro para ejecutar en local.

Ejecución en servidor

Por supuesto, una app isomórfica implica un servidor que escuche peticiones del cliente y le devuelva vistas renderizadas.

No basta con servir solo la "*home*" de tu web app. Seguramente tendrás más vistas a las que el usuario puede acceder directamente poniendo la URL en el navegador, y tu querrás servirles la página apropiada.

A pesar de que Angular Universal soporta .NET y está trabajando en dar soporte a servidores Java, PHP y Python, el soporte principal es, lógicamente, para NodeJS.

Aprende a hacer apps móviles con **Ionic 2** Ver curso

Angular proporciona adaptadores para servidores Node como Express o Hapi.js, de modo que sea extremadamente simple renderizar y devolver los componentes Angular adecuados para una ruta de entrada.

Coming soon…

En mi próximo artículo entraré en código, publicando el detalle paso a paso de cómo lanzar una app Angular usando Angular Universal. ¿Podrás esperar hasta el momento, o vas a mirarlo por tu cuenta?

Si te ha gustado este artículo, compártelo 😉

Compártelo:

Relacionado

¿Angular 2 o Angular 4? - Simplemente Angular
22 marzo, 2017
En "Angular 2"

UPDATED - Intro a Angular (parte I): Modulo, Componente, Template y Metadatos
17 marzo, 2017
En "Angular 2"

Introducción a Angular 2 (parte I) - Modulo, Componente, Template y Metadatos
30 junio, 2016
En "Angular 2"

Published in Angular Javascript

Previous Post
HttpClient VS Http – El nuevo servicio http de Angular

No Newer Posts
Return to Blog

Aprende a hacer apps móviles con **Ionic 2** Ver curso

Be First to Comment

Deja un comentario

Introduce aquí tu comentario...

Author Theme modified by Enrique Oriol

Aprende a hacer apps móviles con **Ionic 2** Ver curso