DISEÑO WEB EN LA ERA MÓVIL

DIEZ *TIPS* SOBRE *RESPONSIBLE RESPONSIVE DESIGN*

TESIS

QUE PARA OBTENER EL GRADO DE MAESTRÍA EN

COMUNICACIÓN CON MEDIOS VIRTUALES

PRESENTA:

ING. JONATHAN ULISES MIRANDA CHARLES

ASESOR: ROSELENA VARGAS VELASCO

MÉXICO, CDMX. DICIEMBRE, 2017

RECONOCIMIENTO DE VALIDEZ OFICIAL DE ESTUDIOS DE LA SECRETARÍA DE EDUCACIÓN PÚBLICA SEGÚN ACUERDO NO. 2005625 DE FECHA 22 DE SEPTIEMBRE DE 2005. CLAVE 2012.

Introducción:

> "La creación del contenido no debería ser una tarea recóndita. No debería ser esto algo inescrutablemente arcano que sólo los expertos y dorados gurús de las ciencias de la computación pueden hacer."
>
> (Brendan Eich – Creador de *JavaScript*, 2014).

Desde la aparición del *iPhone* en 2007, el uso de los dispositivos móviles ha cambiado las dinámicas del ser humano, su crecimiento ha sido constante durante los últimos años, tanto así que para cuando se escriben éstas líneas (2014) ya han superado en ventas al *PC* tradicional y muy pronto lo harán en el consumo de contenidos y servicios en la web.

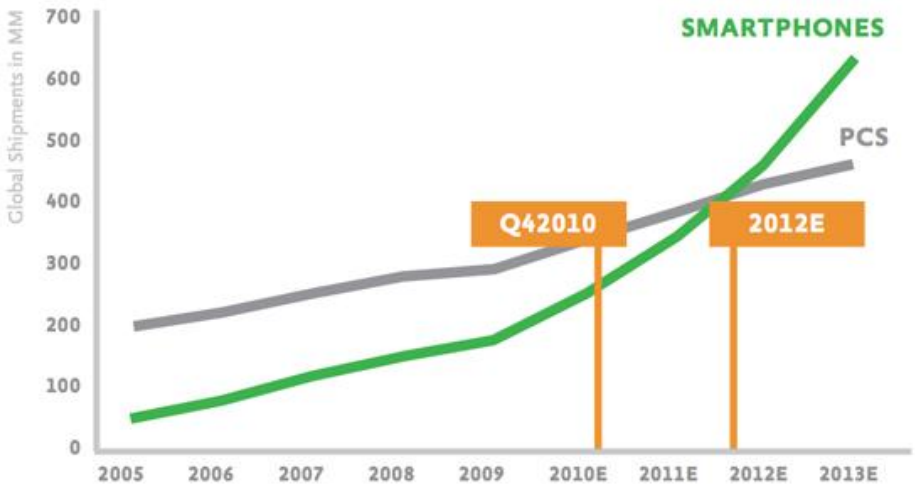

Fig. 0.1 Crecimiento de ventas *smartphones* vs *PC's* (Wrobleski, 8 2011).

Esta investigación analiza el estado de la web a partir del crecimiento de los dispositivos móviles y proporciona una serie de buenas prácticas de diseño web en la era móvil, tomando como eje principal el *Responsive Web Design*.

Con el surgimiento de los dispositivos móviles, el diseño web ha sufrido un cambio de paradigma y por ende las prácticas y técnicas que veníamos utilizando se ven afectadas.

El enfoque de la presente investigación es de carácter práctico, ya que los temas expuestos están relacionados con las tecnologías y técnicas actuales que se recomiendan seguir sobre diseño web en la era móvil.

Introducción.

Índice.

Capítulo I: EL ESTADO DE LA WEB.

2011: El año del móvil.

> "*There's no need to declare this "the year of mobile." If anything, last year was the year of mobile in terms of the growth in both mobile usage and the availability of mobile sites and apps. Now, however, it's time to redesign your mobile site, because your existing version is probably far below users' growing expectations for user experience quality.*"
>
> (Nielsen,2011).

El año 2011 supuso un gran cambio en la forma como consumimos internet debido a varios acontecimientos.

Con el aumento de las ventas en los teléfonos inteligentes y las tabletas, las conexiones de datos más rápidas, la mejora de *hardware* y la explosión de las aplicaciones móviles, el uso de *Internet* en los dispositivos móviles creció rápidamente.

Los teléfonos inteligentes cada vez más grandes y rápidos con pantallas de alta definición y procesadores de varios núcleos, comenzaban a superar las características de muchas de las *PC's* del momento. Estos dispositivos comenzaban a utilizarse cada vez más en todos los ámbitos del ser humano: la educación, los negocios, el hogar, etc.

Ese año un estudio por la agencia de marketing digital, *Tecmark*, demostró que los dispositivos móviles representaron el 6.5% del tráfico web mundial. Las cifras de otras fuentes apoyaron esta tesis, como las del sitio web *Global Stats*.

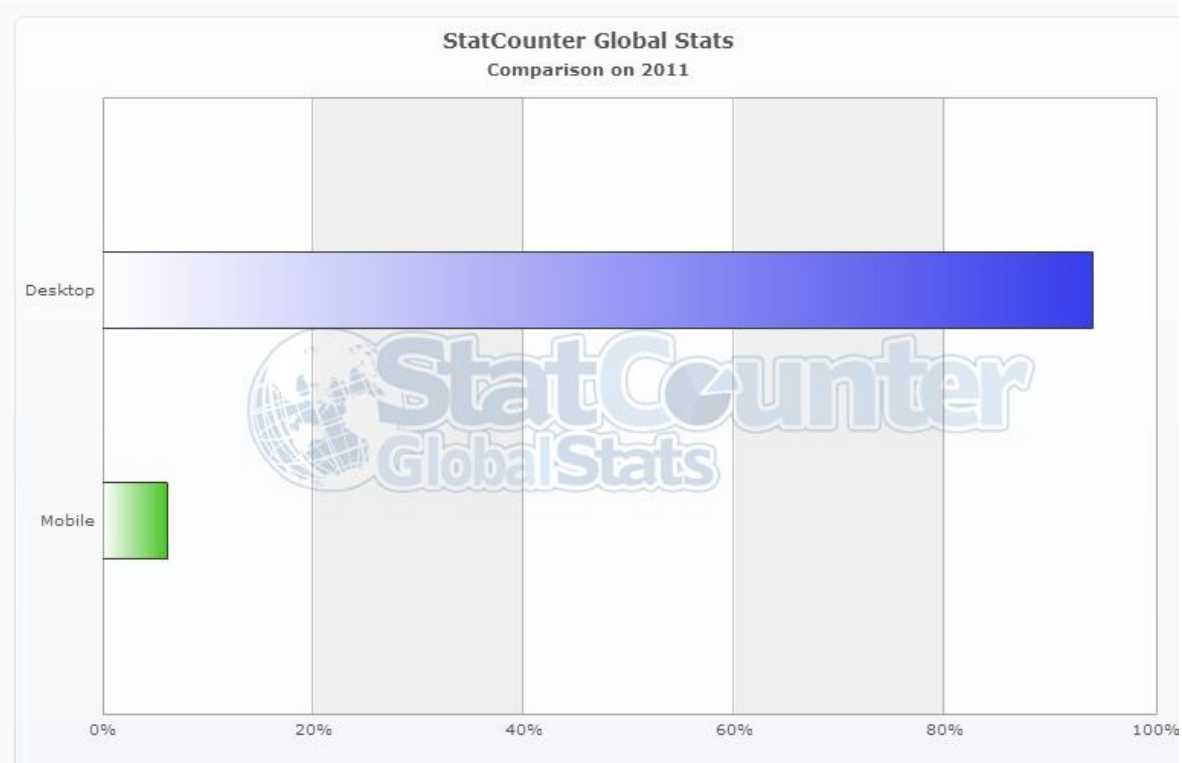

Fig. 1.1 Crecimiento de uso Mobile vs Desktop 2011.

Si bien las cifras para el uso móvil de la web eran relativamente pequeñas en comparación con el tradicional acceso desde el *PC* de escritorio, no se podía ignorar que el uso de la web en el móvil era cada vez mayor y continuaría haciéndolo, con el paso del tiempo.

Varios analistas tecnológicos predecían que para el año 2015 el uso de la web en dispositivos móviles superaría el de tráfico del *PC* de escritorio.

Dicha predicción está a punto de suceder.

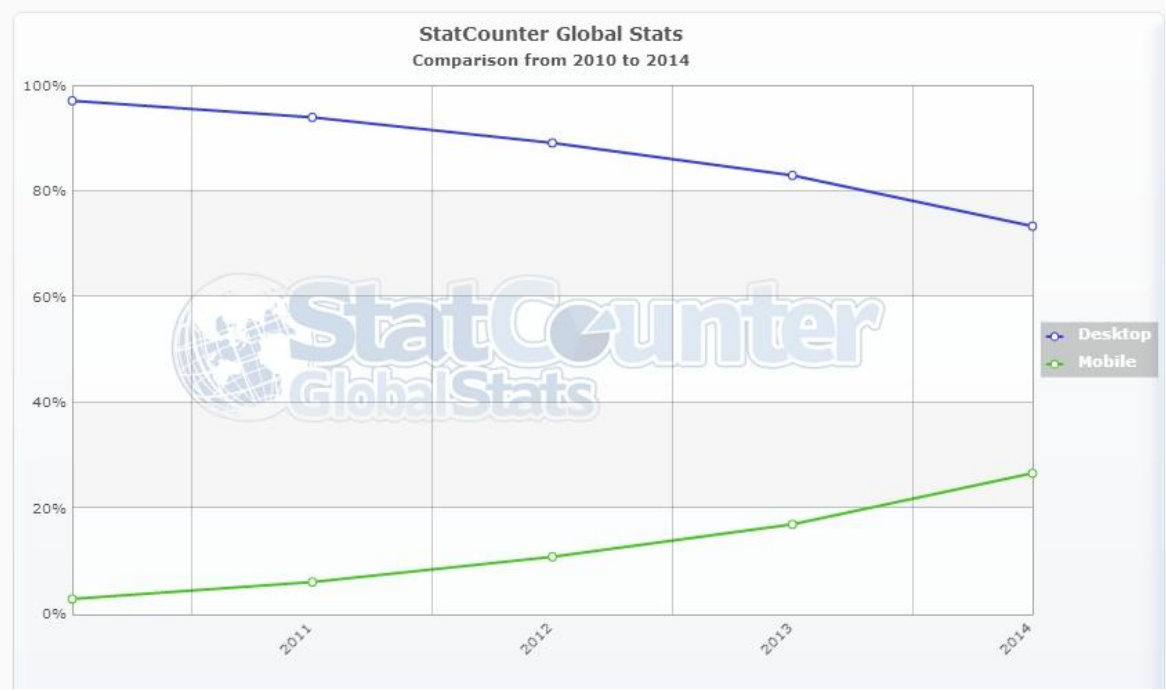

Fig. 1.2 Crecimiento de uso Mobile vs Desktop 2010-2014.

Por otro lado cada mes la compañía de software del navegador Opera, lleva a cabo un análisis de las tendencias que afectan a la web móvil de su navegador (Opera mini) y publica sus resultados en http://www.opera.com/smw/, éstos reflejan el enorme crecimiento continuo del tráfico web de los dispositivos móviles. En el informe de noviembre de 2011 podemos ver 2 hechos importantes:

- Un aumento de 84.2% en usuarios únicos de Opera Mini respecto a septiembre 2010 y
- Un aumento de 114.1% en páginas vistas en Opera Mini respecto a septiembre 2010.

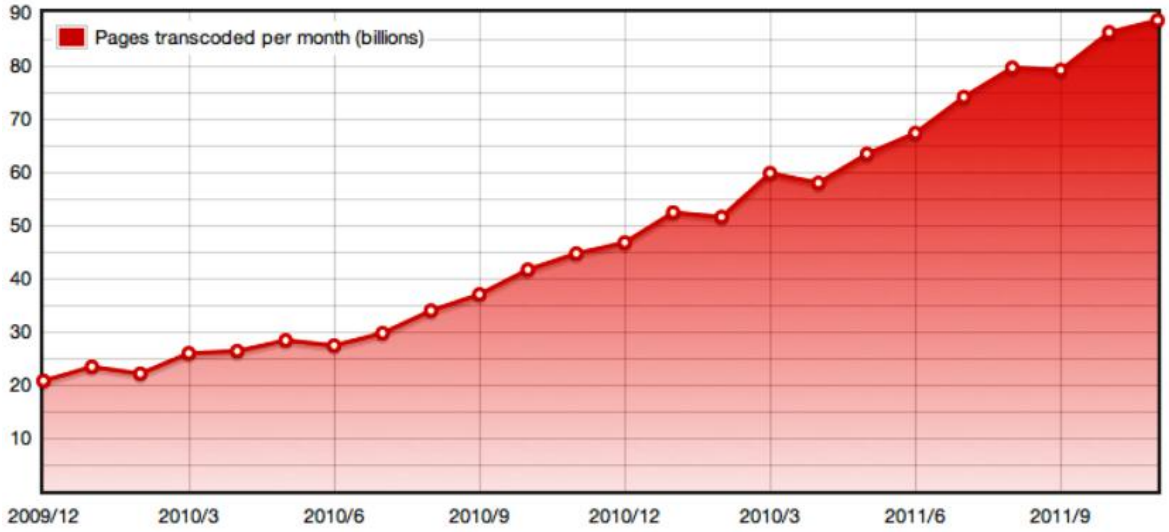

Fig. 1.3 Reporte del estado de la web septiembre 2011 (Opera,2011).

Una vez más, hay que señalar que el porcentaje real de visitas a la web en general de móviles, en comparación al tráfico de escritorio, era todavía pequeño (alrededor de 2%) pero fue evidente que de ahí en adelante el acceso a través del móvil crecería en todos los sectores del ecosistema digital.

Finalmente otro hecho importante con el que culmino el 2011 fue que *Adobe* desistió por seguir desarrollando el *plug-in* de *Flash* para dispositivos móviles publicándolo en el siguiente artículo *Flash to Focus on PC Browsing and Mobile Apps*:

> *"Adobe is all about enabling designers and developers to create the most expressive content possible, regardless of platform or technology. For more than a decade, Flash has enabled the richest content to be created and deployed on the web by reaching beyond what browsers could do. It has repeatedly served as a blueprint for standardizing new technologies in HTML. Over the past two years, we've delivered Flash Player for mobile browsers and brought the full expressiveness of the*

> *web to many mobile devices. However, HTML5 is now universally supported on major mobile devices, in some cases exclusively. This makes HTML5 the best solution for creating and deploying content in the browser across mobile platforms. We are excited about this, and will continue our work with key players in the HTML community, including Google, Apple, Microsoft and RIM, to drive HTML5 innovation they can use to advance their mobile browsers."*
>
> (*Adobe*, 2011).

Con todo lo sucedido en 2011 sumado a la gran difusión que venía teniendo *HTML5* como nuevo estándar de la web, 2012 se convertiría en el año de *HTML5*.

2012: El año de *HTML5*.

