< Blog

123

# Ruta de aprendizaje para hacer desarrollo de páginas web

platziteam 9499 Puntos hace 2 años

**El desarrollo web es una de las carreras mejor pagadas, con mayor demanda y de más rápido aprendizaje**. En esta guía vamos a estudiar las herramientas, conceptos, lenguajes más importantes para este 2020 y todo lo que debes aprender para convertirte en un desarrollador de páginas web como frontend profesional.

## Frontend vs. Backend

El frontend es la parte de una aplicación web que interactúa con los usuarios, la construimos usando HTML, CSS y JavaScript. El backend es la parte que los usuarios no pueden ver, donde sucede la lógica y almacenamiento de información. Podemos usar casi cualquier lenguaje de programación: Python, Node.js, Java, Go, Ruby, C#...

Los usuarios escriben sus contraseñas en el frontend. Pero, por seguridad, el frontend no debe guardar ni validar este tipo de información, ya que cualquier persona podría interceptar la información privada de todos nuestros usuarios

123

Share

Para evitar ese problema hay que enviar los datos al backend. Allí los almacenamos en bases de datos y verificamos si la contraseña es correcta. Y luego el backend se lo indica al frontend para que pueda volver a comunicarse con el usuario. Tener frontend y backend es la forma de construir aplicaciones amigables con los usuarios y sin problemas obvios de seguridad.

Recuerda que como desarrolladora frontend profesional siempre debes aprender sobre los diferentes procesos, equipos y tecnologías que afectan a nuestra aplicación. **No le tengas miedo a aprender backend para crecer como frontend developer.**

## Maquetación con HTML

HTML es el lenguaje que usamos para definir la estructura de una página web. Está compuesto de etiquetas para definir las tareas de cada uno de nuestros elementos (párrafos, links, imágenes...).

Además, podemos definir diferentes atributos en cada etiqueta para especificar aún más el comportamiento que necesitamos.

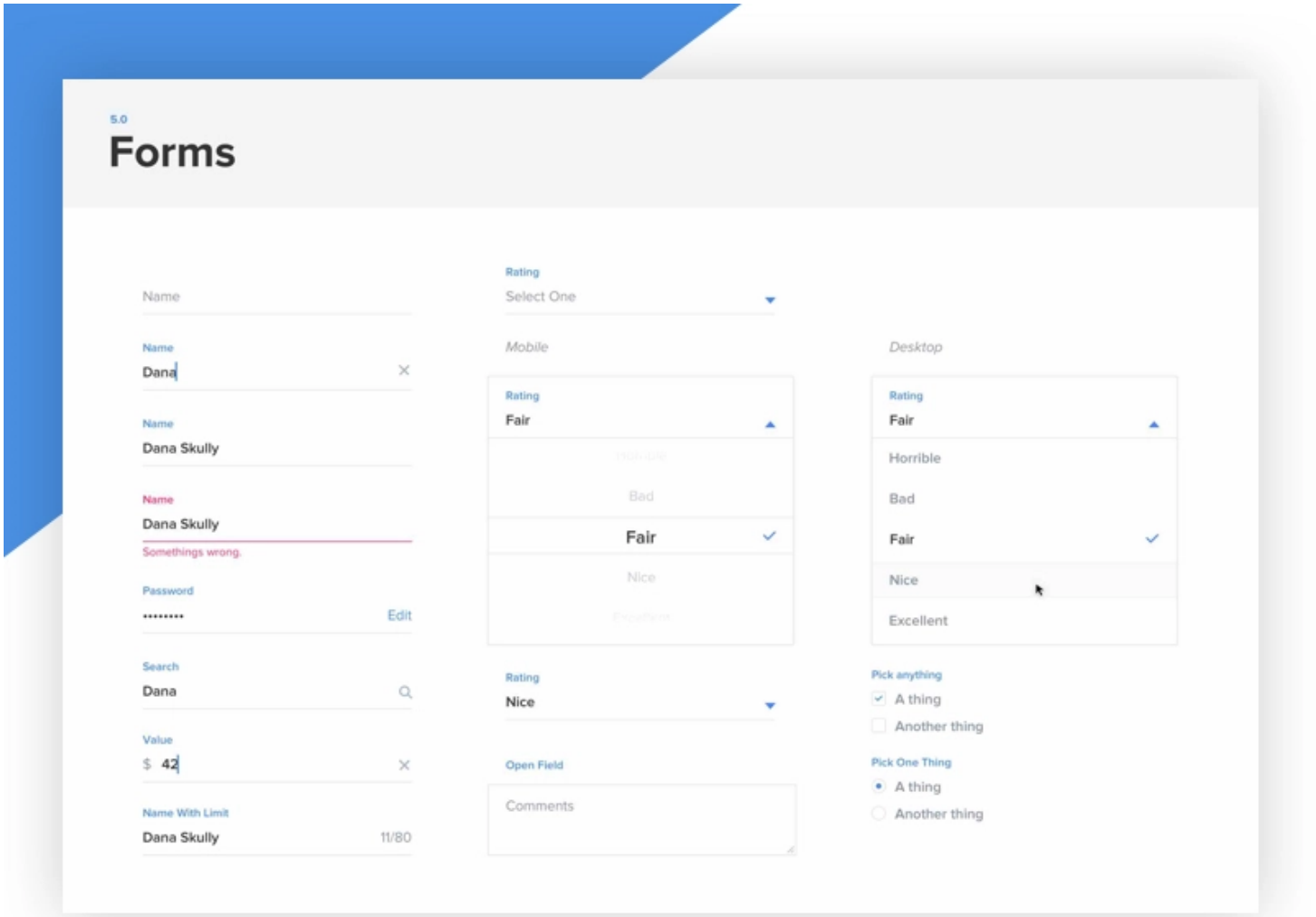
Por ejemplo, el atributo `href` en la etiqueta `a` define la ruta a donde apuntan nuestros enlaces. Y el atributo `src` en la etiqueta `img` define la ruta de la imagen que vamos a mostrar.

```
&lt;p&gt;Esto es un párrafo&lt;/p&gt;

&lt;a href=&quot;https://platzi.com/html&quot;&gt;Esto es un enlace&lt;/a&gt;

&lt;img src=&quot;https://static.platzi.com/media/landing-projects/imagen-proye
```

