TEORIA PEC 1
Introducción al desarrollo front-end

Desarrollo front-end con frameworks Javascript

Máster Universitario en Desarrollo de sitios y aplicaciones web

Estudios de Informática, Multimedia y Telecomunicación

Introducción

La web como plataforma para el desarrollo de software ha evolucionado de forma drástica desde su origen, especialmente en los últimos años, cuando ha sufrido importantes cambios causados por diversos factores.

Simplemente fijándonos en los términos empleados para describir las principales tareas de un proyecto web hace algunos años y en la actualidad, se puede entender el cambio que estamos experimentando. Un salto que nos lleva del diseño web al desarrollo web. Hace no mucho tiempo era habitual identificar el perfil de quienes creaban contenido para la web bajo la etiqueta de «diseñadores», pero hoy en día la complejidad y cantidad de disciplinas que han ido desarrollándose en torno a esta plataforma han convertido ese término en una pieza más de un complejo engranaje.

Diseño, usabilidad, experiencia de usuario, contenido, marketing, maquetación, desarrollo y programación, SEO... son roles habituales hoy en día en cualquier proyecto web.

Las páginas web han pasado de basarse principalmente en un lenguaje de marcado con un fuerte carácter estático (p.ej. HTML + CSS) a basarse en complejos desarrollos de software interconectados con una alta dependencia de la programación.

Los retos y problemas que nos encontramos también sirven como indicador de esta evolución. Donde antiguamente se dedicaba la mayor parte de los recursos a lograr la compatibilidad con un número muy reducido de navegadores (aunque muy inconsistentes), ahora se emplean fundamentalmente en programar auténticas y completas aplicaciones basadas en JavaScript/TypeScript, que han de superar algunos retos especialmente difíciles, como son la usabilidad, accesibilidad, compatibilidad con navegadores, rendimiento tanto de ejecución como del tamaño del proyecto a desplegarse.

TEORÍA

1. Definición de desarrollo front-end

Es importante aclarar, en este punto que siempre que nos refiramos al término «desarrollo front-end» en esta asignatura lo haremos en el contexto de lo que se entiende como tecnologías web, para las que tomamos como referencia la visión ofrecida por Mozilla (creador original de JavaScript).

El término front-end en la actualidad no está ligado solamente a una tarea o lenguaje de programación dentro del proceso de un proyecto web; de hecho, ni tan siquiera está ligado solamente a la web como plataforma, ya que la adaptación de las tecnologías y lenguajes web más allá de Internet ha provocado su inclusión en desarrollos para infinidad de dispositivos como *smartphones*, televisores, consolas e incluso *wearables* (p.ej. *smartwatches*).

Definición desarrollo front-end

Dentro del ámbito de las tecnologías web y en su visión más amplia, este concepto haría alusión a todos los aspectos relativos al desarrollo de la parte visual de una aplicación donde se presentan una serie de contenidos y se crean mecanismos para propiciar la adecuada interacción con el usuario, a la par que la comunicación con el lado de servidor.

Hablando en el contexto de un modelo cliente-servidor, y entendiendo el desarrollo back-end como todo aquél que tiene lugar del lado de servidor con el objetivo de gestionar los datos y la lógica del negocio, el desarrollo front-end sería el resto del trabajo.

2. Principales causas de la evolución del desarrollo front-end

Esta evolución que ha sufrido el desarrollo front-end no se puede entender correctamente sin vincularla a la misma evolución que ha tenido internet, y el impacto que ha llegado a producir en nuestra vida cotidiana.

2.1. Infraestructuras

Un elemento que ha permitido evolucionar las tecnologías web de manera notable ha sido la evolución de las infraestructuras en las que se basa Internet. Mejores sistemas y conectividad, mayor velocidad en el acceso a la información, etc., son factores clave para entender en su conjunto la relevancia que ha adquirido Internet y con ella las tecnologías web.

2.2. Tecnología

La vertiginosa evolución tecnológica que hemos sufrido en los últimos años, con la continua aparición de nuevos dispositivos y aplicaciones donde está muy presente la conexión a Internet, han colaborado en el auge de las tecnologías web y, por ende, del desarrollo front-end.