> "*HTML5* es el presente de la web y si no estás asimilando lo que está pasando ya eres parte de la vieja generación de desarrolladores. Eso tendría que tenerte preocupado. Entender *HTML5* es entender que hoy nos conectamos desde teléfonos móviles, tabletas, *eBooks*, *netbooks*, computadores y otra gama de dispositivos. Es entender que se acabaron los *webmasters* y hoy hablamos de equipos multidiciplinarios de empresas de tecnología que cuentan con *frontends*, *backends*, *sysadmins*, *mobile devs*, *comunity managers* y arquitectos de información en los proyectos que están reiventando mercados y generando tráfico e ingresos."
>
> (Vega, Van Der Henst, 2011).

En estricto sentido *HTML5*, es la nueva versión del lenguaje de marcado de etiquetas con el que hacemos la web, pero va más allá, el concepto engloba el estado actual de las tecnologías con las que se crea *internet*.

HTML5 es hablar de *HTML*, *CSS* y *JavaScript* que en conjunto, generan una sinergia de tecnologías, que nos ayudan a elaborar sitios que antes eran impensables con estándares, y que necesitábamos de tecnologías como *Flash*, que no son nativas de la web.

Hay que destacar que el uso desmedido de la tecnología *Flash* para hacer diseño web en la década anterior (los años 2000) impulsó el avance de los estándares que tenemos actualmente.

Gracias a la integración de *CSS3* y *JavaScript* con la nueva especificación *HTML5* podemos lograr sitios más interactivos, mejor diseñados y todo sin la necesidad de implementar extensiones o software no nativos a la web.

Por lo anterior *HTML5* se podría definir con la siguiente fórmula:

Fig. 1.4 Fórmula *HTML5*.

HTML5 es la suma de la nueva versión del estándar *HTML* más la nueva versión de *CSS* la cual nos permite darle estilos como nunca antes a nuestro contenido, más el conjunto de *API´s* de *JavaScript* que le dan interactividad a la web.

Sintetizando esta fórmula podríamos decir que *HTML5* proporciona tres características fundamentales para la web:

- *HTML* define estructura y contenido.
- *CSS* define diseño y presentación.
- *JavaScript* define funcionalidad e interactividad.

La misma *W3C*, qué es el consorcio encargado del estándar así lo ha promulgado elaborando un logo y toda una campaña de marketing para promoverlo.

> *"AN HTML5 LOGO*
>
> *It stands strong and true, resilient and universal as the markup you write. It shines as bright and as bold as the forward-thinking, dedicated web developers you are. It's the standard's standard, a pennant for progress. And it certainly doesn't use tables for layout.*
>
> *We present an HTML5 logo."*
>
> (*W3C*, 2011).

Fig. 1.5 Sitio Oficial de *HTML5*.

Para 2012 *HTML5* estaba incompleto, se seguían proponiendo mejoras en el draft y en la práctica algunas de sus características no habían sido implementadas en todos los navegadores. Este era uno de los argumentos más usados por sus detractores para no implementarlo.

HTML5 comenzó a definirse teóricamente en 2008, en la práctica empezó a implementarse en 2010. Para el 2012 con todos los sucesos ya mencionados, *HTML5* estaba en la antesala de su hegemonía y aunque no estaba terminado y no lo estaría hasta 2014 que la *W3C* cerrará oficialmente el *draft*, los dispositivos y navegadores (móviles y de escritorio) de entonces ya soportaban muchas de sus implementaciones y con el paso del tiempo siguen y siguen aceptándolas. Con lo anterior es importante mencionar que *HTML5* no es un software que alguna empresa haga y se tenga que terminar para liberarse, por el contrario es una especificación viva, que se implementa y se experimenta día a día a través de los navegadores, los cuales son responsables de que cada vez haya más compatibilidad; incluso ya se esta comenzando el borrador de la siguiente versión: *HTML5.1*.

Otro factor para considerar 2012 como el año de *HTML5* es que en los eventos de tecnología de las grandes empresas como *Google*, *Microsoft* y *Apple* presentaron diferentes dispositivos compatibles con *HTML5*:

- *Google* presentó teléfonos y tabletas *Android* (*Nexus 7*) con la nueva versión del sistema operativo (*JellyBean 4.1*), se lanzó *Chrome* para todo tipo de dispositivos, incluyendo el *iPad*, la *Nexus Q*, los *Google Glass*, estos últimos tomando como punto de partido *HTML5* para el desarrollo de sus aplicaciones.
- *Apple* presenta su nuevo sistema operativo móvil *iOS 6*, el *iPad 3* y los nuevos equipos *Macbook* con mayor densidad de pixeles en sus pantallas (*Retina Display*) salen a la luz.
- *Microsoft* lanza su tableta *Surface* y su sistema operativo *Windows* 8 con la interfaz *Metro*, más orientado a dispositivos móviles y tactiles.

Todo ello favoreciendo y proliferando el ecosistema digital de *HTML5*.

2013: El año de *JavaScript*.

> "*JavaScript* pasó de ser el lenguaje horrible que la gente 'aguantaba' a un entorno maduro de desarrollo usado en poderosos proyectos tanto en el cliente como en el servidor. *Node.js*, junto con su madura selección de librerías, el nuevo *jQuery* que abandona *IE8* e inventos como *Meteor.js* y *Backbone.js* son mis proyectos favoritos para mostrar el futuro de *JavaScript*."
>
> (Vega, 2013).

Para el año 2013 todo mundo hablaba de *HTML5*, incluso la gente sin conocimientos técnicos (los clientes) hacían comentarios como: 'quiero que mi sitio se vea en dispositivos móviles' o 'no quiero que uses Flash porque ya no se ve en móviles' o 'quiero que mi sitio tenga *HTML5* y sea responsive'.

Anteriormente se menciono que *HTML5* es la sinergia de *HTML*, *CSS* y *JavaScript*, al ser las piedras angulares de las nuevas tecnologías de la web, el consorcio de la *W3C* ha hecho una clasificación de 8 implementaciones que integran el nuevo estándar:

	**Semántica:** Dar sentido a la estructura y contenido es nodal en *HTML5*, gracias a microdatos, microformatos, nuevas etiquetas, la redefinición de otras y nuevos atributos y elementos de formulario, que en conjunto propician una web basada en datos tanto para sus usuarios como para las máquinas y programas existentes en ella.
	**Fuera de Línea y Almacenamiento:** Se puede comenzar a desarrollar rápidamente aplicaciones web incluso si no hay conexión a *Internet*, gracias a la caché de *HTML5*, así como almacenamiento local, indexado de bases de datos, y acceso a archivos locales.

	**Acceso al Dispositivo:** Comenzando con la *API* de geolocalización, se están desarrollando increíbles innovaciones de acceso a dispositivos, desde el acceso de entrada de audio / vídeo a micrófonos y cámaras, hasta los datos locales como los contactos, sin olvidar los eventos y la orientación del dispositivo.
	**Conectividad:** Conectividad más eficiente significa más chats en tiempo real, juegos más rápidos y una mejor comunicación entre el cliente y el servidor, web *sockets* y eventos al servidor están empujando como nunca que esto se lleve a cabo.
	**Multimedia:** Audio y video son ciudadanos de primera clase en la web *HTML5*, que viven en armonía con sus aplicaciones y sitios. ¡Luces, cámara, acción!
	**Efectos, Gráficos y 3D:** Impresionantes efectos visuales de forma nativa en el navegador, gracias a *SVG*, *Canvas*, *WebGL* y características *3D* de *CSS3*.

	**Rendimiento e Integración:** Haga que sus aplicaciones y contenidos web, sean dinámicos y más rápidos con una variedad de técnicas y tecnologías como *web workers* y *XMLHttpRequest2*. Nadie debe esperar la carga de un sitio viendo su reloj.
	***CSS3*:** Ofrece una amplia gama de estilización y efectos, sin tener que sacrificar estructura semántica o rendimiento en la aplicación web. Nuevos selectores, nuevas propiedades, animaciones, transformaciones *2D/3D*, esquinas redondeadas, efectos de sombra y fuentes descargables. Adicionalmente *Web Open Font Format* (*WOFF*) ofrece flexibilidad y control tipográfico más allá de lo que la web ha ofrecido antes.

Fig. 1.6 Tabla de las implementaciones de *HTML5*.

De éstas 8 implementaciones descritas, 6 tienen que ver directamente con *JavaScript*, sólo exceptuando semántica y *CSS3* las demás son nuevas interfaces de programación de aplicaciones (*API's* - *Application Programming Interface*) del lenguaje.

Con *HTML5*, *JavaScript* paso de un lenguaje de script a toda una plataforma de desarrollo *frontend*, pero, ¿cómo lo logro? A continuación una breve historia:

JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas, conforme el navegador lee el

código lo va ejecutando. Se diseñó para realizar tareas que pudieran añadir interactividad a los documentos *HTML*, por ejemplo: creando elementos dinámicamente, validando formularios, detectando características del navegador del usuario, almacenar y recuperar información del visitante, reaccionar a eventos, entre otras.

Fue creado por Brendan Eich de Netscape en mayo de 1995, originalmente su nombre era Mocha, mismo que cambiaron a los tres meses por LiveScript, finalmente cuando Netscape firmó una alianza con Sun Microsystems para el desarrollo del lenguaje de programación cambiaron el nombre por el de JavaScript. La razón del cambio fue exclusivamente por marketing, ya que Java era la palabra de moda en el mundo informático. A pesar de ello, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java.

La primera versión de *JavaScript* implementada en el *Netscape Navigator 3* fue un completo éxito. Al mismo tiempo, *Microsoft* lanzó una vil copia llamada *JScript* con su navegador *Internet Explorer 3*, al que le cambiaría el nombre a *VisualBasicScript* para burlar demandas legales.

Para evitar una guerra de tecnologías, *Netscape* decidió estandarizar el lenguaje y en 1997 se envió la especificación *JavaScript 1.1* al organismo *ECMA* (*European Computer Manufacturers Association*), mismo que lo estandarizó como un lenguaje de *script* multiplataforma e independiente de cualquier empresa bajo la denominación *ECMA-262*, en el que se definió por primera vez el lenguaje *ECMAScript*.

A partir de ese momento comenzaron a emerger conceptos, librerías y tecnologías que fueron fortaleciendo el entorno de *JavaScript*, como *AJAX* tecnología que llevo la interacción a otro nivel permitiendo ejecutar procesos de manera asíncrona sin necesidad de recargar el navegador y *JSON* un formato ligero de intercambio de datos, que con el

tiempo ha ido reemplazando a *XML*, para el transporte de información entre aplicaciones web, sistemas informáticos y plataformas de desarrollo.

A continuación una tabla que muestra el estado de *JavaScript* hasta el 2012 elaborada por su creador, misma que se puede consultar en el siguiente enlace.

Fig. 1.7 '*The State of JavaScript*' por Brendan Eich.

Hoy, *JavaScript* se está reinventando gracias a las nuevas *API's HTML5* como *Audio/Video*, *Geolocation*, *Storage*, *Cache*, *WebSQL*, *IndexDB*, *WebSockets*, *WebWorkers*, *ServerSentEvents*, *RequestAnimationFrame*, *History*, *FileAccess*, *GetUserMedia*, *TouchEvents*, *Vibration*, *Orientation*, *DragAndDrop*, *WebGL*, *Canvas*, *SVG*, entre otras; y proyectos que se

suman día a día para mejorar el ecosistema *JavaScript* como *jQuery*, *CoffeeScript*, *DART*, *Node.js*, *Angular.js*, *Backbone.js*, *Modernizr*, etc.

Lo antes mencionado contribuyo para que en 2013 *JavaScript* se colocara en el lugar número 9 del top de lenguajes de programación según el índice *TIOBE* y además fuera el lenguaje con mayor crecimiento de dicho año.

Jan 2014	Jan 2013	Change	Programming Language	Ratings	Change
1	1		C	17.871%	+0.02%
2	2		Java	16.499%	-0.92%
3	3		Objective-C	11.098%	+0.82%
4	4		C++	7.548%	-1.59%
5	5		C#	5.855%	-0.34%
6	6		PHP	4.627%	-0.92%
7	7		(Visual) Basic	2.989%	-1.76%
8	8		Python	2.400%	-1.77%
9	10	^	JavaScript	1.569%	-0.41%
10	22	^^	Transact-SQL	1.559%	+0.98%
11	12	^	Visual Basic .NET	1.558%	+0.52%
12	11	v	Ruby	1.082%	-0.69%
13	9	vv	Perl	0.917%	-1.35%
14	14		Pascal	0.780%	-0.15%
15	17	^	MATLAB	0.776%	+0.14%
16	45	^^	F#	0.720%	+0.53%
17	21	^^	PL/SQL	0.634%	+0.05%
18	35	^^	D	0.627%	+0.33%
19	13	vv	Lisp	0.604%	-0.35%
20	15	vv	Delphi/Object Pascal	0.595%	-0.32%

Fig. 1.8 Top 20 Lenguajes de Programación 2013, Índice *TIOBE*.

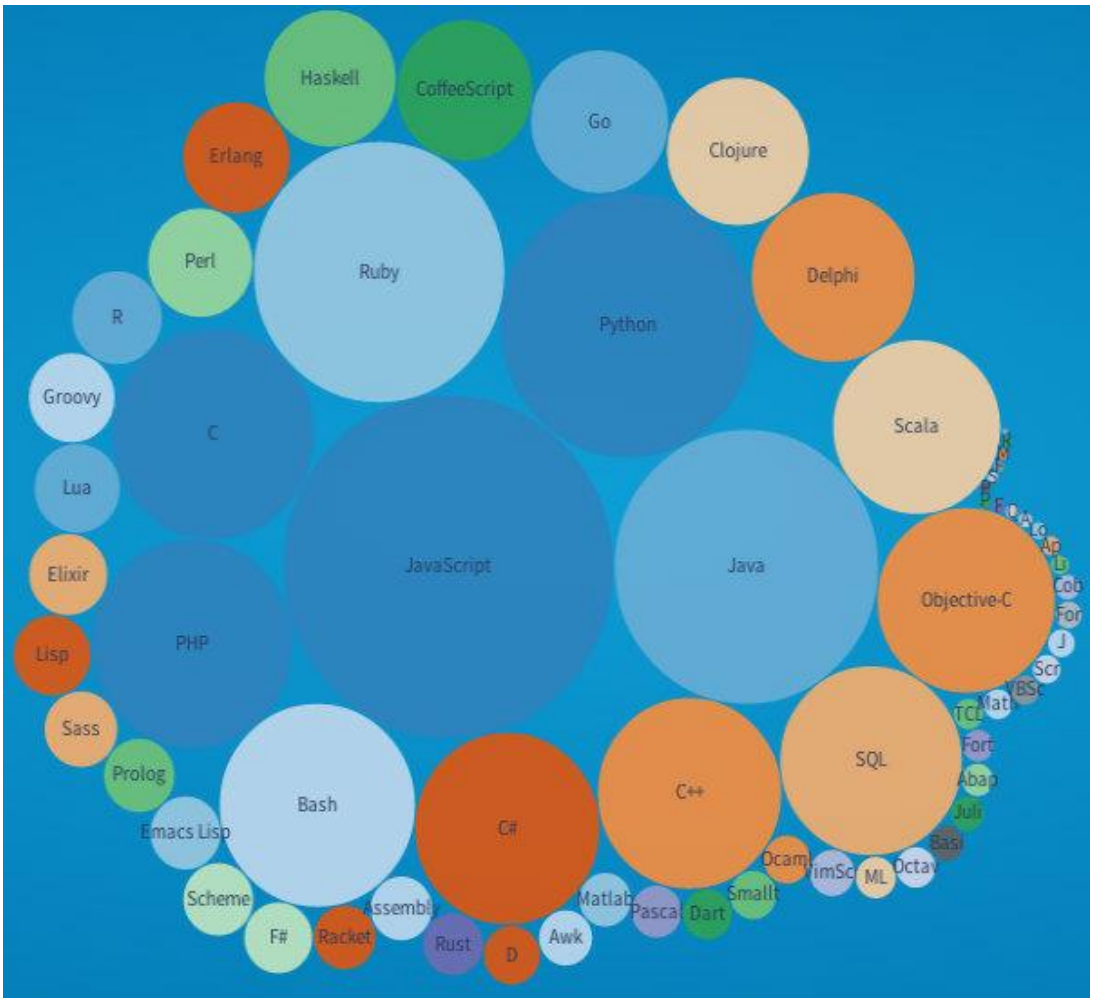

Fig. 1.9 Crecimiento Lenguajes de Programación 2013, Índice *TIOBE*.