Los formularios son uno los elementos más importantes para cualquier sitio web




---

- **HTML en la era de Internet Explorer**

Una gran característica de HTML es que funciona a pesar de que cometamos cierto tipo de errores. Esto sucedió por culpa de Internet Explorer, el ex navegador oficial de Microsoft que quería matar al resto de navegadores y por eso rompía los estándares de la web todo el tiempo.

Entonces, los navegadores modernos (Chrome, Firefox, Safari) decidieron crear sistemas aprueba de fallos para que los sitios rotos de la época empezaran a

Internet Explorer murió. Microsoft es consiente de que era terrible y su nuevo navegador oficial es Microsoft Edge (y funciona muy bien). Ahora todos nos adaptamos principalmente a Google Chrome.

---

- **HTML Semántico y Optimizado para SEO**

La mayoría de sitios web tienen una estructura muy desordenada. Pero en HTML cada etiqueta tiene una función muy bien definida. Siempre debemos buscar la etiqueta más adecuada para agrupar o mostrar el contenido a nuestros usuarios.

Esto lo conocemos como HTML Semántico. Y es una de las mejores prácticas que una desarrolladora frontend debe aprender.

Además, también debemos evaluar la forma en que distribuimos el contenido. Buscamos y configuramos las mejores palabras clave usando diferentes meta etiquetas y algunas otras estrategias para que nuestro sitio web aparezca en los primeros resultados de Google.



## Maquetación con CSS

HTML inicialmente también se encargaba de los estilos de una página web. Conseguir los resultados que necesitábamos era muy doloroso, debíamos usar tablas combinadas con imágenes y diferentes estrategias super complicadas.

definir algunos atributos (casi siempre son class y id) para que CSS pueda encontrar nuestros elementos y darles estilo.

```
HTML:

&lt;header id=&quot;header&quot;&gt;
  &lt;h1 class=&quot;header-title&quot;&gt;Platzi&lt;/h1&gt;
&lt;/header&gt;

CSS:

#header {
  background: darkgreen;
}
#header .header-title {
  color: white;
}
```

---

### Modelo de Caja en CSS

Lo más importante para aprender CSS es aprender a pensar en cajas. Nuestros elementos pueden o no comportarse como cajas. Y de esto depende el lugar donde se posicionarán automáticamente y qué código necesitamos para acomodarlos donde realmente necesitamos.

Aprende más en las clases Modelo de caja y Tipos de Display en el **Curso Definitivo de HTML y CSS**.

---

### Responsive Design, Flexbox y CSS Grid

El responsive design consiste en construir aplicaciones web que funcionan bien sin importar el tamaño del dispositivo de los usuarios.