Hemos alcanzado el punto en el que un *smartphone* supera en prestaciones a grandes equipos de sobremesa de hace escasos años, en el que una prenda de vestir nos ofrece y recopila información útil o en el que un microordenador conectado a diversos sensores nos abre un abanico infinito de aplicaciones prácticas.

2.3. Técnica

La madurez y evolución de la tecnología requiere de mejoras técnicas que saquen el máximo partido de ella. Se ha acompañado la evolución tecnológica con importantes avances en el terreno del desarrollo y especialmente en el desarrollo front-end (por la creciente cantidad de aplicaciones que se le dan).

2.4. Cultura, sociedad y economía

El uso intensivo y globalizado de Internet, su incursión en nuestras vidas incluso desde edades muy tempranas en las generaciones actuales, han acentuado los factores citados anteriormente, e Internet ha sido la base sobre la que crear una gran industria, cuyo negocio arrastra importantes inversiones que cierran el círculo para fomentar esa evolución.

3. Aplicaciones actuales del desarrollo front-end

Como se ha indicado en el apartado anterior, hoy en día la adopción de las tecnologías web como alternativa al desarrollo de software nativo en diversas plataformas ha potenciado el concepto de desarrollo front-end, así como las disciplinas, tecnologías y lenguajes vinculados a él.

En ocasiones incluso podemos encontrar plataformas donde las tecnologías web se han apoderado de la etiqueta de «nativo», habiendo sido concebidas bajo esa característica.

Todo ello nos lleva a una importante diversidad en los campos de aplicación del desarrollo front-end.

3.1. Desarrollo de sitios web

Obviamente, la raíz original para el que se aplicaba el desarrollo front-end sigue siendo, hoy en día, una de las aplicaciones más habituales del mismo: la creación de páginas/sitios web.

Si bien el objetivo final es el mismo (una presencia en Internet), el trabajo se ha profesionalizado notablemente y ha llevado a la necesidad de aumentar la especialización de los perfiles, por el aumento de la complejidad de este tipo de proyectos a causa de un mayor número de funcionalidades disponibles en los navegadores, un mayor número de dispositivos donde visualizarse y un

mayor número de herramientas, *frameworks*, librerías y lenguajes de programación que nos permiten trabajar con este objetivo.

3.2. Desarrollo de aplicaciones web

Más allá de la creación de sitios web, en la actualidad los navegadores se han consolidado como plataforma sobre la que desplegar aplicaciones, que cada día se acercan más a las de escritorio, pero sumando las ventajas que pueden ofrecer las tecnologías web y el acceso desde cualquier lugar y equipo, siempre que se disponga de conexión a Internet.

Si bien las herramientas, tecnologías, lenguajes, etc. pueden ser muy similares a los empleados para crear páginas web, este tipo de aplicaciones se caracterizan por tener un peso mucho mayor en la parte de programación JavaScript/TypeScript que en la de maquetación, incluyendo la comunicación con el servidor, que es prácticamente inevitable hoy en día.

Además, al ser su entorno de ejecución un navegador, son aptas para, si se desea, ser adaptadas a teléfonos inteligentes, tabletas, televisores, etc., de manera que sean accesibles y usables en este tipo de dispositivos.

3.3. Desarrollo de aplicaciones para smartphones y tablets

Al igual que se pueden crear aplicaciones con tecnologías web para el navegador, existen soluciones que nos permiten crear una app para *smartphones* y *tablets*, rompiendo la barrera impuesta por los navegadores y optando a capacidades nativas de los sistemas operativos de dichos dispositivos.

Estas aplicaciones son comparables en muchos escenarios a las aplicaciones nativas (i.e. app en Android y iOS) puesto que se utilizan los sensores propios de los dispositivos a través de bibliotecas/frameworks web tales como Phonegap/Apache Cordova, Ionic o NativeScript..