2014: El año del *Responsive Web Design*.

"The control which designers know in the print medium, and often desire in the web medium, is simply a function of the limitation of the printed page. We should embrace the fact that the web doesn't have the same constraints, and design for this flexibility. But first, we must 'accept the ebb and flow of things.'."

(Allsopp, 2000).

La llegada del *iPhone*, en 2007, implicó un cambio en la forma de visualizar la web; nos dio un teléfono inteligente que podía mostrar las páginas web a colores tal y como lo hacíamos en la Computadora tradicional además los usuarios podían acercarse y navegar al contenido con la interacción de sus dedos a cualquier parte del sitio.

El poder creciente de los dispositivos móviles ha llevado a un aumento en las capacidades de los navegadores web, no sólo en teléfonos inteligentes de gama alta, incluso en dispositivos de gama baja se incluyen características avanzadas en sus navegadores web.

Teniendo en cuenta este aumento en la capacidad del navegador móvil, surgió un nuevo concepto de diseño web, el "*Responsive Web Design*" mencionado por primera vez el 25 de mayo de 2010 por Ethan Marcotte, en éste articulo del sitio web *alistapart.com*.

> *"In recent years, I've been meeting with more companies that request 'an iPhone website' as part of their project. It's an interesting phrase: At face value, of course, it speaks to mobile WebKit's quality as a browser, as well as a powerful business case for thinking beyond the desktop. But as designers, I think we often take comfort in such explicit requirements, as they allow us to compartmentalize the problems before us. We can quarantine the mobile experience on separate subdomains, spaces distinct and separate from 'the non-iPhone website.' But what's next? An iPad website? An N90 website? Can we really continue to commit to supporting each new user agent with its own bespoke experience? At some point, this starts to feel like a zero sum game. But how can we—and our designs—adapt?*

Let's consider an example design. I've built a simple page for a hypothetical magazine; it's a straightforward two-column layout built on a fluid grid, with not a few flexible images peppered throughout. As a long-time proponent of non-fixed layouts, I've long felt they were more 'future proof' simply because they were layout agnostic. And to a certain extent, that's true: flexible designs make no assumptions about a browser window's width, and adapt beautifully to devices that have portrait and landscape modes."

(Marcotte, 2010).

Las herramientas que permiten el *Responsive Web Design* han estado en movimiento desde años atrás. En abril de 2001, la primera propuesta del *W3C* para los medios de consulta (*media queries*) a la web se publicó de forma teórica, a partir de 2004 los navegadores de la época comenzaron a apoyar la propuesta, con esta respuesta en octubre de 2005 la *W3C* lanza un *draft* sobre mejores prácticas en web móvil y en 2009 se comenzó a implementar en los navegadores de teléfonos inteligentes y las tabletas. Para el 7 de junio de 2011, un año después de publicar su artículo Marcotte sentaría las bases del *Responsive Web Design* en su libro homónimo al concepto.

"From mobile browsers to netbooks and tablets, users are visiting your sites from an increasing array of devices and browsers. Are your designs ready? Learn how to think beyond the desktop and craft beautiful designs that anticipate and respond to your users' needs. Ethan Marcotte will explore CSS techniques and design principles, including fluid grids, flexible images, and media queries, demonstrating how you can deliver

a quality experience to your users no matter how large (or small) their display."

(Keith, 2011).

El *Responsive Web Design* o traducido al español como "Diseño Web Sensible" es una respuesta humana a los avances tecnológicos de los dispositivos actuales.

El término "sensible" se utilizó por primera vez en la fusión del arte y la arquitectura, en referencia a los espacios que están influenciados por sus ocupantes y de las formas en que interactúan con ella. Una habitación sin uso podría reducirse para dar cabida a la habitación de al lado. Del mismo modo, un sitio web debe diseñarse y adaptarse a las necesidades del lector.

En pocas palabras, Diseño Web Sensible es la capacidad que tiene un sitio web para ser visualizado correctamente independientemente del dispositivo que lo consuma. No se trata exclusivamente de reacomodar la información sino de presentar el contenido de manera óptima para el dispositivo que el usuario este usando.

El principal beneficio es brindar una mejor experiencia de accesibilidad y usabilidad al visitante del sitio. Cualquiera que haya tratado de usar un sitio no optimizado para móviles desde un teléfono ha experimentado la frustración de desplazamiento y zoom para encontrar lo que necesita. Por tal motivo es fundamental mantener la flexibilidad y la experiencia del usuario en nuestros sitios. Otro beneficio importante es que un enfoque sensible alivia la necesidad de crear múltiples versiones de un sitio web, ahorrando tiempo de desarrollo y presupuesto, incluso en algunos casos un sitio con diseño web sensible puede ahorrar la necesidad de crear una aplicación nativa.

Para 2012 con las contribuciones que *HTML5* trajo, el concepto del *Responsive Web Design* comenzó a popularizarse y su práctica no demoró en disiparse por la web. Sin embargo como todo lo nuevo implica rechazo, experimentación y aceptación, durante el 2013 aumento el número de sitios sensibles y también el número de publicaciones que lo criticaban, de hecho, si en este momento el lector googleara lo siguiente: "responsive web design bad practice", seguramente encontraría diferentes artículos y publicaciones que mencionan los efectos negativos y las malas prácticas del *Responsive Web Design*.

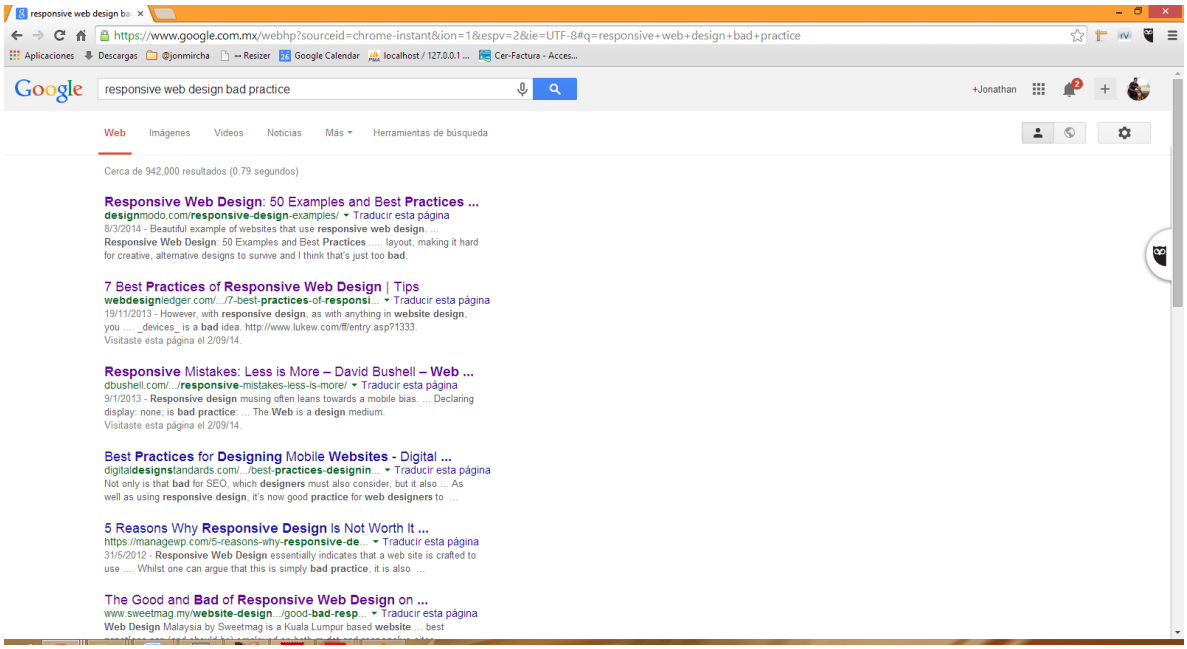

Fig. 1.10 Búqueda *Google*: *responsive web design bad practice*.

Como ya se menciono, al ser un concepto nuevo, implica que tenga simpatizantes, detractores y que se haga mucha experimentación, y eso fue lo que paso durante el 2013, si el lector revisa todas las publicaciones generadas por la búsqueda mencionada en la figura anterior, se podrá dar cuenta que todas estas publicaciones que rechazan o critican al *Responsive Web Design*, la mayoría se generaron

en el 2013, a continuación se listan 5 artículos agregando su fecha de publicación:

- *5 Reasons Why Responsive Design Is Not Worth It* (31-05-12).
- *Responsive Mistakes: Less is More* (09-01-13).
- *Responsive Web Design is Not the Future* (01-04-13).
- *5 Reasons Why Responsive Web Design Sucks for Ecommerce Businesses* (06-09-2013).
- *The 8 biggest responsive web design problems* (17-12-13).

De estos 5 artículos podemos ver que 4 fueron publicados durante el 2013. La mayoría de estos artículos critican al *Responsive Web Design*, por malas prácticas que los diseñadores fueron adquiriendo con esta nueva filosofía de diseño web.

Entre las principales podemos destacar tres:

1. **Centrarse en el contenido y no en el dispositivo:** Los diseñadores piensen que *Responsive Web Design* es simplemente adaptar un sitio web de escritorio a dispositivos móviles. Si recordamos la definición que líneas atrás esta investigación dio sobre el concepto: "Diseño Web Sensible es la capacidad que tiene un sitio web para ser visualizado correctamente independientemente del dispositivo que lo consuma. **No se trata exclusivamente de reacomodar la información sino de presentar el contenido de manera óptima para el dispositivo que el usuario este usando**". La parte que se ha puesto en negrita, es justamente el error que comenten muchos diseñadores: centrarse en el contenido y no en el dispositivo que consume el contenido, es decir, lo que tratan de hacer, es

reacomodar y reajustar el contenido que se ve en computadora para tabletas y teléfonos inteligentes, y puede que para algunos sitios que no tengan grandes cantidades de información o componentes tecnológicos (librerías de código, efectos de animación, componentes para llevar a cabo cierta tarea, *API's* o implementaciones *HTML5*, etc.) se pueda, pero para aquellos sitios que consumen grandes cantidades de información o que usan y/o abusan de componentes tecnológicos, tratar de mostrar todo ese contenido en un móvil, sin pensar en las capacidades del dispositivo, podría repercutir en su rendimiento dejando una mala experiencia para el usuario. **SI TODO ES IMPORTANTE NADA ES IMPORTANTE**.

2. **Ocultar contenido en vez de cargar el contenido necesario**: Uno de los abusos más grandes de los diseñadores con el *Responsive Web Design*, es el uso del atributo *CSS* "*display:none*", para ocultar contenido, el uso de esta regla *CSS* es una mala práctica pues el dispositivo lo único que hace es ocultar el contenido de la visualización del navegador, pero éste sigue cargando el contenido, por ejemplo si en un documento *HTML* se tiene la inclusión de un mapa de *Google*, éste objeto aproximadamente pesará 1MB, si se decide que en computadoras de escritorio y tabletas el mapa se visualice y que para teléfonos inteligentes no, y se aplica la regla *CSS* "*display:none*", lo único que se estará haciendo es ocultar el mapa en los teléfonos, pero el dispositivo seguirá consumiendo 1MB de contenido, entonces con ello sólo se está haciendo *Responsive Web Design* a nivel visual, pero el contenido no se está optimizando, en el siguiente capítulo se explicará cómo evitar esta mala práctica.

3. **No optimizar los recursos multimedia:** Por elemento multimedia se debe entender, todo aquello que no es texto, por ejemplo las imágenes, los videos, los mapas, contenidos externos como *pdfs*, *applets*, *swf*, *iframes*, etc. Dichos elementos deben de adaptarse y editarse dependiendo la resolución del dispositivo que lo consuma, un error típico de los diseñadores sobre todo con las imágenes es cargar un solo archivo para todos los dispositivos e irlo adaptando al tamaño de su contenedor, si la imagen es más pequeña que la resolución en que se muestra, esta se verá pixeleada, si es más grande que la pantalla entonces, el dispositivo estaría descargando un archivo más grande de tamaño de lo que podría visualizar. Otro aspecto importante en los objetos multimedia es que conforme las tecnologías web han avanzando, el hardware de los equipos también, actualmente tenemos dispositivos (de escritorio y móviles) que tienen mayor densidad de pixeles (lo que *Apple* bautizó como *Retina Display* en 2012), es decir donde antes existía un pixel, ahora caben 2 o 4 dependiendo de la resolución, esto lo que hace es que el texto se vea muy legible, pero las imágenes, sino están optimizadas para estas pantallas se ven borrosas, en el siguiente capítulo se explicará cómo hacer esta optimización de los elementos multimedia.

Con lo anterior, la respuesta de los expertos que difundieron y apoyaron el *Responsive Web Design*, fue inmediata. En marzo de 2013 Justin Avery, desarrollador web australiano, funda el sitio *ResponsiveDesignIs*, cuyo objetivo es ser difundir y publicar recursos, artículos y todo contenido orientado a realizar buenas prácticas de *Responsive Web Design*:

> *"Responsive design is the combination of flexible grids, flexible imagesand media queries. We help take the complexity out of*

responsive design with details on Design, Development and Strategy. Read our Articles for new tips, visit the Resources for helpful tools or see what others are doing."

(Avery, 2013).