Hace algunos años era una buena característica para nuestras aplicaciones, pero

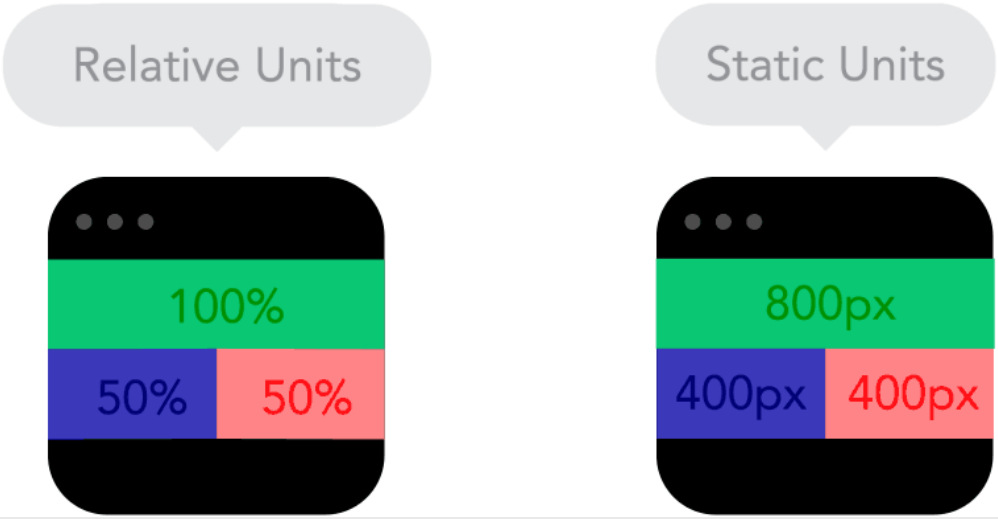
Esto lo logramos con **media queries**. Instrucciones de CSS que añaden nuevos estilos a nuestros elementos dependiendo del tamaño de la pantalla donde está corriendo la aplicación.

```
#hero {
  background-color: green;
}

@media (min-width: 768px) {
  #hero {
    background-color: darkgreen;
  }
}
```

Pero no solo se trata de definir estilos estáticos diferentes para cada tamaño. También existen propiedades de CSS que nos permiten ajustar automáticamente nuestro contenido. Es decir, que su tamaño y posición dependa directamente del tamaño de nuestros dispositivos.

Entre estas propiedades se encuentran **Flexbox** y **CSS Grid**. Herramientas que nos permiten dividir la pantalla de nuestros usuarios por filas y columnas para que acomodar nuestros elementos sea mucho más fácil.



> Recuerda que Flexbox y CSS Grid no son competencia, sino herramientas que trabajan muy bien en conjunto.

---

- **Bootstrap**, **Foundation** y **Materialize**

**Bootstrap, Materialize y Foundation** son herramientas con estilos predefinidos para componentes muy comunes en la mayoría de aplicaciones web. Solo debemos incluir sus estilos y seguir algunas reglas para conseguir resultados muy aceptables. Aunque, por supuesto, luego necesitamos personalizar un poco.

Aprender estas herramientas es muy importante. Pueden ayudarnos bastante con los estilos generales de aplicaciones web muy grandes. Muchas empresas eligen etas herramientas por eso.

Pero recuerda que aprender a construir estilos 100% es un requisito para cualquier desarrollador frontend. Lo más importante al aprender estas herramientas es no depender o limitarnos a ellas.

---

- **Optimización de rendimiento web**

La **Optimización de Carga de Sitios Web** consiste en mejorar el tiempo que tardan nuestras aplicaciones en darle respuestas a los usuarios. Además de ayudar a mejorar la experiencia de nuestros usuarios, también ayuda a mejorar nuestro posicionamiento en buscadores.

Por ejemplo, en vez de cargar nuestros íconos uno por uno, podemos descargar una sola imagen muy grande con todos los íconos y mostrar solo la parte que necesitamos en cada ícono usando CSS.

Esta estrategia se llama **sprites**. Y hay muchas otras que podemos aprender para

- **Fundamentos de Diseño** / **Diseño de Interfaces**

Las herramientas de diseño como Photoshop, Illustrator, Sketch y Zeplin son extremadamente utilizadas en el desarrollo frontend. Con ellas encontramos las medidas, colores, posiciones y demás datos con los que transformaremos estos diseños en páginas web de verdad.

Pero no solo se trata de saber usar esas herramientas. Aunque saber **diseñar no es un requisito** para convertirte en Frontend Developer, definitivamente es una habilidad que puede ayudarte muchísimo en tu trabajo del día a día.

Nuestra principal responsabilidad es comunicarnos con los usuarios. Así que adquirir buen gusto sobre diseño nos ayudará a mejorar la experiencia que les ofrecemos.