3.4. Desarrollo de aplicaciones smart TV

Uno de los entornos en los que se han adoptado las tecnologías web como base del desarrollo de aplicaciones nativas es el de las smartTV, donde las aplicaciones basadas en HTML5 son la principal y más generalizada opción de desarrollo. Una demostración de este hecho es que Smart TV Alliance, tal y como indica en sus FAQ, ha elegido las tecnologías web como principal herramienta para el desarrollo y soporte de aplicaciones, incluso extendiendo las capacidades sobre los navegadores convencionales en algunas áreas (como *streaming*, DRM, etc.).

Smart TV Alliance

Es una de las principales organizaciones apoyada por importantes fabricantes y proveedores para la evolución y estandarización de esta plataforma. Bajo su paraguas actúan agentes del sector tan relevantes como LG, Philips, Panasonic, TOSHIBA, IBM...

Uno de los retos particulares que se da en este entorno es el hecho de trabajar para pantallas especialmente grandes y con mecanismos de interacción por parte del usuario no tan precisos y ágiles como pueden ser los ratones, *trackpads* etc.

3.5. Desarrollo para otros dispositivos

En la actualidad, el uso de Internet se ha trasladado mucho más allá de dispositivos como todos los que hemos citado anteriormente, originándose el término «IoT» (Internet de las cosas; *Internet of Things*), para aglutinar un sinfín de dispositivos y aplicaciones que se relacionan directamente con objetos físicos que, habitualmente, en combinación con sensores u otro tipo de información, mejoran su funcionalidad.

Dentro de las aplicaciones englobadas en lo que se conoce como IoT, y basándose en el entorno en el que son aplicadas, se podría realizar la siguiente categorización (fuente: https://en.wikipedia.org/wiki/Internet_of_Things):

- Smart wearable
- Smart home
- Smart city
- Smart environment
- Smart Enterprise

Como se puede observar, los campos de aplicación son totalmente variados, englobando prendas de vestir/complementos, sistemas para el hogar, sistemas para dotar de inteligencia a las ciudades, al entorno y las empresas.

Es notable destacar que este campo ha captado la atención de las principales empresas tecnológicas a escala mundial como Google, Intel, IBM, Cisco, Siemens, etc., existiendo estudios que estiman que el crecimiento de este tipo de dispositivos conectados superará a los teléfonos móviles en el año 2020.

3.6. Desarrollo de aplicaciones de servidor

Para terminar, no podemos dejar de lado un aspecto que ha cambiado, en los últimos años, la manera de ver los lenguajes utilizados históricamente para el desarrollo front-end, que es el salto de JavaScript al lado del servidor, es decir, al lado back-end. Este salto se ha logrado principalmente gracias a Node.js. Curiosamente, esta aproximación no es nueva, ya que a finales de 1994 Netscape ya lanzó algo similar con su Netscape Enterprise Server.

Sobre Node.js

Creado en el año 2009 por Ryan Dahl, y apadrinado por la empresa Joyent, Node.js es un *runtime* multiplataforma, de código abierto, basado en ECMASCript (el mismo estándar que emplea JavaScript) y construido sobre la base del motor de JavaScript V8 de Chrome, que emplea un modelo de entrada y salida asíncrono basado en eventos.

Su concepción fue orientada a la creación de aplicaciones de red con un alto grado de escalabilidad, entre los que obviamente se encuentran los servidores.

Más información en https://nodejs.org

Más allá del hecho de poder disponer de un servidor programado en JavaScript, la auténtica revolución en el uso de Node.js ha surgido con su gestor de paquetes (npm) que, como sus creadores indican, en la actualidad se ha convertido en el mayor ecosistema de librerías open source del mundo.

Sobre npm

npm es un gestor de paquetes o librerías para Node.js que se utiliza para el desarrollo front-end y básicamente para el desarrollo basado en JavaScript. npm cuenta con más de 200.000 paquetes o librerías y ronda en la actualidad los cien millones de descargas al día, lo que nos puede dar una clara visión del impacto que ha tenido este ecosistema en el uso diario de la comunidad de desarrolladores web, y más concretamente en el desarrollo front-end.