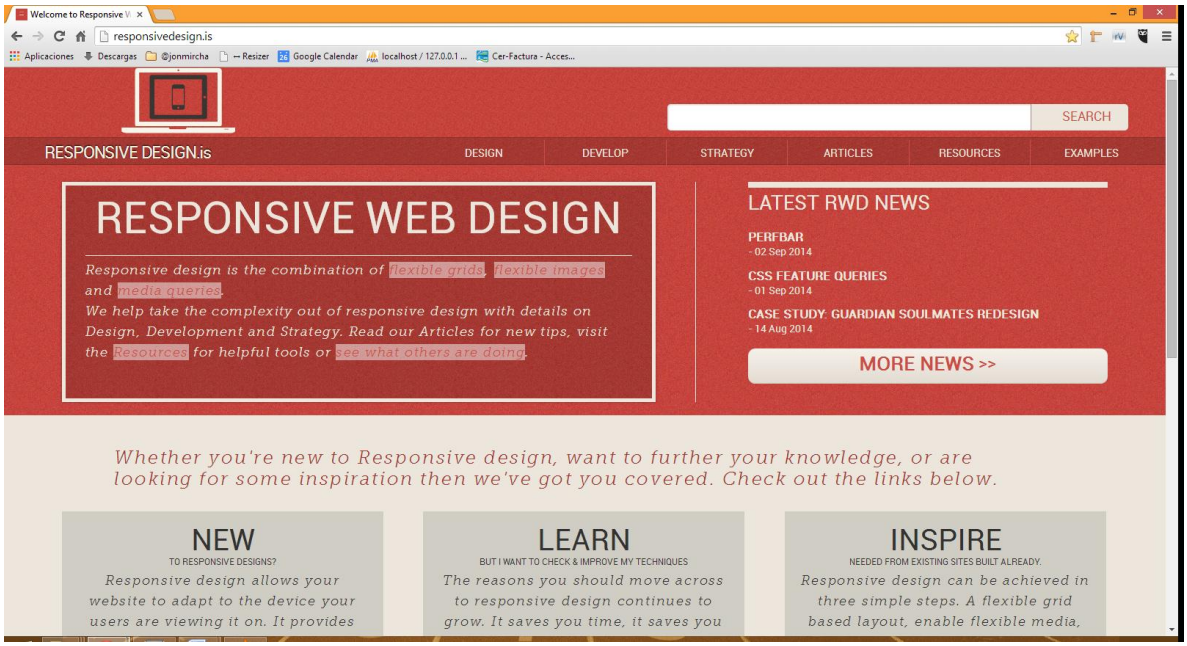

Fig. 1.11 Sitio *responsivedesign.is*.

Esta iniciativa tuvo buena aceptación entre la comunidad de diseñadores web responsive y pronto se comenzó a generarse contenido de calidad en buenas prácticas surgiendo un nuevo concepto el "***Responsible Responsive Design***", que en síntesis es hacer *Responsive Web Design* de forma responsable, es decir, ejecutando buenas prácticas. Para finales de 2013 y principios de 2014, el concepto comenzó a extenderse, tanto así que la gente de *A List Apart* ya prepara un libro que saldrá para finales del 2014. Con esta consolidación en buenas prácticas podemos considerar al 2014 como el año del *Responsive Web Design*.

Capítulo II: ***RESPONSIBLE RESPONSIVE DESIGN*.**

> *"You're designing responsively. But are you doing it* responsibly*?*
>
> *While responsive design has immeasurably improved multi-device and multi-browser visual layout, many responsive development challenges remain. How does your design adapt to changing bandwidth rates and device capabilities? How do you serve the right content in the right context? What does mobile really mean? Harnessing deep experience in the evolution of responsive web design."*
>
> (Jehl, 2014).

***Responsive Web Design*.**

Como se mencionó en el capítulo anterior, es la capacidad que tiene un sitio web para ser visualizado correctamente independientemente del dispositivo que lo consuma. No se trata exclusivamente de reacomodar la información sino de presentar el contenido de manera óptima para el dispositivo que el usuario este usando.

No se trata de la construcción de sitios web móviles. Marcotte esbozó un conjunto de principios que permiten a un sitio web ser lo suficientemente flexible como para funcionar bien en diferentes resoluciones, de manera fluida para adaptarse a la pantalla que lo esté consumiendo ya sea teléfono, tableta o equipo de escritorio.

> *"Fluid grids, flexible images, and media queries are the three technical ingredients for responsive web design, but it also requires a different way of thinking. Rather than quarantining*

our content into disparate, device-specific experiences, we can use media queries to progressively enhance our work within different viewing contexts. That's not to say there isn't a business case for separate sites geared toward specific devices; for example, if the user goals for your mobile site are more limited in scope than its desktop equivalent, then serving different content to each might be the best approach.

But that kind of design thinking doesn't need to be our default. Now more than ever, we're designing work meant to be viewed along a gradient of different experiences. Responsive web design offers us a way forward, finally allowing us to 'design for the ebb and flow of things'."

(Marcotte, 2010).

Los tres principios que Marcotte propuso son:

1. **Grillas fluídas**, es decir, una maquetación pensada en porcentajes y proporciones en vez de dimensiones fijas.
2. **Imágenes flexibles**, que se adapten al dispositivo que las consuma, pero como lo mencionamos anteriormente, no sólo basta con tener imágenes, todo objeto que no sea texto debe tener la capacidad de adaptarse.
3. El uso de la regla ***media queries*** de *CSS*, que es lo más cercano a una condicional de programación que tenemos en las hojas de estilo en cascáda, y que nos permite preguntar por las características (ancho, alto, color, orientación, etc.) del medio (dispositivo) que visualiza el sitio.

Estos son los pilares en los que se sustentan las bases del *Responsive Web Design*, las cuales son solo recomendaciones y buenas prácticas de Hojas de Estilo en Cascada *CSS*, pero esto no es suficiente, para poder lograr un *Responsible Responsive Design* es necesario tener buenas prácticas de estructura y contenido (*HTML*), diseño y presentación (*CSS*) y funcionalidad e interactividad (*JavaScript*).

A continuación, la presente investigación propone una serie de buenas prácticas para lograr un Responsible Responsive Design.

Diez tips sobre *Responsible Responsive Design*.

Nota: La presente investigación es una guía de buenas prácticas en *Responsive Web Design*, es decir, tips y consejos que le sirvan al diseñador web de la era móvil para que sus sitios sean responsables con el contenido visualizado en los dispositivos, con esto se queda en el entendido que el usuario lector ya tiene conocimientos de *HTML*, *CSS* y *JavaScript*, ya que esta investigación NO pretende ser un libro que enseñe conceptos, reglas y definiciones de estas tecnologías, sino como aprovechar sus recursos para hacer sitios responsables, por lo que cuando se mencione algún concepto de maquetación o programación en consideración al usuario lector que no conozca el término se proporcionaran links donde podrá ahondar más en el tema.

Aclarando lo anterior, a continuación se enlistan 10 consejos para lograr un *Responsible Responsive Design*:

I. Estructura de documento *responsive*.

Todos los documentos que elaboremos en un sitio web *responsive* deberán tener la estructura *HTML5* con los siguientes elementos:

- *DOCTYPE HTML* que indica que es un tipo de documento *HTML5*.
- La etiqueta *HTML* con su atributo *lang* especificando el idioma del documento.
- La etiqueta *meta viewport*, encargada del correcto funcionamiento del *responsive web design* en los dispositivos.
- La etiqueta *meta charset* especificando el juego de caracteres que tendrá el documento.
- Las hojas de estilos en cascada que tengamos enlazadas al documento siempre deberán invocarse en la cabecera (antes de *</head>*) para el correcto renderizado del contenido en el navegador y dar un efecto de que el sitio carga rápido.
- Los scripts que tengamos enlazados al documento siempre deberán invocarse al final del contenido (antes de *</body>*) ya que si los colocamos antes pueden bloquear el renderizado del contenido en el navegador.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=2, user-scalable=yes" />
    <meta charset="UTF-8" />
    <title>Responsive Design</title>
    <link rel="stylesheet" href="css/estilos.css" />
</head>
<body>
    <section>
        Aquí el contenido
    </section>
    <script src="js/scripts.js"></script>
</body>
</html>
```

Fig. 2.1 Estructura de documento *responsive*.

II. Configuración de la etiqueta *meta viewport*.

En el punto anterior se menciona el uso de la etiqueta *meta viewport* en la estructura del documento *responsive*. Dicha etiqueta es importante porque si la omitimos el sitio no se visualizará correctamente en los dispositivos móviles, como a continuación se muestra:

Fig. 2.2 Documento sin *viewport* y Fig. 2.3 Documento con *viewport*.

Como podemos apreciar en la imagen 2.2 que no tiene *viewport* configurado, el documento se ve como si fuera una versión para computadoras de escritorio, a diferencia de la imagen 2.3 donde el *viewport* está configurado y adaptado para el dispositivo que lo visualiza (estás capturas de pantalla fueron tomadas con un *Samsung Galaxy S4* en modo vertical).

El *viewport* fue introducido por *Apple* en *safari* móvil, para ayudar a mejorar la presentación de los sitios en dispositivos como el *iPhone*, *iPod Touch* o *iPad*; posteriormente se volvió un estándar en todos los navegadores.

La etiqueta *viewport* es el área visible de nuestro navegador y nos permite definir el ancho, alto y escala del área usada por el navegador para mostrar el contenido.

El *viewport* tiene 5 atributos importantes:

- ***width*:** define el ancho visible, suele usarse la constante ***device-width*** para que se adapte al ancho del dispositivo.
- ***height:*** define el alto visible, suele usarse la constante ***device-height*** para que se adapte al alto del dispositivo.
- ***initial-scale*:** define la escala inicial a la que deberá visualizarse el contenido, siendo 1 el 100%.
- ***maximum-scale:*** define la escala máxima a la que se podrá hacer zoom al contenido, por ejemplo si se configura 3 significa que se podrá hacer un zoom de hasta el 300%.
- ***user-scalable*:** define si el usuario puede escalar o no el contenido, sus valores son yes por defecto y no para bloquear esta propiedad.

De los 5 atributos mencionados, hay 2 que son muy importantes definir el ancho y la escala inicial, para que el contenido se adapte al tamaño del dispositivo, adicionalmente se recomienda que no se especifique la máxima escala, ni bloquear la capacidad de escalar el contenido, ya que independientemente de que el documento se diseñe con responsive, algunos usuarios con capacidades visuales diferentes, podrían tener problemas al ver el contenido y desearían aumentar el tamaño del contenido, si definimos una escala máxima o bloqueamos la capacidad de escalar el contenido produciremos en estos usuarios una mala y frustrante experiencia de usuario. La propiedad del alto es opcional

definirla, se utilizara cuando necesitemos condicionar el alto del contenido.

Por lo anterior la etiqueta meta viewport tendría que definirse de la siguiente manera:

<meta name="viewport" content="width=device-width,initial-scale=1" />

III. Maquetación fluida.

Antes de la llegada de los dispositivos móviles, los sitios web se construían con anchos fijos, particularmente para 2 tipos de resoluciones: 800 x 600 píxeles y 1024 x 768px. Sin embargo, las dimensiones de ancho fijo en píxeles son inflexibles y no pueden adaptarse a diferentes resoluciones. En cambio, cuando se usan proporciones relativas usando valores porcentuales se permite que el diseño fluya de forma natural con la resolución del dispositivo.

En síntesis debemos de olvidarnos de maquetar con pixeles y comenzar a pensar en proporciones relativas a la pantalla (porcentajes).

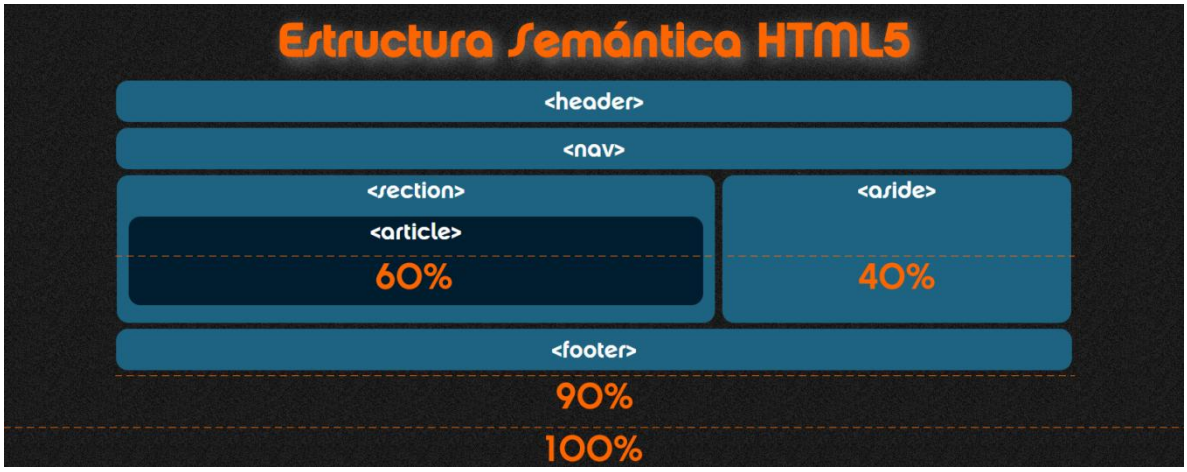

Fig. 2.4 Maquetación Fluida.

A nivel de reglas *CSS* es recomendable dejar de utilizar la propiedad float, que es con la que se venía trabajando en el proceso de acomodar los elementos de un documento *HTML*, su desventaja es que rompe la

disposición natural del contenido *HTML* y superpone los elementos ya sea a la izquierda o a la derecha, obligando a limpiar la flotación de los elementos que se tengan por debajo de los que se han flotado.

En cambio es recomendable comenzar a emplear nuevas formas de maquetado a través de los diferentes valores de la propiedad *display*, como es el caso de: *inline-block*, *inline-table* o *flex*. Dichos valores son más amigables con la filosofía de trabajo del *Responsive Web Design*.

A continuación se enlistan enlaces con más información y ejemplos prácticos sobre los valores *CSS* aquí mencionados.

- Propiedad *float*.
- Propiedad *display*.
- Valor *inline-block*.
- Valor *inline-table*.
- Valor *flex*.

IV. Objetos flexibles.

En la creación de diseños fluidos donde se utilizan porcentajes en lugar de anchos de píxel fijo adaptar el contenido para ajustarse a la pantalla de forma natural es un aspecto muy importante.

En la web, tenemos 2 tipos de contenido principalmente: Texto y lo que no es texto, a los que llamaremos objetos.

Entonces dentro de la categoría de objeto entran las imágenes, videos, audios, *canvas*, *svg*, contenido externo insertado en un *iframe* como un video, mapa, documento *pdf*, *feeds* de redes sociales, etc.

A diferencia del texto, los objetos no pueden por defecto fluir de manera natural conforme a la disposición de una plantilla *responsive*, el problema es que los objetos tienen un valor determinado de ancho y

alto fijo, entonces para que se vuelvan flexibles es necesario adaptar su tamaño al ancho de su elemento contenedor, esto se logra con el atributo *CSS* de máxima anchura igualado a 100%:

max-width:100%;

Con esto, los objetos fluirán al tamaño de su elemento contenedor volviéndose flexibles (estás capturas de pantalla fueron tomadas con un *Samsung Galaxy S4* en modo vertical).

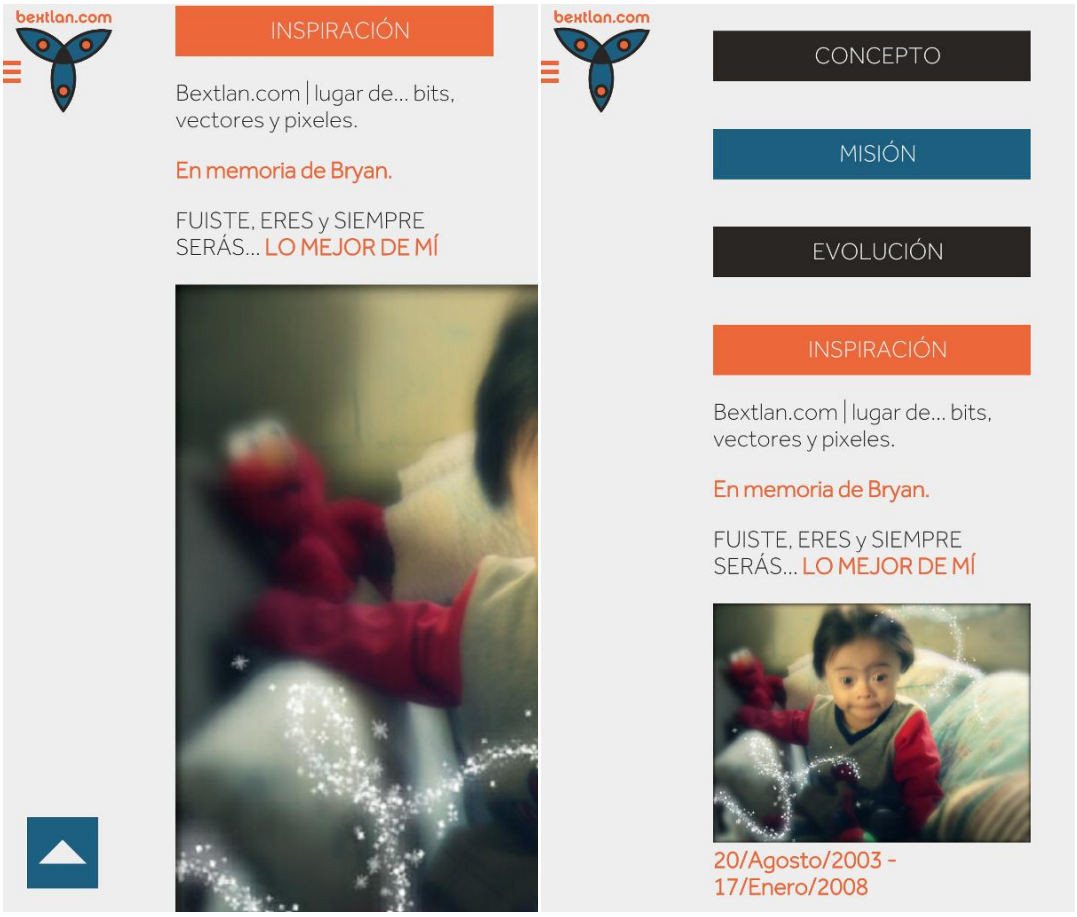

Fig. 2.5 Objeto no flexible y Fig. 2.6 Objeto flexible.

V. Media Queries.

Desde la especificación de *CSS 2.1*, las hojas de estilo han tenido cierto grado de capacidad para el reconocimiento de dispositivos mediante el uso de tipos de medios. Por ejemplo:

<link rel="stylesheet" href="print.CSS" media="print" />

La línea anterior nos permite aplicar reglas *CSS* separadas para cuando nuestro documento se imprima.

Sin embargo, mientras que la especificación incluye una colección de tipos de medios, dirigido a identificar clases de dispositivos, estos a menudo eran ignorados por los navegadores y dispositivos.

Con *CSS*3, la *W3C* perfeccionó y mejoró los tipos de medios con características multimedia y con la capacidad de preguntar dichas características a los medios.

Esto no sólo hace posible inspeccionar el contenido que se entrega al dispositivo, sino también las características físicas reales del dispositivo.

El uso de media queries permite pedir fácilmente al navegador sus características, tales como anchura, altura, relación de aspecto y la orientación. La sintaxis es la siguiente:

```
@media screen and (max-width:480px) and (orientation:portrait) {
/* Código CSS que se aplicará cuando se cumpla la media queries */
}
```

La línea anterior define reglas *CSS* para medios que tengan pantalla con una máxima anchura de 480px y una orientación vertical.

Es importante recordar que *CSS* no es un lenguaje de programación, sino uno de definición de estilos a través de atributos y valores, con las media queries la versión 3 nos entrega lo más cercano a un condicional de programación para tomar decisiones y así aplicar unos u otros estilos dependiendo el medio en el que nos encontremos y sus respectivas características.

La lista completa de características multimedia de la W3C, sobre media queries es la siguiente:

Nombre	Definición	Con prefijo max y min
width	El ancho del área de visualización.	SI
height	El alto del área de visualización.	SI
device-width	La anchura de la superficie de representación del dispositivo.	SI
device-height	La altura de la superficie de representación del dispositivo.	SI
orientation	Acepta los valores de *portrait*(vertical) y *landscape*(horizontal).	NO
aspect-ratio	Relación entre la anchura de la zona de visualización de toda su altura. Por ejemplo: en una computadora de escritorio, se podría consultar si la ventana del navegador se encuentra en una relación de 16:9.	SI
device-aspect-ratio	Proporción de anchura de la superficie de representación del dispositivo a través de su altura. Por ejemplo: en una computadora de escritorio, se podría consultar si la pantalla está en una relación de 16:9.	SI
color	El número de bits por componente de color del dispositivo. Por ejemplo, un dispositivo de color de 8 *bits* podría pasar con éxito una consulta de (*color: 8*). Dispositivos de color no deben devolver un valor de 0.	SI

color-index	El número de entradas en la tabla de búsqueda de color del dispositivo de salida. Por ejemplo, *@media screen and (min-color-index:256)*	SI
monochrome	Al igual que el color, la característica monocromática permite ponernos a prueba el número de bits por píxel en un dispositivo monocromático.	SI
resolution	Pruebas de la densidad de los píxeles en el dispositivo, Por ejemplo, screen and (*resolution:72dpi*) or screen and (*max-resolution:300dpi*).	SI
scan	Para la navegación basada en tv, mide si el proceso de exploración es progresiva o escaneo.	NO
grid	Comprueba si el dispositivo es una pantalla basada en la red, como los teléfonos con funciones con una fuente de ancho fijo. Se puede expresar simplemente como (*grid*).	NO

Fig. 2.7 Tabla de atributos media queries *CSS3*.

El sitio *http://mediaqueri.es* proporciona una visión clara de cómo los creadores de sitios web están utilizando estas técnicas para ofrecer sitios sensibles.

VI. Puntos de interrupción (*breakpoints*).

Con la lista de propiedades que se dio en la figura 2.7 puede que el usuario lector en este momento se sienta confundido y agobiado sobre cuál o cuáles atributos usar para aplicar el *Responsive Web Design*, esto dependerá de las características y necesidades de nuestro proyecto, por ejemplo si estamos desarrollando un trabajo tipo

videojuego o presentación de diapositivas a lo mejor necesitaremos validar el alto, la anchura y la orientación del dispositivo, pero si estamos desarrollando un sitio web convencional generalmente sólo tendríamos que validar la anchura, para ello es importante identificar los puntos de interrupción.

Marcotte en su libro nos propone lo siguiente:

> "Nuestro proceso de diseño se inicia mediante encuestas a los diferentes dispositivos para los que estamos planeando diseñar. Desde la investigación, vamos a compilar una lista de los puntos de interrupción de resolución: Los anchos horizontales que necesitaremos para dar cabida en nuestro Responsive Web Design.
>
> Eso no quiere decir que las resoluciones por encima o por debajo de este umbral serán ignorados, o que no vamos a dar cabida a algunos dispositivos cuyas resoluciones no se enumeran. (Después de todo, el Responsive Web Design se basa en una cuadrícula flexible, por lo que hay cierta independencia de la resolución en ello.) Pero la construcción de una lista ayuda a definir un ámbito para nuestros esfuerzos, que nos permite identificar los dispositivos más utilizados por nuestro público, y la mejor manera de probar en contra de sus respectivas resoluciones.
>
> Nuestra lista podría ser:

320 pixeles	Para dispositivos con pantallas pequeñas, como los teléfonos en modo vertical.
480 pixeles	Para dispositivos con pantallas pequeñas, como los teléfonos, en modo horizontal.

600 pixeles	Tabletas pequeñas, como el *Amazon Kindle* (600×800) y *Barnes & Noble Nook* (600×1024), en modo vertical.
768 pixeles	Tabletas de diez pulgadas como el *iPad* (768×1024), en modo vertical.
1024 pixeles	Tabletas como el *iPad* (1024×768), en modo horizontal, así como algunas pantallas de ordenador portátil, *netbook*, y de escritorio.
1200 pixeles	Para pantallas panorámicas, principalmente portátiles y de escritorio.

Fig. 2.8 Tabla de puntos de interrupción (*breakpoints*) de Marcotte.

Con esa lista en la mano, es hora de que el diseño comience a ser serio."

(Marcotte, 113).

Con la tabla que Marcotte nos propone prácticamente están cubiertos teléfonos inteligentes, tabletas, phablets(se les llama así por su tamaño, son más grandes que un teléfono convencional, pero más pequeños que una tableta promedio, por ejemplo el *Galaxy Note*), *laptops*, *netbooks*, computadoras de escritorio y pantallas, tanto en modo vertical como horizontal, por lo que con estos 6 puntos de interrupción sólo tendríamos que validar un solo atributo en las *media queries*: el ancho.

De cualquier forma en el sitio *screensiz.es*, el usuario puede estar revisando constantemente las características físicas con las que se fabrican los dispositivos (teléfonos, tabletas y equipos de escritorio).

En el siguiente punto se revisará como utilizar estos puntos de interrupción dependiendo del enfoque responsive con el que se decida trabajar: Centrándose en el móvil (*Mobile First*) o en el equipo de escritorio (*Desktop First*).

VII. Enfoques *responsive*: *Mobile First vs Desktop First*.

Los beneficios del *Responsive Web Design* parecen claros, una mejor experiencia de usuario y el uso más eficiente de los recursos de los dispositivos y el acceso al contenido.

Al implementar el *Responsive Web Design* se requieren ciertos cambios en el enfoque tradicional del desarrollo de un sitio. Idealmente, el proceso del *responsive* debe comenzar a partir del diseño de la pantalla del dispositivo móvil y de ahí trabajar para la tableta y el equipo de escritorio, este enfoque se conoce como ***Mobile First***, ya que la planeación del proyecto se centra primero en la interfaz móvil y de ahí se va implementando progresivamente para interfaces más grandes, dicho enfoque también recibe el nombre de ***Progressive Enhancement*** (mejora progresiva) ya que de manera gradual va adaptando el sitio a las capacidades del dispositivo.

Sin embargo, en la práctica este enfoque puede ser todo un reto, las expectativas y hábitos de pensamiento de las personas con la web en su mayoría son a través de su experiencia de escritorio, incluso podrían llegar a sentirse incómodos al proporcionar retroalimentación sobre el móvil como su primera experiencia con un nuevo diseño. Las expectativas para móviles basados en experiencias de escritorio son un desafío importante.

Para contrarrestar y hacer amigable el efecto del cambio de paradigma que plantea el *Responsive Web Design* centrándose en el móvil, existe otro enfoque, que consiste en seguir haciendo el diseño web como se ha hecho desde sus inicios: centrándose en el equipo de escritorio, a este enfoque lo llamaremos ***Desktop First*** y consiste en primero diseñar la interfaz para resoluciones grandes y de ahí ir degradando a las pequeñas, dicho enfoque también recibe el nombre de ***Graceful***

Degradation (degradación elegante) ya que de manera gradual va simplificando el sitio a las capacidades del dispositivo.

En la práctica este enfoque permite seguir pensando en la pantalla de escritorio e ir degradando la adaptación al dispositivo móvil.

A nivel de código *CSS* el usar un enfoque u otro implica un cambio de filosofía en el maquetado del sitio, por ello es importante definir bajo que enfoque se diseñará.

A continuación se presenta un ejemplo práctico para analizar la diferencia de los enfoques.

Fig. 2.9 Maquetación *Responsive Web Design* con enfoque *Desktop First*.

Fig. 2.10 Maquetación *Responsive Web Design* con enfoque *Mobile First*.

En las figuras anteriores se puede ver un documento *HTML* maquetado con la misma estructura, exceptuando los textos de cabecera, pié y la imagen de cabecera, el contenido es el mismo como se puede ver en el código fuente *HTML* de ambos documentos que pueden descargar en los siguientes enlaces:

- *desktop-first.html*
- *mobile-first.html*

La diferencia de ambos documentos es el enfoque con el que se hicieron, mientras que el archivo *desktop-first.html* se centro pensando en el equipo de escritorio el archivo *mobile-first.html* lo hizo con el móvil.

Si analizamos las hojas de estilo en cascada de cada archivo, que pueden encontrar en los siguientes enlaces:

- *desktop-first.css*
- *mobile-first.css*

Podemos observar que:

1. En las *media queries* se están usando los puntos de interrupción (*breakpoints*) propuestos por Marcotte, pero de diferente manera, mientras que en el archivo *desktop-first.css* se validan consultando por la máxima anchura de la pantalla (*max-width*) en el *mobile-first.css* se hacen consultando por la mínima anchura de pantalla (*min-width*).
2. Los puntos de interrupción están ordenados de forma inversa, en *desktop-first.css* del mayor al menor (1200px 1024px 768px 600px 480px 320px) y en *mobile-first.css* del menor al mayor sumándole 1 a cada breakpoint para conservar el mismo rango de

alcance que en *desktop* (321px 481px 601px 769px 1025px 1201px).

3. El código *CSS* general (por *CSS* general, entiéndase el código que esta antes de comenzar las media queries) está planeado para diferentes interfaces, si quitamos los estilos *CSS* a partir de las media queries en ambos archivos el resultado al ver los documentos *HTML* en diferentes dispositivos, sería el siguiente:

Fig. 2.11 Archivo *desktop-first.html* visto en un equipo de escritorio.

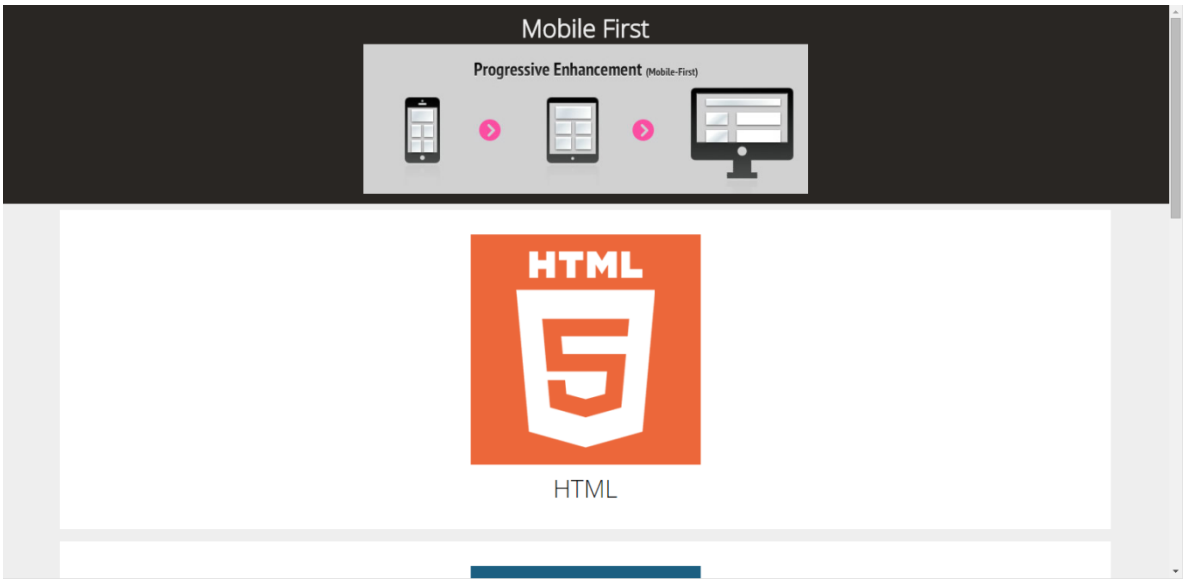

Fig. 2.12 Archivo *mobile-first.html* visto en un equipo de escritorio.

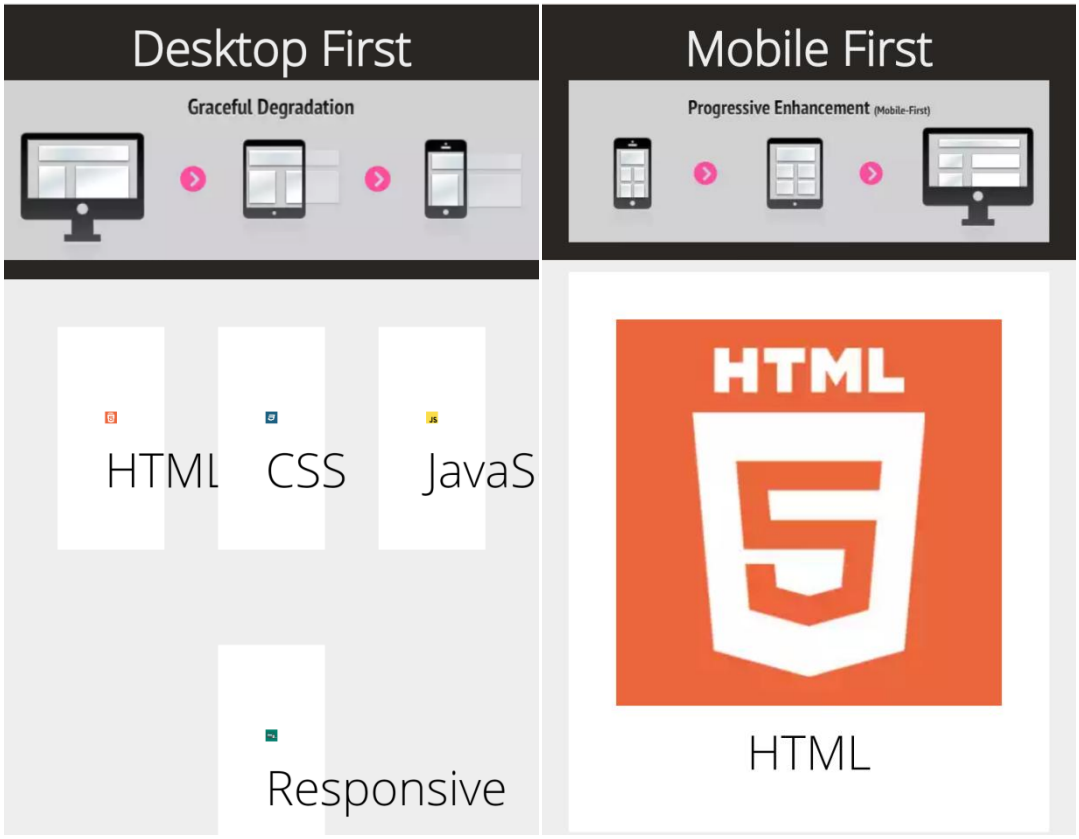

Fig. 2.13 y 2.14 Archivo *desktop-first.html* y *mobile-first.html* visto en un teléfono *Samsung Galaxy S4* en modo vertical.

Como podemos observar en la figura 2.11 y 2.13 el documento *desktop-first.html* fue diseñado centrándose en equipos de escritorio, ya que al quitar las media queries en el código *CSS*, podemos ver como en el equipo de escritorio la visualización es correcta, mientras que en el teléfono se rompe la maquetación dejando problemas en el texto y las imágenes; en cambio las figuras 2.12 y 2.14 del documento *mobile-first.html* que fue diseñado centrándose en el móvil, podemos ver como en el equipo de escritorio el contenido se ve a una columna centrada dejando mucho espacio a los lados y generando scroll vertical del navegador para poder visualizar todo el contenido, mientras que en el teléfono el contenido se sigue mostrando a una columna pero adaptado al ancho de la pantalla.

Como lo hemos observado cada enfoque tiene sus características y sus reglas para implementarlo, en el estricto sentido el concepto del *Responsive Web Design* nace con la llegada de los dispositivos móviles por lo que lo ideal sería trabajar con el enfoque *Mobile First*, pero a veces dependiendo de las características y necesidades del sitio a diseñar, podría resultar más eficiente aplicar el enfoque *Desktop First*.

Para terminar con este apartado a continuación se muestran dos tablas la primera con las características más importantes de cada enfoque y la segunda con una lista de circunstancias donde se podrían aplicar cada uno:

Responsive Web Design		
Enfoque	*Mobile First*	*Desktop First*
Técnica	*Progressive enhancement*	*Graceful degradation*
Objetivo	Equipo Móvil	Equipos de Escritorio
CSS General	Equipo Móvil (Ancho Pantalla <= 320px)	Equipos de Escritorio (Ancho Pantalla > 1200px)
Media Queries	Se valida la mínima anchura (*min-width*)	Se valida la máxima anchura (*max-width*)
Breakpoints	Del menor al mayor 321px, 481px, 601px, 769px, 1025px y 1201px	Del mayor al menor 1200px, 1024px, 768px, 600px, 480px y 320px

Fig. 2.15 Tabla Características: *Mobile First vs Desktop First*.

¿Cuándo usar?	
Mobile First	• Cuando ya tienes experiencia en responsive design. • Si el sitio a desarrollar es nuevo. • Si el sitio ya existe y se desea rediseñar. • Si la prioridad del cliente es la interfaz móvil. • Si el objetivo del sitio es que se consuma en móviles. • Si se descartará el soporte a navegadores viejos. • Con el tiempo este enfoque es el que prevalecerá.
Desktop First	• Si eres de la vieja escuela del diseño web y el responsive es nuevo para ti. • Si eres novato en responsive design. • Si el sitio ya existe y lo único que se desea es aplicarle responsive. • Si la prioridad del cliente es la interfaz de escritorio. • Si el objetivo del sitio es que se consuma en escritorio. • Si se le va a dar soporte a navegadores viejos. • Con el tiempo este enfoque tenderá a desaparecer.