Y lo contrario también es cierto: **las mejores diseñadoras web saben maquetar**, entienden cómo funciona CSS, saben que todo en la web son componentes (cajitas) y aplican esta mentalidad para diseñar interfaces.

El primer paso es aprender los fundamentos del diseño: peso, color, contraste, dirección… Luego podemos aprender sobre el diseño de interfaces, las buenas prácticas para cada tamaño de pantalla y la importancia de un buen sistema de diseño.

---

- **Preprocesadores de CSS**

A algunos desarrolladores les da pereza aprender sobre compiladores de CSS. Pero la verdad es que **Sass, Less y Stylus** son muy utilizados en equipos de todo el mundo para hacer más fácil, rápido y ordenado nuestro desarrollo.

CSS no es un lenguaje de programación, pero con estas herramientas podemos usar

un poco más limpio, escalable y fácil de mantener.

Además, para el proceso de compilación debemos usar la terminal. Así que usar precompiladores de CSS puede ayudarnos a conocer sobre Unix y Linux, una parte fundamental en el mundo de desarrollo de software profesional.

> 10 errores que delatan a un programador junior

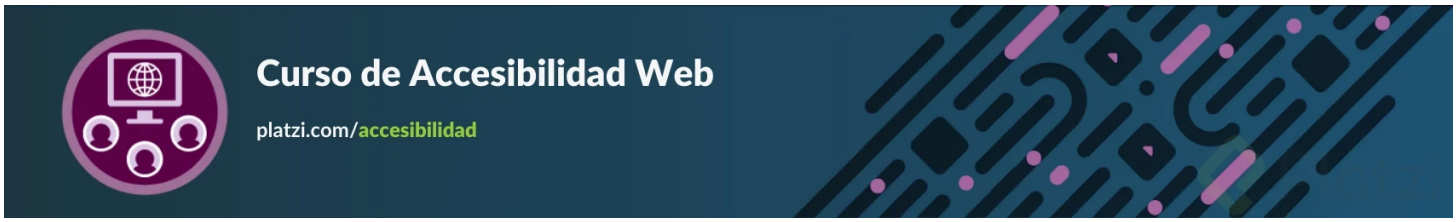
---

- **Accesibilidad Web**

La accesibilidad es una de las características por la que menos equipos se preocupan. Consiste en asegurarnos de que cualquier persona pueda usar nuestras aplicaciones, incluso si tienen problemas de vista, usar lectores de pantalla o tienen alguna discapacidad que no les permite interactuar con un mouse.

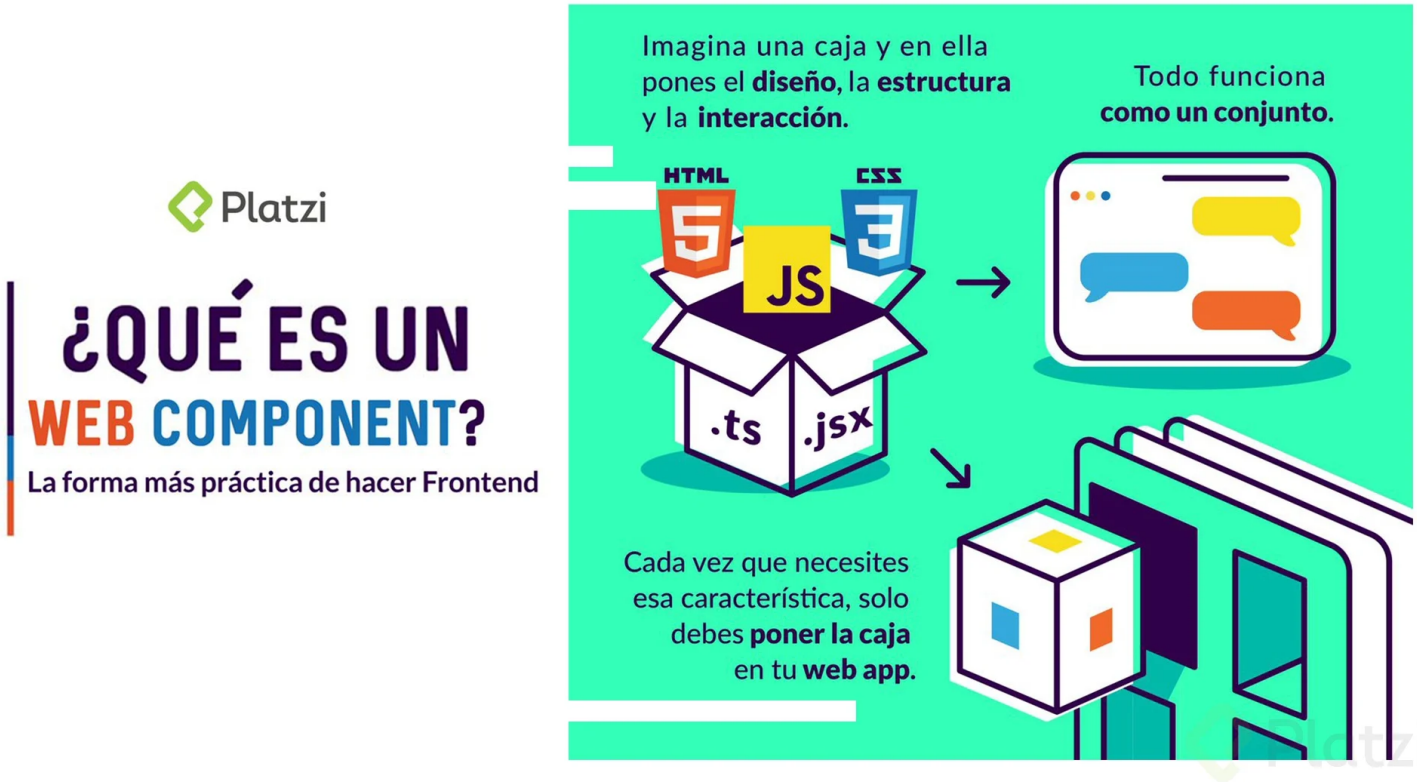
Debemos aprender a usar colores contrastantes, preparar nuestra estructura HTML para que sea operable por medio de robots, crear elementos específicamente para omitir o acceder al contenido importante, entre muchas otras estrategias.