Más información en https://www.npmjs.com/

Algunos de los paquetes más empleados de npm nos permiten trabajar de una forma ágil y sencilla con importantes herramientas front-end, como, por ejemplo: preprocesadores CSS (SASS, LESS, Stylus), gestores de tareas automatizadas (Gulp, Grunt), optimizadores de imágenes (imagemin), generadores de proyectos (yeoman), sistemas de testeo automatizado (jasmine, jest), previsualización en el navegador (live-server), etc.

3.7. Desarrollo de aplicaciones de escritorio

Las aplicaciones de escritorio no han quedado fuera del alcance de las tecnologías relacionadas con el desarrollo front-end, y en la actualidad existen multitud de proyectos que trasladan las ventajas del desarrollo con HTML5, CSS y JavaScript a entornos como los escritorios de los sistemas operativos con muchas más funcionalidades y capacidades que las que habitualmente se disponen en el navegador.

Entre estas alternativas no solamente se incluyen software específico de empresas, sino que muchas empresas desarrolladoras de navegadores y editores han aportado su granito de arena a este objetivo. Éste es el caso de Electron, una de las soluciones *open source más* reconocidas. Electron permite construir aplicaciones de escritorio utilizando tecnologías Web (HTML, CSS, JavaScript/TypeScript). Con esta solución se han creado algunas aplicaciones reconocidas como Slack, GitBook, Visual Studio Code o el propio editor Atom. Soporta aplicaciones para Windows, Mac y Linux.

Otro ejemplo es Node WebKit. Nacido del “node-webkit project” de Intel y basado en la combinación de Chromium y Node.js, Node WebKit es una solución que nos permite escribir aplicaciones nativas con tecnologías web. Cuenta con aplicaciones de reconocido prestigio realizadas con este sistema como: WhatsApp o Messenger de Facebook para escritorio, Intel XDK. Soporta desarrollo de aplicaciones para Windows, Mac y Linux.

4. Front-end Roadmap

En la web https://roadmap.sh/frontend se detalla el camino que debe tomar un desarrollador front-end en el aprendizaje de tecnologías. A continuación, repasaremos las tecnologías fundamentales:

4.1. HTML/CSS/Básico de JavaScript

Inicialmente es fundamental que se aprenda HTML semántico, conceptos de SEO y accesibilidad en la estructuración de HTML, se construyan *layouts* básicos con CSS 3, así como el uso de `MediaQueries` para crear páginas *responsives*.

En cuanto al nivel de JavaScript que se requiere inicialmente, es aquél que es utilizado para crear un dinamismo básico en una página Web, pero no como lenguaje para crear aplicaciones de gran envergadura. Para ello es necesario conocer la sintaxis del lenguaje y características de ES6+ y Javascript modular, así como el manejo del DOM (*Document Object Model*) para crear dinamismo en la estructura HTML en tiempo de ejecución. Aprender cómo usar una API de Ajax como puede ser `XHR` o fetch para conectarse con un back-end o realizar cargas de documentos de manera dinámica. En este momento es cuando se aprenden conceptos de JavaScript fundamentales como son: `Hoisting`, `Event Bubbling`, `Scope`, `Prototype`, `Shadow DOM` y cómo funcionan los navegadores web, el protocolo HTTP/S, las DNS, etc.

4.2. Uso de gestores de paquetes

`npm` y `yarn` facilitan la instalación y configuración de bibliotecas de terceros, y es aquí cuando comienza el desarrollo de gran envergadura de una aplicación. `npm` y `yarn` surgen como repositorios/gestores de librerías, similar a los repositorios de paquetes en las distribuciones Linux. Es aquí cuando se pueden tomar dos caminos claramente diferenciados (pero no excluyentes) a la hora de profundizar en el desarrollo front-end: **Diseño o Desarrollo**.

- **Diseño**: si se toma este camino se deben desarrollar estilos más avanzados haciendo uso de pre-procesadores de CSS como **SASS**, **PostCSS** o **Less** que permitan crear la estructuración de la aplicación web a un nivel de orden mayor, llegando hasta las arquitecturas de CSS como serían **BEM, OOCSS** o **SMACSS**.