Fig. 2.16 Tabla ¿Cuándo usar?: *Mobile First vs Desktop First*.

VIII. Pruebas *responsive*.

Independientemente del enfoque que el lector usuario decida utilizar para aplicar *Responsive Web Design*, es fundamental hacer pruebas a nuestros diseños, para ver como se visualizarán en los dispositivos.

Lo ideal sería tener físicamente diferentes dispositivos (equipos de escritorio, tabletas y teléfonos inteligentes) en diferentes plataformas (*Windows*, *Macintosh*, *Linux*, *WPhone*, *iOS*, *Android*, etc.) y probando con distintos navegadores (*Chrome*, *Firefox*, *IE*, *Safari*, *Opera*, etc.) para hacer pruebas directamente en ellos, desafortunadamente en la práctica a veces esto no es posible, para solventar este problema la misma comunidad en la web ha puesto a disposición diferentes herramientas para poder hacer pruebas.

Existen algunos sitios que nos ayudan a simular el tamaño de la resolución de los diferentes dispositivos móviles, por ejemplo:

En el sitio *http://mattkersley.com/responsive/* se ofrece una herramienta en la cual se tiene que ingresar la dirección url del documento que se quiere simular y la herramienta lo que hace es mostrarnos en diferentes tamaños de pantalla como se vería nuestro documento para diferentes dispositivos, considerando las siguientes medidas:

- Teléfonos pequeños (240px x 320px).
- *iPhone* (320px x 480px).
- Tabletas pequeñas (480px x 640px).
- *iPad* vertical (768px x 1024px).
- *iPad* horizontal (1024px x 768px).

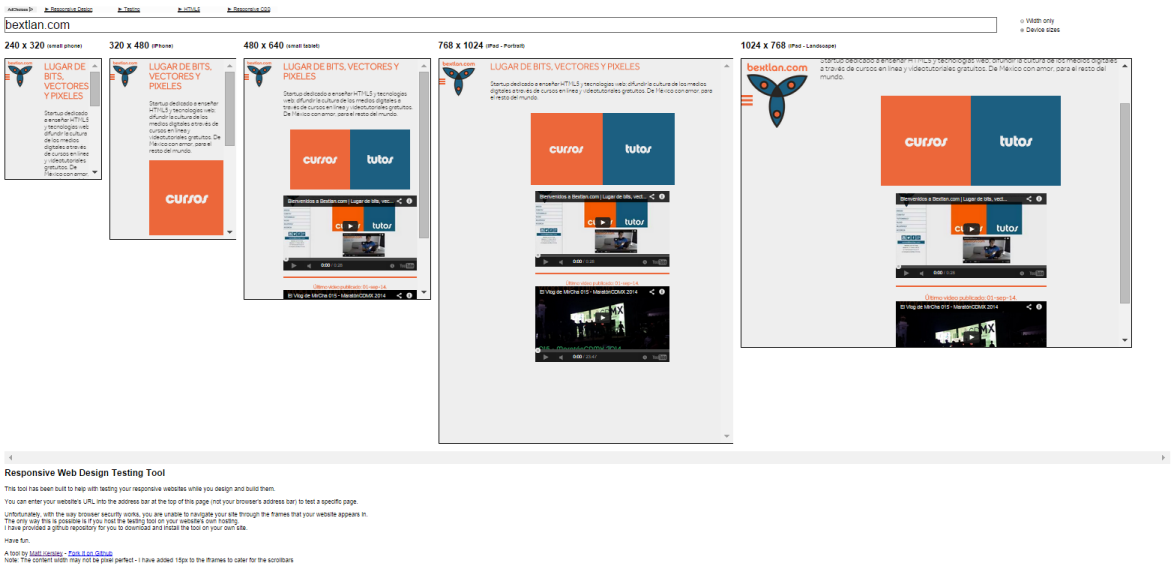

Fig. 2.17 Sitio de pruebas *responsive* *http://mattkersley.com/responsive/*.

Otro sitio que ofrece un servicio similar, inclusive simulando el sitio dentro de una imagen de dispositivo móvil es *http://www.responsinator.com/*

Fig. 2.18 Sitio de pruebas *responsive* *http://www.responsinator.com/*.

Otra opción para hacer pruebas es instalando extensiones a los navegadores web, los que más opciones tiene son *Google Chrome* y *Mozilla Firefox*.

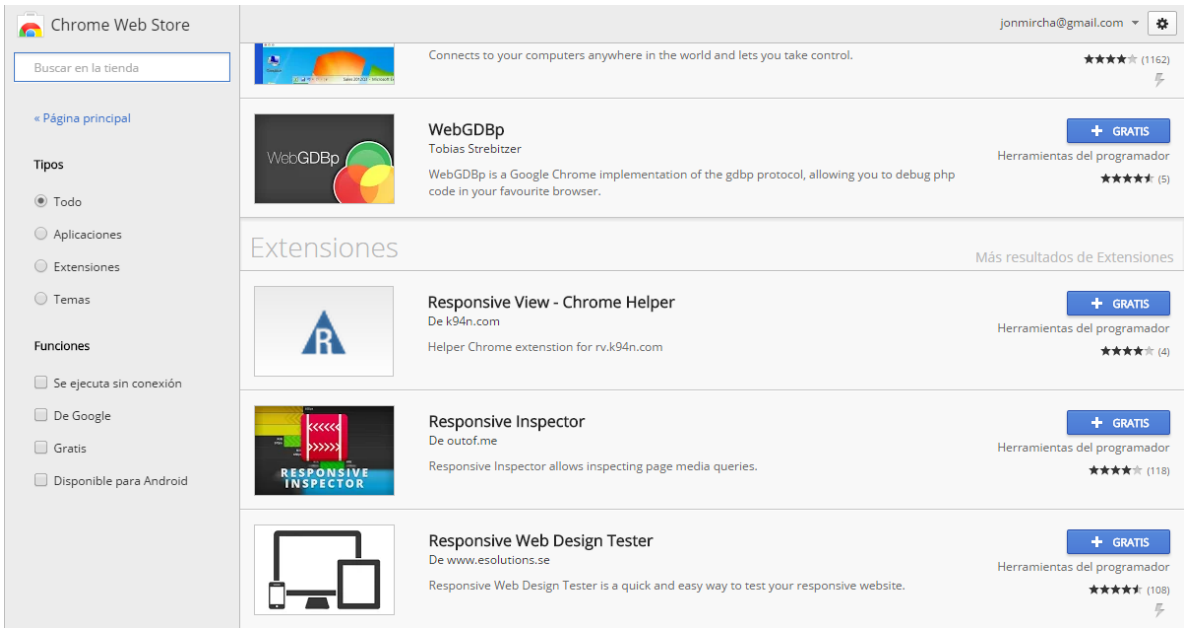

Fig. 2.19 Extensiones *de Google Chrome* para pruebas *responsive*.

Todas estas opciones son de gran ayuda si no se tienen dispositivos físicamente a la mano, sin embargo tienen un inconveniente, pueden hacer más lento nuestro proceso de trabajo, ya que para hacer pruebas en dichas opciones hay que guardar nuestro código, subirlo a un servidor y después hacer la petición desde el sitio de pruebas o activar las extensiones en los navegadores pudiendo hacer que funcione más lento.

Generalmente cuando estamos maquetando en *CSS* la dinámica de trabajo es la siguiente: escribir código *CSS* en nuestro editor de código, guardar el archivo, pasarnos del editor de código al navegador, recargar el documento *HTML* en el navegador y ver los cambios hechos, si a esta dinámica le agregamos los pasos mencionados para emplear las herramientas online o las extensiones de los navegadores, se puede volver una tarea muy tediosa.

Para evitar este retraso de tiempo Jonathan MirCha propone una técnica muy sencilla que se mencionó en un *streaming* organizado por la comunidad *CodeJobs*, *Bextlan* y el Instituto ICONOS el 08 de octubre de 2012.

Dicha técnica consiste en pintar de color un elemento *HTML* que exista en todas las secciones de nuestro sitio o proyecto y lo único que hay que hacer al ver los cambios en el navegador es redimensionar la ventana de éste y dependiendo el color con el que se pinte el elemento es la media queries en la que nos encontramos.

Basándose en los puntos de interrupción propuestos por Marcotte, tenemos 6 *breakpoints*, en cada uno se pintará el elemento que tomemos como referencia, para hacerlo más fácil se propone tomar los colores de los sistemas *RGB*(*red*, *green*, *blue*) y *CMYK*(*cyan*, *magenta*,

yellow), dependiendo del enfoque responsive con el que se decida trabajar esta es la combinación de colores propuesta:

Enfoque *Desktop First*		**Enfoque *Mobile First***	
Color original	*CSS* General	*Yellow*	*CSS* General
Red	1200	*Magenta*	321
Green	1024	*Cyan*	481
Blue	768	*Blue*	601
Cyan	600	*Green*	769
Magenta	480	*Red*	1025
Yellow	320	Color original	1201

Fig. 2.20 Tabla de color para pruebas *responsive*.

Con estos valores lo que se pretende es que independientemente del enfoque responsive con el que se decida trabajar, el rango de color se mantenga en cada punto de interrupción, quedando de la siguiente manera:

- Resoluciones iguales o mayores a 1201px el color original que tenga el elemento en la maquetación.
- Resoluciones de 1200px a 1025px color *red*.
- Resoluciones de 1024px a 769px color *green*.
- Resoluciones de 768px a 601px color *blue*.
- Resoluciones de 600px a 481px color *cyan*.
- Resoluciones de 480px a 321px color *magenta*.
- Resoluciones menores o o iguales a 320px color *yellow*.

Si vemos el código de las hojas de estilo en cascada de los ejercicios tratados en el punto VII (*desktop-first.css* y *mobile-first.css*) se puede

ver que a la etiqueta *HTML* footer se le aplico esta técnica de colores. Si observamos en el navegador los documentos *HTML* correspondientes (*desktop-first.html* y *mobile-first.html*) podemos ver como dicha etiqueta cambia de color en cada punto de interrupción como se mencionó en líneas anteriores:

Fig. 2.21 y 2.22 Documentos *desktop-firs.html* y *mobile-first.html* prueba *responsive* de color a más de 1200px (color original).

Fig. 2.23 y 2.24 Documentos *desktop-first.html* y *mobile-first.html* prueba *responsive* de color rango de 1200 a 1025px (color *red*).

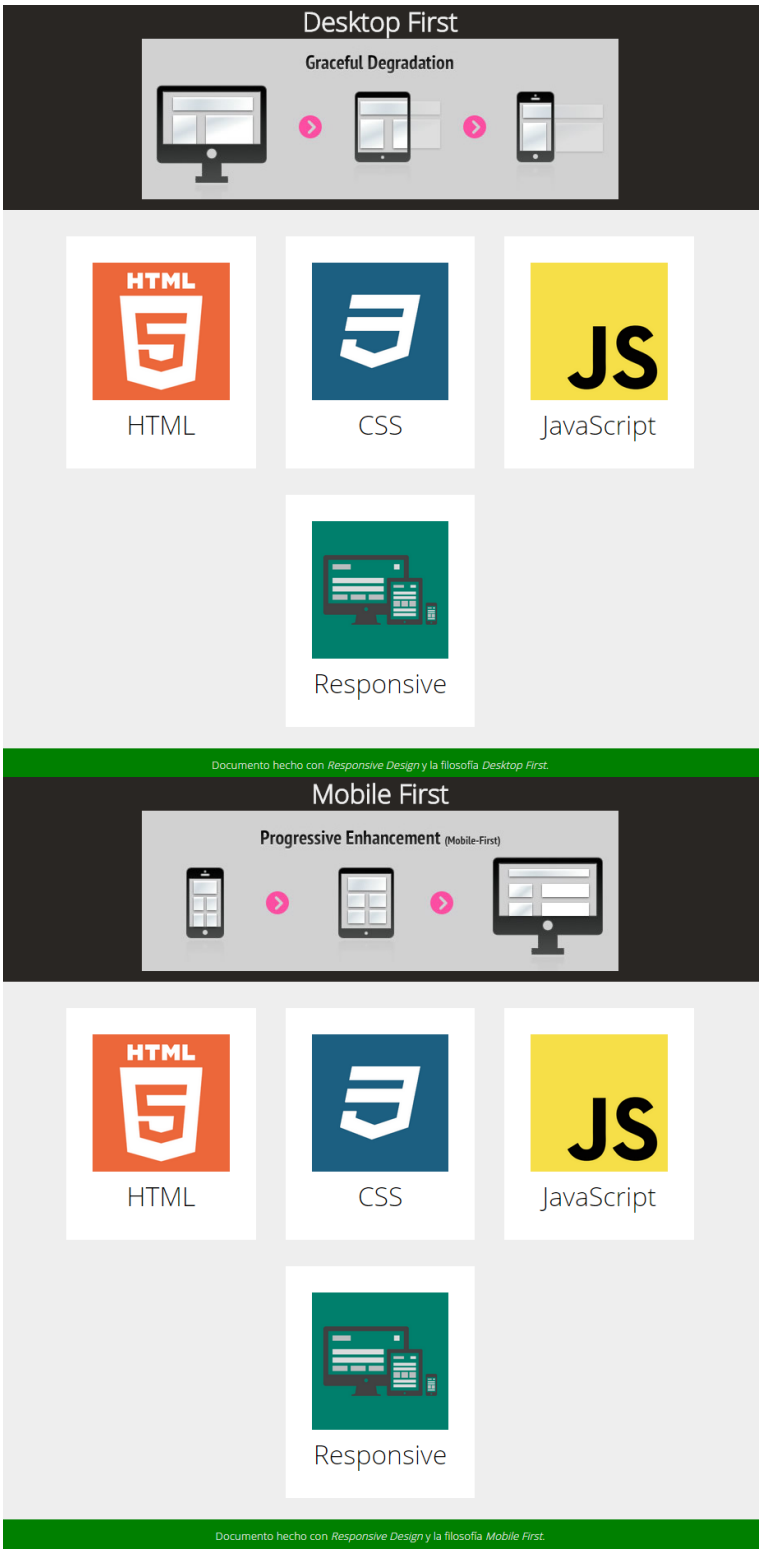

Fig. 2.25 y 2.26 Documentos *desktop-first.html* y *mobile-first.html* prueba *responsive* de color rango de 1024 a 769px (color *green*).

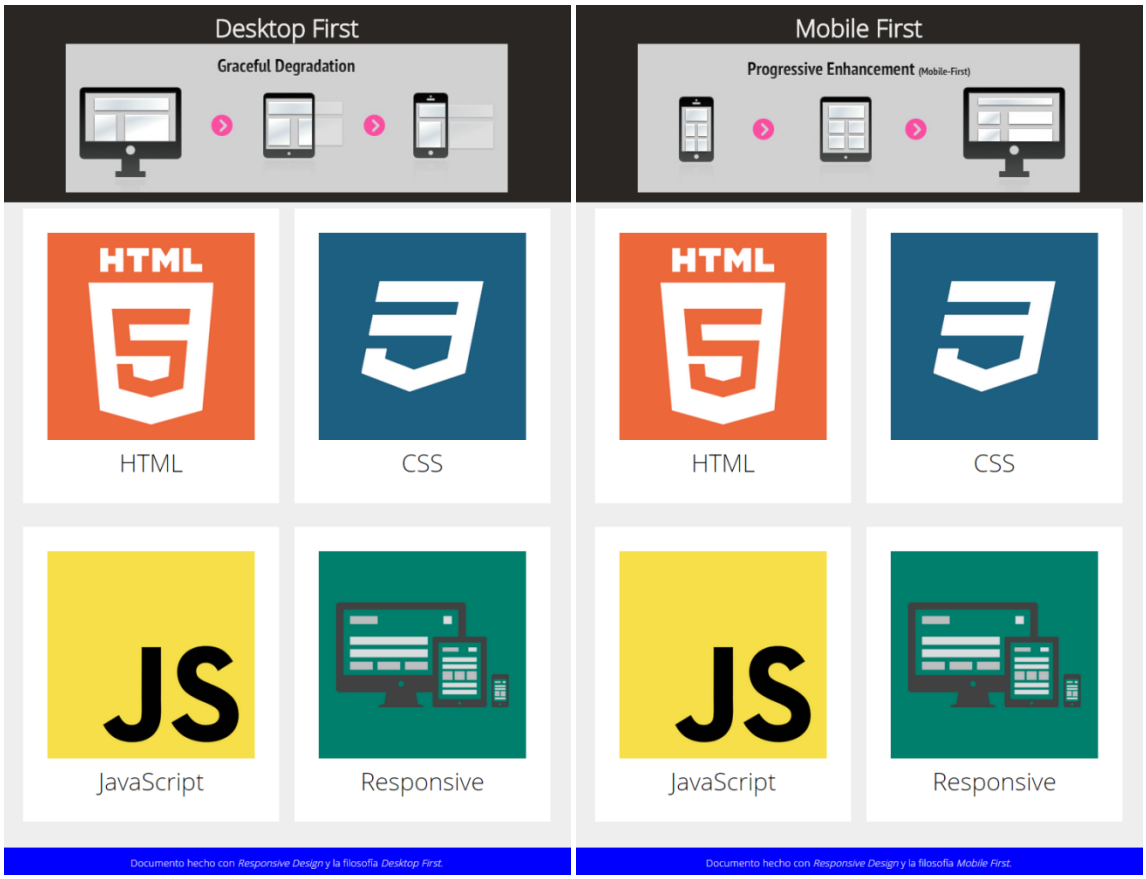

Fig. 2.27 y 2.28 Documentos *desktop-first.html* y *mobile-first.html* prueba *responsive* de color rango de 768 a 601px (color *blue*).

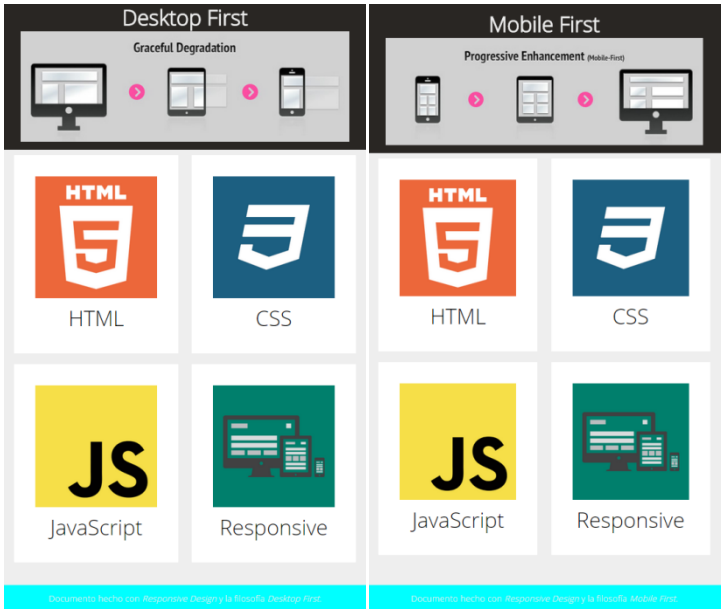

Fig. 2.29 y 2.30 Documentos *desktop-first.html* y *mobile-first.html* prueba *responsive* de color rango de 600 a 481px (color *cyan*).

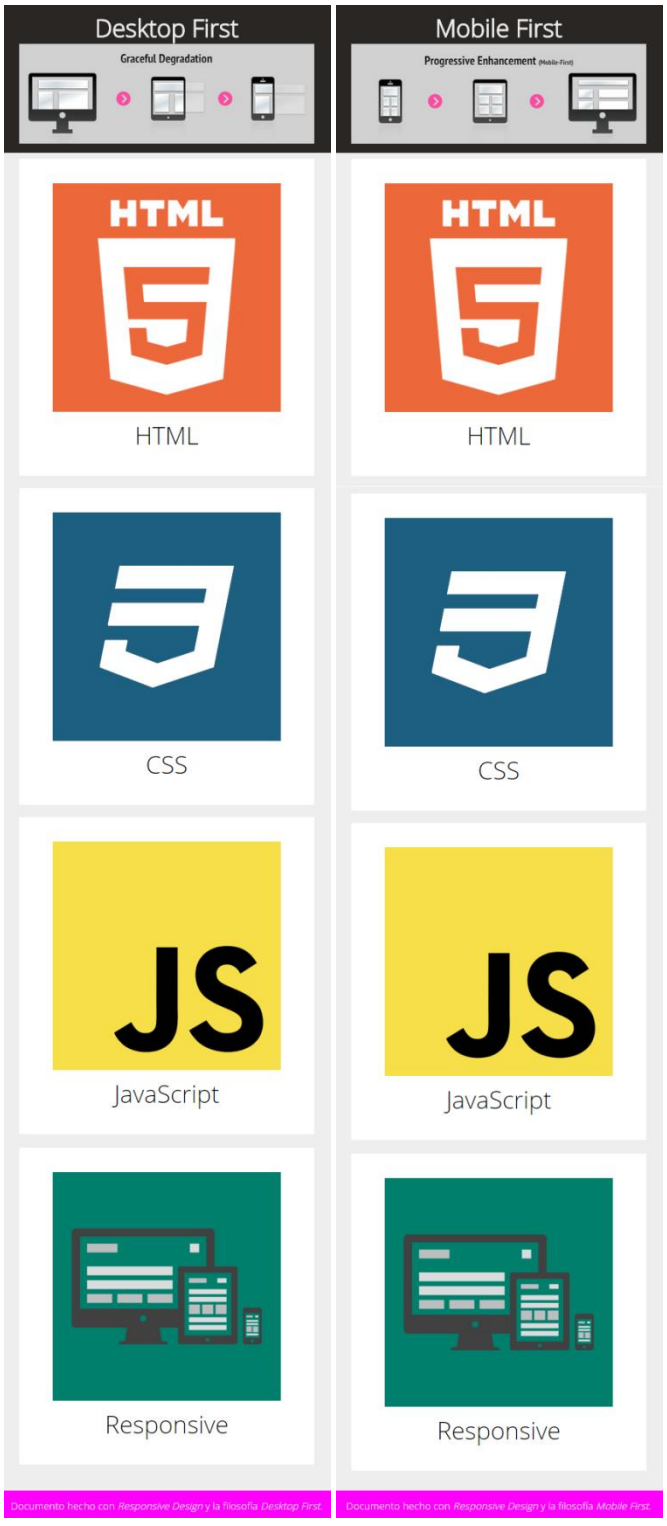

Fig. 2.31 y 2.32 Documentos *desktop-first.html* y *mobile-first.html* prueba *responsive* de color rango de 480 a 321px (color *magenta*).

Fig. 2.33 y 2.34 Documentos *desktop-first.html* y *mobile-first.html* prueba *responsive* de color a menos de 320px (color *yellow*).

Con esta técnica de colores, el diseñador puede identificar en que media queries se encuentra y así no perder tiempo consultando herramientas externas que ralenticen su proceso de trabajo.

Una vez que el sitio se termina es importante mencionar que el estilo de color que se aplico en esta técnica deberá comentarse, para que cuando el proyecto esté en producción el usuario final sólo vea el color original que se le asigne al elemento en nuestro diseño; si más adelante se le tuviera que hacer algún cambio al sitio, nuevamente se des comenta el estilo de color para trabajar la maquetación.

IX. Contenido *responsible*.

En el diseño de un sitio web tradicional, no se empezaba a trabajar con el contenido hasta haberlo construido, y la generación del mismo formaba parte de la validación y pruebas finales del proyecto.

Actualmente, con la llegada de los dispositivos móviles, comprender el contenido se vuelve más importante que nunca. La diferencia entre el escritorio y el teléfono inteligente es el tamaño de pantalla lo que hace más pequeño el espacio disponible para la presentación de elementos de navegación, multimedia, texto, etc. En una filosofía de diseño responsive lo recomendable es la elaboración de la estrategia de contenido desde el principio, y crear el contenido tan pronto como sea posible también. Dar prioridad a lo que hay que presentar al usuario en el ámbito móvil ayudará a evitar las páginas de contenido excesivamente largas o la necesidad de romper un fragmento de contenido en varias páginas, que casi garantizan más tiempo de descarga y el potencial de agotamiento del usuario debido a la impaciencia.

En la web se tiene 2 tipos de contenido principalmente:

1. Texto y
2. Todo lo que NO es texto, es decir, Multimedia (imágenes, videos, mapas, *feeds* de redes sociales, *pdf's*, *swf's*, *svg's* etc.).

En una filosofía de diseño *responsive*, ambos contenidos tiene que trabajarse de manera responsable para la correcta visualización y carga de los mismos.

Para el texto es importante utilizar tipografías seguras que son aquellas que pueden interpretar todos los navegadores en cualquier plataforma o sistema operativo.

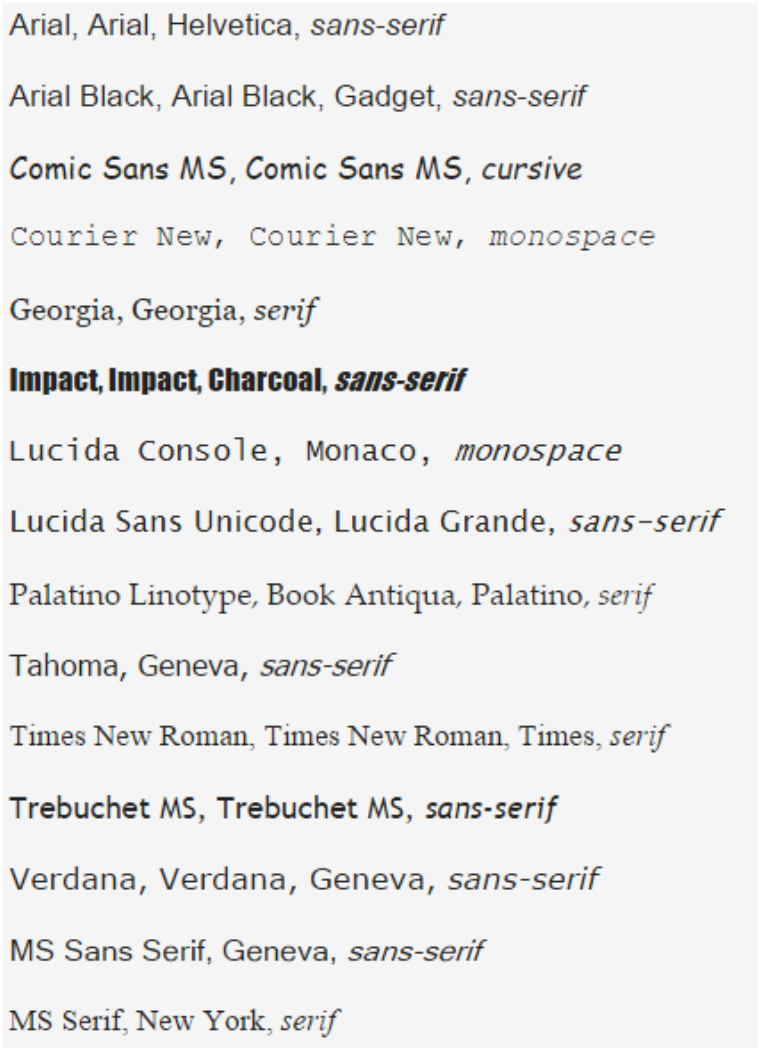

Fig. 2.35 Tipografías seguras.

Adicionalmente se pueden utilizar las tipografías proporcionadas de forma libre por *Google* en su proyecto *Google Fonts*.

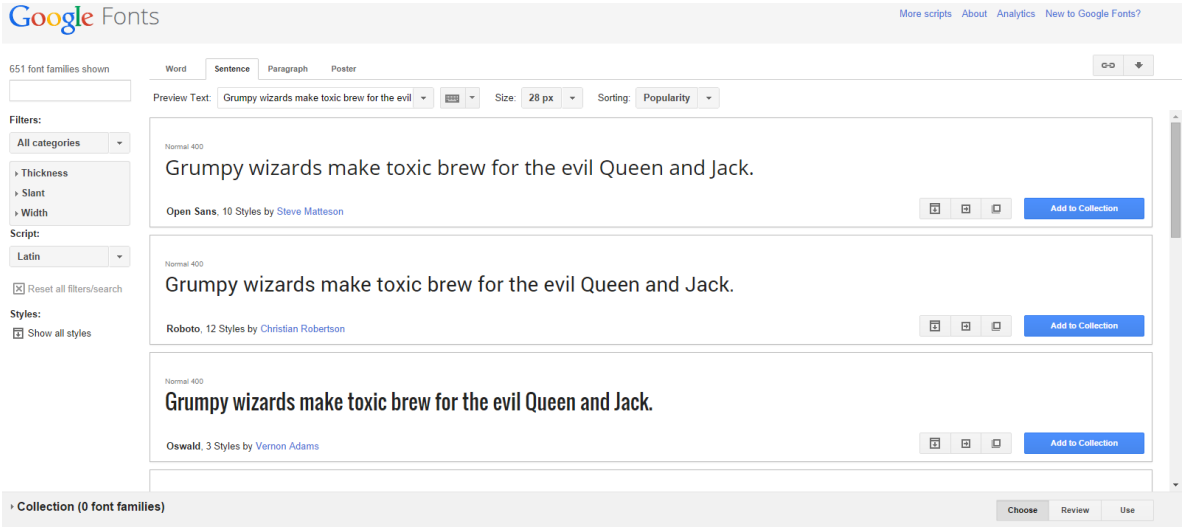

Fig. 2.36 *Google Fonts*.

Respecto al tamaño de los textos en la web es importante usar medidas relativas y escalables como los em, ex o %.

- 1 em = Ancho de la letra "M" de la fuente definida por defecto en el navegador.
- 1 ex = Alto de la letra "x" de la fuente definida por defecto en el navegador.
- 100% = Medida por defecto del navegador.

Adicionalmente existe una unidad de medida llamada rem que es el equivalente al em de la raíz del documento *HTML* es decir la unidad de medida de la etiqueta *HTML*. La diferencia entre usar em y rem es que los primeros siempre serán relativos al valor del elemento padre contenedor, mientras que los segundos, siempre serán relativos al elemento *HTML*.

Si eres de los diseñadores que siguen pensando todo en pixeles puedes consultar el siguiente sitio para la conversión entre pixeles, puntos, porcentajes y *em's*.

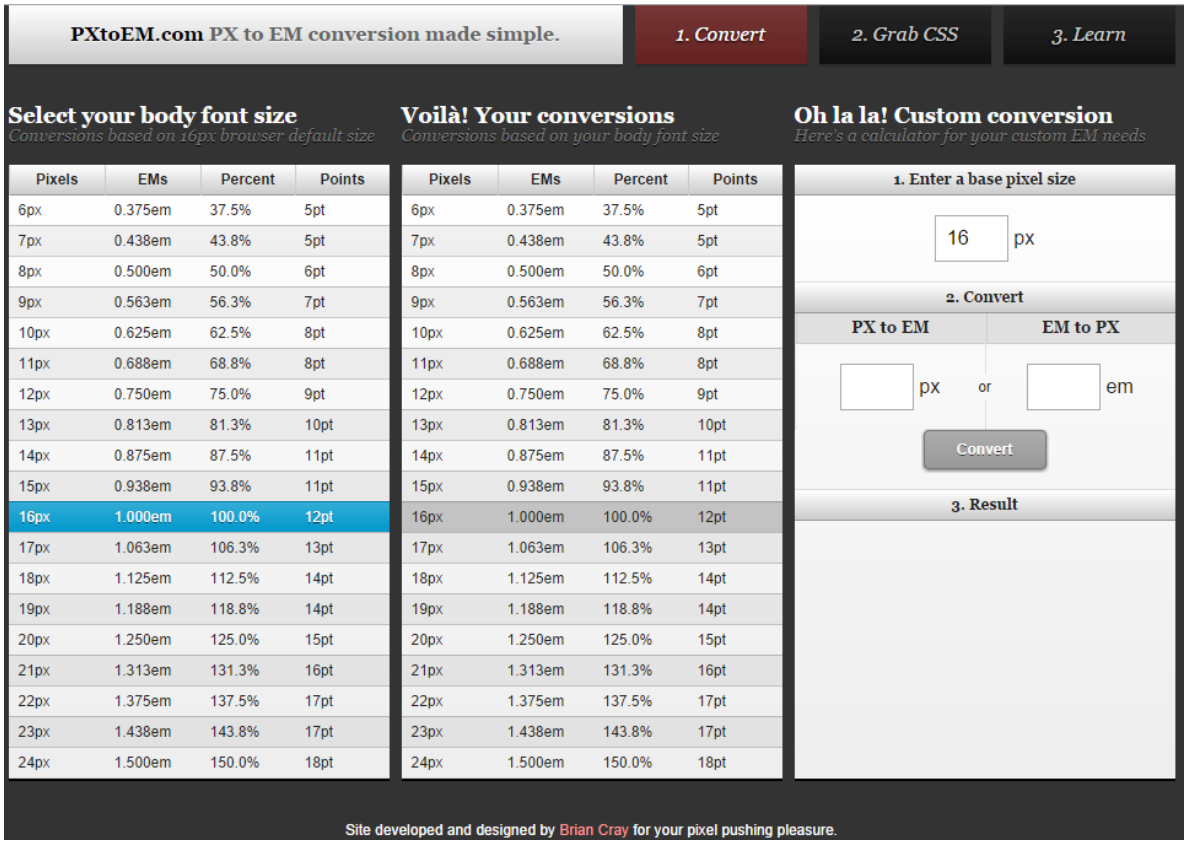

Fig. 2.37 Sitio para la conversión de pixeles a *em's* http://pxtoem.com.

Como se puede observar en la figura anterior la medida base inicial para el *body* del documento *HTML* es:

16px = 1em = 100% = 12pt

Esta equivalencia nos indica el valor que el navegador le da por defecto al texto que encuentra en la etiqueta *body* del documento *HTML*, si trabajamos con unidades relativas y escalables como los *em's* y porcentajes, es una buena práctica siempre especificar un tamaño de fuente para el *body*, ya que al ser el contenedor padre de todas las etiquetas que trabajen con texto en el documento, éstas tomarán como referencia el valor que se le haya definido a este.

Dicho valor para el *body* puede ser especificado en porcentajes o en em's, incluso en pixeles ya que aunque sea una medida absoluta, al ser este el elemento de referencia, los valores de los demás elementos tomarán como medida base el valor del *body* para realizar la conversión a *em's* o porcentajes, como lo muestra la tabla de la figura 2.37.

El tamaño mínimo recomendable para asignarle al *body* de un documento *HTML* es 16px (tamaño que asigna por defecto la mayoría de navegadores), ya que si llegáramos a asignar un valor más pequeño corremos el riesgo de que los usuarios tengan problemas para leer.

Finalmente, respecto a los elementos multimedia uno de los problemas que se tienen con los dispositivos tanto de escritorio como móviles, es que desde el año 2012 comenzaron a salir equipos con pantallas de mayor densidad de pixeles, lo que *Apple* bautizó como "*Retina Display*".

Básicamente, *Retina Display,* es el aumento al doble de pixeles en la pantalla (pixeles físicos). Por ejemplo 16px en *CSS* en un monitor estándar, corresponden a 16px en la pantalla, pero a diferencia, en un monitor *Retina*, 16px corresponderían a 32px; esto para mantener una correcta proporción. Siguiendo el principio anterior, 16px en una imagen, equivalen igualmente al doble en una pantalla *Retina*.

Fig. 2.38 Ejemplo de imágenes en un dispositivo normal (Sin *Retina Display*).

Fig. 2.39 Ejemplo de imágenes en un dispositivo *Retina Display*.

En las figuras 2.38 y 2.39 se ha tomado la captura de pantalla de un ejemplo de imágenes con y sin soporte para Retina *Display*.

Podemos ver que en la figura 2.38 las tres imágenes (‘a’, ‘b’ y ‘c’) se ven iguales, dicha captura de pantalla se tomo en una computadora de escritorio con resolución normal; a diferencia de la figura 2.39 que se tomo en un teléfono *Samsung Galaxy S4* en modo vertical, cuya pantalla es Retina, en dicha captura se puede apreciar como la imagen ‘a’ se ve borrosa a diferencia de la ‘b’ y ‘c’ que están adaptadas para este tipo de pantallas.

El documento de ejemplo *retina-display.html*, así como su hoja de estilos *retina-display.css* se puede acceder a ellos dando clic sobre su nombre. Analicemos el código de las 3 imágenes:

- a, es una imagen de 256px de anchura por altura insertada directamente en el *HTML*, eso significa que en una pantalla retina esta ocupara 512px de ancho por alto, es por ello que en la figura 2.39 se ve borrosa.

Para dar soporte a equipos *Retina*, se puede hacer con *CSS* o *JavaScript*, a continuación se explican ambas técnicas:

- b, es una imagen insertada como fondo *CSS* en una *div* a la que se le ha dado la anchura, la altura y el tamaño de fondo de la imagen (ancho de 256px y alto de 256px).