Además de ayudar a que tu página web ofrezca mejores experiencias para todo tipo de personas, aprender de accesibilidad web es una habilidad que puede ayudarte a destacar como frontend.




## Componentes Reactivos

Los componentes reactivos son como cajitas con el código HTML, CSS y JavaScript de cada pequeño elemento de nuestra aplicación.



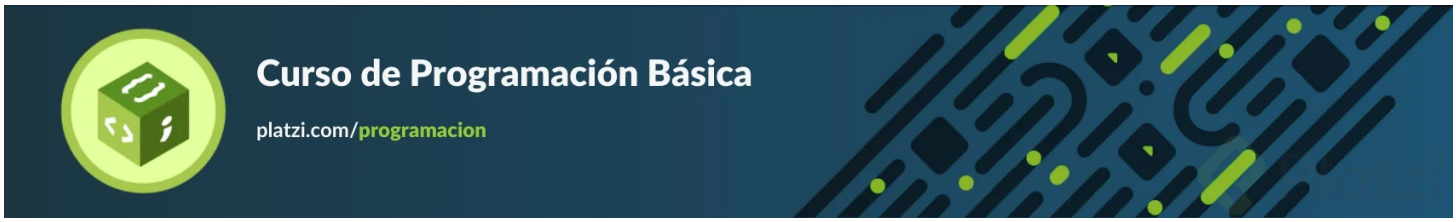

Es un poco extraño porque estamos acostumbrados a trabajar por separado la estructura, los estilos y la lógica en JavaScript. También solíamos trabajar todo el código en un solo archivo gigante. Pero con componentes reactivos debemos pensar en componentes independientes que pueden comportarse diferente dependiendo de en qué parte de la aplicación los ubicamos.

Este concepto lo introdujo Facebook con React.js. Podemos encontrarlo, por ejemplo, en un sistema de comentarios. Este componente aparece en muchas partes de la aplicación, pero se comportará diferente dependiendo si hay o no comentarios de nuestros amigos, si los comentarios están deshabilitados, si alguien más ha escrito algún comentario...

Los componentes reactivos son la forma más práctica y organizada de trabajar las interfaces de un sitio web.

## Interactividad y Programación en JavaScript

A diferencia de HTML y CSS, JavaScript sí es un lenguaje de programación. Si aún no lo has hecho, recuerda que puedes aprender a programar tomando el Curso Gratis de Programación Básica en Platzi.




---

- **Asincronismo en JavaScript**

El asincronismo es uno de los conceptos más difíciles de aprender en JavaScript. Nuestro código no necesariamente se ejecuta en el mismo orden en que lo escribimos. Algunos procesos pueden tardar un tiempo que no controlamos desde el frontend. Y, aún así, debemos preparar el código para ejecutar alguna acción cuando esta tarea se complete.