- **Desarrollo**: seleccionar un *framework* de trabajo y crecer en base a él, ya sea **Angular** o **Vue**, o la biblioteca de vista **React** que puede ser complementada con diferentes bibliotecas para armar un *framework* de igual características (i.e. igual de potente) que los dos anteriores.

4.3. Frameworks CSS

En esta parte del camino es imprescindible, tanto para los desarrolladores como diseñadores, el conocer *frameworks* de CSS como son **Bootstrap** y **Materialize CSS**. Estos *frameworks*, dado que facilitan la creación de multitud de componentes visuales (p.ej. tablas, desplegables, menús, etc.) son ampliamente utilizados en los diseños de infinidad de sitios webs.

4.4. Linters y formatters

Tanto si se trabaja en equipo o solo es imprescindible manejar herramientas que normalicen el código fuente que estamos generando. Para ello se pueden utilizar herramientas como **Prettier, ESLint o TSLint**.

4.5. Herramientas de gestión de módulos

JavaScript nació sin un modelo de módulos nativo, de ahí que haya herramientas, tales como **Webpack, Parcel y Rollup**, que han facilitado esta tarea, aunque tienen una cierta complejidad por sí mismas.

4.6. Uso de un lenguaje tipado

JavaScript carece de comprobación de tipos, lo cual es ventajoso cuando se desarrolla en lenguajes interpretados en los que no se desarrollan aplicaciones de gran complejidad. Pero si JavaScript se utiliza para crear aplicaciones de gran envergadura puede ser muy complicado mantener el control, de ahí que surjan los lenguajes como **TypeScript** y **Flow**. Estos lenguajes se construyen sobre JavaScript y son utilizados en la fase de desarrollo, puesto que, para la fase de producción, se deberán convertir los ficheros generados con los lenguajes tipados a JavaScript, que es el lenguaje que interpretan correctamente los navegadores.

4.7. Testing

Hoy en día desarrollar *software* sin realizar pruebas es un error tan grave como desarrollar *software* sin herramientas de depuración adecuadas. En el desarrollo front-end se desarrollan tests a nivel unitario, integración y E2E (*end-to-end*). En el marco de pruebas unitarias y de integración existen dos herramientas líderes: **Jasmine** y su *fork* **Jest**. En cambio, en pruebas E2E existen herramientas como `Selenium`, `Cypress` o el propio `Jest`.

4.8. Programación reactiva

La evolución de los sistemas asíncronos en el desarrollo front-end ha sido el siguiente: *callbacks*, promesas, *async/await* y la llegada de observables. **RxJS** es la biblioteca de facto en JavaScript sobre la que se construye la programación reactiva. Esto permite manejar flujos de datos en lugar de datos

simples de manera asíncrona y no solamente en conexiones con el servidor (back-end), sino también entre elementos de nuestra aplicación front-end.

4.9. Gestión del estado

Cuando la aplicación es de gran envergadura y estamos en el contexto de una aplicación web, necesitamos controlar el estado de la aplicación. Aquí surge el patrón de flujo de datos REDUX a partir de FLUX e implementaciones de los *frameworks* front-end como son VueX, NgRX y Redux (React) que permiten la gestión del estado.

Ver también

Existen caminos similares para back-end y DevOps (ver https://roadmap.sh/).

5. Repaso a ES5

NOTA: Este apartado es un resumen extraído de https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript .

JavaScript es un lenguaje de programación que te permite realizar actividades complejas en una página web — cada vez hay más páginas web que hacen más cosas que sólo mostrar información estática — como mostrar actualizaciones de contenido en el momento, interactuar con mapas, animaciones gráficas 2D/3D etc. Puedes estar seguro que JavaScript está involucrado en todo esto. Es la tercera capa del pastel de los estándares en las tecnologías para la web, dos de las cuales son **HTML** y **CSS**.