```
#retina-imagen
{
    background: url(../img/bextlan.png) no-repeat center;
    background-size: 256px 256px;
    height: 256px;
    width: 256px;
}
```

Fig. 2.40 Código *CSS* imagen b.

Este código es el que estaría funcionando en equipos de pantalla normal, para los dispositivos con *Retina Display* adicionalmente se tiene que agregar una media queries que pregunte por la densidad de pixeles de la pantalla del dispositivo y si esta es mayor a 1.5, lo que se hace es reemplazar la imagen de fondo por una imagen al doble de tamaño, continuando con el ejemplo, la imagen *bextlan@2x.png* tiene un tamaño de 512px de anchura por altura, lo que hace que se siga visualizando la imagen de fondo en la etiqueta div que tiene un tamaño de 256px de anchura por altura, pero la imagen que se descarga es la de 512px, con esto se logra que la imagen no se vea borrosa en pantallas *Retina*.

```
/*
Mediaqueries para soporte a pantallas con mayor densidad
de pixeles (Retina Display)
*/
@media only screen and (-ms-min-device-pixel-ratio: 1.5),
    only screen and (min--moz-device-pixel-ratio: 1.5),
    only screen and (-webkit-min-device-pixel-ratio: 1.5),
    only screen and (min-device-pixel-ratio: 1.5){