Esto lo encontramos, por ejemplo, en los comentarios de Facebook. Cuando le das clic al botón de ver más, los nuevos comentarios no aparecen de inmediato, JavaScript debe esperar a que el backend logre darle una respuesta. Pero, cuando reciba la información, podrá mostrar el resultado en el navegador.

Hay diferentes herramientas para programar asincronamente. La más vieja de todas son los **callbacks**, una función que le pedimos a JavaScript que ejecute cuando

```
recibirXAsincronamente(function callback(x) {
  console.log(x);
});
```

Ahora utilizamos *promesas*. Herramientas un poco mejor preparadas para que la aplicación no se rompa en casos inesperados. Por ejemplo, si los datos que esperamos nunca llegan o son incorrectos.

```
recibirXAsincronamente()
  .then(x =&gt; console.log(x))
  .catch(err =&gt; console.error(err))
```

También podemos utilizar promesas con una la sintaxis de *async await*.

```
async function app() {
  try {
    const x = await recibirXAsincronamente();
  } catch(err) {
    console.error(err);
  }
}
```

---

- ## APIs, AJAX y Fetch

Las APIs son estructuras de transporte de datos que se encargan de comunicar al frontend con el backend. Hacer peticiones a APIs es el uso más común del asincronismo en JavaScript.

Consumir APIs es una de las principales responsabilidades del frontend. Para crearlas o consumirlas podemos usar diferentes herramientas:

– **XML**: es muy similar a HTML, pero no carga información visual (para humanos),

verborrágico, se necesita mucho código para definir poca información.
– **JSON**: es muy similar a JavaScript, pero no es un lenguaje de programación (con variables, funciones, ciclos y condicionales), sino un lenguaje para transportar datos con una sintaxis mucho mejor que XML. La gran mayoría de APIs usan JSON.
– **GraphQL**: es la capa de transporte de datos más moderna. Facilita un montón encontrar los datos que necesita el frontend sin complicar demasiado al backend. Es casi un lenguaje de scripting.

---

- **Web Workers**

Nuestras computadoras tienen múltiples procesadores. En JavaScript por defecto solo podemos utilizar uno. Afortunadamente, los Workers nos permiten ejecutar diferente código JavaScript al mismo tiempo.

Esto es muy útil cuando debemos realizar múltiples procesos al mismo tiempo. En Instagram, por ejemplo, queremos que carguen las publicaciones y en paralelo las stories. Y para un mejor rendimiento podemos programar cada petición en un hilo diferente.

---

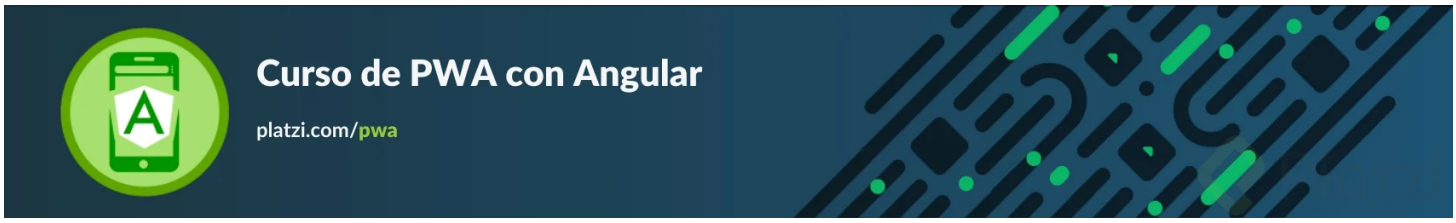
- *Service Workers y Progressive Web Apps*

Los service workers son un tipo de web workers que nos ayudan a que nuestras aplicaciones webs tengan características de aplicaciones móviles: experiencias offline-first, mejorar la carga notificaciones push, incluso podemos agregarlas a nuestro home screen como si fueran una app más de nuestro celular.

Las aplicaciones que implementan estas características las conocemos como Progressive Web Applications. Y tienen muchas ventajas. Nos permiten mejorar la experiencia para los usuarios con tiempos de carga mucho más cortos, facilitar las

> Progressive Web Apps vs. Aplicaciones nativas

Tal vez tu sitio web no tiene que convertirse 100% en una PWA. Pero aprender a trabajar con los service workers te ayudará a implementar diferentes características que definitivamente harán más felices a tus usuarios.




---

- **Manipulación del DOM**

El DOM es el código HTML que se transforma en páginas web. Nuestras páginas web pueden cambiar su estructura a medida que los usuarios interactúan con ella. Así que muy seguido vamos a usar JavaScript para manipular el DOM.