- HTML es un lenguaje de marcado que usa la estructura para dar un sentido al contenido web, por ejemplo, define párrafos, cabeceras, tablas, imágenes y vídeos en la página.

- CSS es un lenguaje de reglas en cascada que usamos para aplicar un estilo a nuestro contenido en HTML, por ejemplo, colocando colores de fondo, fuentes y marginando nuestro contenido en múltiples columnas.

- JavaScript es un lenguaje de programación que te permite crear contenido nuevo y dinámico, controlar archivos multimedia, crear imágenes animadas y muchas otras cosas más.

Las 3 capas se complementan una con la otra.

El núcleo de JavaScript consiste en características comunes de programación que te permiten hacer cosas como:

- Almacenar valores útiles dentro de variables.
- Realizar operaciones con trozos de texto (i.e. `String`).
- Hacer funcionar código en respuesta a algunos eventos que están ocurriendo en la página web. Por ejemplo, usamos un evento *click* para detectar cuándo un botón es cliqueado para luego hacer correr el código que actualice la etiqueta de texto.
- Etc.

Pero lo que es más emocionante es la adición de funcionalidades construidas como una capa superior del lado cliente del lenguaje JavaScript. Estas funcionalidades se presentan en APIs, i.e. *Application Programming Interfaces*. Las APIs te proporcionan funcionalidades extra que puedes usar en tu código JavaScript. En este sentido, existen las *Browser APIs*, que son APIs construidas dentro de tu navegador web y que son capaces de hacer cosas complejas. Por ejemplo:

- El **DOM** (Modelo de Objeto de Documento) **API** te permite manipular, crear, remover y cambiar códigos escritos en lenguaje HTML y CSS, incluso aplicando dinámicamente nuevos estilos a tu página web, etc. Cada vez que, por ejemplo, aparezca un aviso (popup) en una página web o se muestre nuevo contenido, la DOM API actúa.

- La **Gelocation API** recupera información geográfica. Así es como Google Maps es capaz de encontrar tu ubicación y mostrarla en un mapa.

- La **Canvas** y **WebGL APIs** permiten crear gráficos 2D y 3D animados.

- APIs de Audio y Vídeo como **HTMLMediaElement** y **WebRTC** permite hacer cosas realmente interesantes con multimedia, p.ej. escuchar un audio o un vídeo, grabar un vídeo con tu webcam, etc.

También existen APIs de terceros que no son construidas e incluidas por defecto en el navegador, pero que se pueden usar importándolas en tu proyecto web. Éste el caso de las APIs de Twitter, Google Maps, OpenStreetMap, etc.

Una vez presentadas las virtudes de JavaScript, se debe realizar un repaso exhaustivo de este lenguaje, comenzando con ES5 (versión de 2009). Para ello tendremos que hacer uso de este recurso https://learning.oreilly.com/library/view/javascript-the-definitive/9781491952016/ y leer los siguientes apartados:

- Types, Values and Variables
- Objects
- Arrays
- Functions
- Classes
- Asynchronous JavaScript

NOTA: Como miembros de la Comunidad UOC, tenéis disponible un gran catálogo de recursos de la editorial O'Reilly. Para acceder debéis seguir los siguientes pasos:

* Acceder a https://learning.oreilly.com

* Cuando te pida el *login* y *password*, escribir en el *campo Email Address or Username* vuestro correo electrónico de la UOC, p.ej. dgarciaso@uoc.edu

* Después pulsáis fuera con el ratón y desaparecerá el campo *password*. Además el botón de *Sign In* cambiará por un botón rojo con el texto *Sign In with Single Sign On*.

* A partir de aquí seguid los pasos que os vaya pidiendo, seguramente os llevará a una página de autenticación con el logo de la UOC. Además, os llegará un e-mail a vuestra cuenta de correo electrónico de la UOC avisando de que os habéis suscrito a este catálogo.

* Una vez tengáis acceso al catálogo de O'Reilly, para ver el contenido de los enlaces que os proporcionamos, sólo tenéis que estar autenticados en O'Reilly y abrir una pestaña nueva en vuestro navegador y copiar el enlace que os facilitamos.