Antes usábamos jQuery para manipular directamente el DOM. Pero jQuery se quedó atrás desde hace muchos años. Hoy podemos usar herramientas como Angular para cumplir esta misma tarea.

Además, Vue y React utilizan el **Virtual DOM**. Una copia del DOM mucho menos pesada, que nos permite realizar operaciones más rápido y actualizar el DOM solo cuando sea necesario.

> El Stack ideal para JavaScript en 2020

---

- **TypeScript**

que los navegadores no entienden el código en TypeScript, así que hay que compilarlo a JavaScript para que funcione correctamente.

TypeScript ha sido desde el 2019 uno de los lenguajes más amados en el mundo. Estudiarlo puede ayudarnos a aprender sobre buenas prácticas y programación orientada a objetos. Además, puede ser nuestro primer gran acercamiento a aprender nuevos lenguajes de programación.

---

- **NPM** y **Webpack**

Webpack y NPM son herramientas que los desarrolladores frontend profesionales utilizan todos los días. Nos ayudan a organizar nuestro flujo de desarrollo, trabajar con librerías externas a nuestro proyecto y optimizar el tamaño de nuestro código.

NPM es el gestor de dependencias más popular en JavaScript. Nos permite publicar e instalar paquetes en JavaScript para el navegador o Node.js. Además, con NPM ejecutamos la mayoría de comandos para iniciar, compilar o depurar nuestros proyectos.

Webpack nos permite escribir JavaScript moderno, organizar nuestros archivos, combinar nuestro código junto con el del resto de librerías que utilizamos y optimizar este código para que nuestras aplicaciones carguen tan rápido como sea posible.

Diferentes tecnologías como React Native y Electron han introducido a JavaScript en el mundo de las aplicaciones nativas. Hoy es posible programar apps móviles o de escritorio en JavaScript.

Aún debemos aprender los lenguajes, herramientas y flujos de trabajo para este tipo de aplicaciones. Pero JavaScript puede ayudarnos a entrar en este nuevo mundo con una curva de aprendizaje mucho más suave y obteniendo resultados impresionantes desde el principio.

> Como crear tu propia App Móvil desde cero

## Conclusiones

El frontend no es fácil. Es necesario especializarnos, actualizarnos constantemente y entender muy bien las herramientas con las que trabajamos para entregar un resultado responsable.

Una desarrolladora profesional no tiene miedo de aprender nuevos lenguajes o tecnologías. Aprender a maquetar es el inicio. Pero aprender a programar y conocer muy bien JavaScript es lo que verdaderamente te ayudará a crecer en esta carrera. Y, por supuesto, en Platzi puedes aprender todas estas tecnologías para impulsar tu carrera profesional.

**#NuncaParesDeAprender**

B I U Insertar código Enlace Imagen

Suma tu comentario + 2

Ordenar por: Top

18 **onyx** 8535 Puntos 2 años

Y por eso digo... como me gusta el Backend

Responder + 2

8 **PabloValenzuela** 30053 Puntos 2 años

Esta información me hizo estallar la cabeza, Fue un resumen completo de lo que es verdaderamente el desarrollo web. Lo mejor es que todo lo que necesitamos saber este en Platzi.

Responder + 2

6 **luistua** 24491 Puntos 2 años

No saben qué valioso es dar este tipo de contextos porque luego, justo por no saber en dónde está uno parado es fácil desanimarse.

Es decirte: ¿Quieres ser frontend? Esto es lo que necesitas aprender.

Responder + 2

4 **nicoaguilar** 1845 Puntos 2 años

Muy buen post. Gracias Platzi!

123

Share

3

A todo esto le sumaría este completísimo vídeo de Fredy que explica toda la ruta de un desarrollador profesional. https://www.youtube.com/watch?v=ExsGyZDBlJQ

Responder + 2

**alvarohrv** 4503 Puntos

2 años

3

Me llego justo a tiempo este articulo ^^!

Responder + 2

**juand_silva** 27709 Puntos

2 años

3

No estoy tan de acuerdo con esta cita del post:

> Una gran característica de HTML es que funciona a pesar de que cometamos cierto tipo de errores. Esto sucedió por culpa de Internet Explorer, el ex navegador oficial de Microsoft que quería matar al resto de navegadores y por eso rompía los estándares de la web todo el tiempo.

En realidad HTML se concibió desde el inicio con una naturaleza permisiva, porque la idea era que las **personas comunes y corrientes publicaran contenido para contruir la Web como la conocemos hoy**. Si eso no fuera así la web no sería tan polular como lo es ahora, si no que se limitaría a contenido hecho o programado por desarrollares.

Fuente: https://developer.mozilla.org/es/docs/Learn/HTML/Introduccion_a_HTML/Debugging_HTML

Responder + 2

**luisrovez** 9502 Puntos

2 años

3

Falta una opción de guardar las artículos para leerlos después 😅, ano ser que ya lo tengan 🤔.

**juandc**

2 años

2

En la app sí tienes eas opción. 😉

Responder + 2

**leonardc** 882 Puntos

2 años

2

Excelente información, mil gracias...

Responder + 2

123

Share

Responder + 2

2

 **RogerFD_** 35633 Puntos 2 años

Excelente resumen!

Responder + 2

2

 **Caso_Torres** 9961 Puntos 2 años

excelente

Responder + 2

2

 **jordiifl** 2306 Puntos 2 años

Toda una entrada de desarrollo web.

Responder + 2

2

 **Bervivez** 139021 Puntos 2 años

Y hay quienes piensan que el frontend es fácil, y a lo mejor lo sea, pero ser muy bueno en lo que haces y destacarte (y eso va para cualquier rama) eso es otra cosa... Muchas gracias por el articulo!! me cae como anillo al dedo.

Responder + 2

2

 **warias7** 3394 Puntos 2 años

buena información, gracias

Responder + 2

2

 **pamelatgtz** 4746 Puntos 2 años

Muchas gracias por la información, aunque me doy cuenta de que todavía tengo mucho por aprender.

Responder + 2

1

 **juli_dossanto5** 5394 Puntos 2 años

Las carreras que tienen que ver con Desarrollo Frontend son bien pagadas por las empresas?

123

Share

1 Una vez hice esa pregunta, y me respondieron que las carreras mejor pagadas son las relacionadas con la inteligencia artificial, debido a que hay cada día más demanda y poco personal calificado que quiera adentrarse a esas disciplinas.

**jayroht93** 2 años

2 Hola amigo, hay bastante oferta de trabajo. En mi caso entre 1700 a 2000 USD, trabajando con vue o react, complementando o teniendo conocimiento en otras herramientas como aws, apis, conocimientos básicos de backend, etc. También conosco personas que andan al rededor de 3000 USD.

**Andresdst** 2 años

2 Jayroht93 de que pais eres?

Responder + 2

**fercas44** 662 Puntos un año

1 Buena info!!

Responder + 2

**gutierrez-diego** 9755 Puntos un año

1 Un artículo groseramente completo en lo que corresponder ser un Fronted, toda la información que se necesita saber esta aquí. Es decir, con esta información arrancas y el resto depende de ti, información adicional es que ames lo que haces porque más allá que estudies algo que no te apasione terminarás abandonandolo en el futuro. Ame más el Frontend con este artículo.

Responder + 2

**jecm** 19155 Puntos un año

1 Muchas gracias por la enseñanza!

Responder + 2

**dbzdavidbaez** 22323 Puntos un año

1 Excelente resumen...

Motiva a seguir aprendiendo y unir todos estos conceptos...

123

Share

1

Madre Naturaleza y Madre Tecnología las amo y gracias por existir en mi vida ! Gracias Platzi cada vez quiero saber más y aprender mejor.

Responder + 2

1

 **omargabriel_15** 38707 Puntos 9 meses

Esta información me hizo estallar la cabeza, Fue un resumen completo de lo que es verdaderamente el desarrollo web. Lo mejor es que todo lo que necesitamos saber este en Platzi.

Responder + 2

1

 **gutierrez-diego** 9755 Puntos 2 años

Siento que es un largo camino pero se hace con gusto, lo mas dificil ya lo hice que fue comenzar, ahí voy Fronted...👍😳💥🚀👊🎖💯👏🎛

Responder + 2

0

 **KarolPerezM** 3307 Puntos un año

Todo esto es nuevo para mí. 😯😳

Responder + 2

## Entradas relacionadas

27

### ¿Por qué debes optimizar tu website?

La optimización de rendimiento a los sitios web ofrece un gran número de beneficios: Incrementa el porcentaje de conversión de nuestros usu

 demian

123

Share

Sin importar la velocidad de internet que un usuario tenga, siempre se espera que al abrir una página web el sitio sea ágil. Si el usuario s

fannytaviles

65

## Cómo optimizar tu tiempo para estudiar en Platzi

Hoy en día vivimos en un mundo donde cada vez tenemos menos tiempo (tiempo para nuestras familias, nuestros hijos, nuestros estudios, descan

bustosfredy

123

Share