    #retina-imagen
    {
        background-image: url(../img/bextlan@2x.png);
    }

}
```

Fig. 2.41 Código de la *media queries* para soporte a pantallas *Retina*.

Es importante mencionar que como se muestra en la figura 2.41 a la media queries del *min-device-pixel-ratio* hay que agregarle sus prefijos correspondientes *(-ms-* para *IE*, *-moz-* para *Firefox* y *–webkit-* para los restantes *Chrome*, *Safari*, *Opera*, *Android*, *iOS*, *BB*, etc.) para que todos los navegadores la detecten. Esta solución es muy efectiva ya que en las pantallas normales se estaría descargando la imagen original y en las

Retina la que está al doble de tamaño, lo que hace que pantallas normales no descarguen una imagen de mejor calidad y mayor peso, sin embargo es una técnica muy tediosa, ya que si nuestro sitio tuviera diferentes imágenes y de diferentes tamaños, habría que hacer esto para cada imagen, lo que haría un código *CSS* muy extenso.

Para lidiar con ello la agencia digital *imulus* creó una librería *JavaScript* que hace el trabajo sucio por nosotros: *Retina.js*.

- c, es una imagen de 256px de anchura por altura insertada directamente en el *HTML* para pantallas normales y al mismo tiempo es una imagen de 512px de anchura por altura insertada directamente en el *HTML* para pantallas *Retina*, c fue implementada con *Retina.js*.

¿Cómo funciona *Retina.js*?

Cuando los usuarios cargan una página, *retina.js* comprueba cada imagen de la página para ver si hay una versión de alta resolución de la imagen en el servidor. Si existe una variante de alta resolución, el *script* la intercambia por la original.

Lo único que hay que hacer es subir la imagen de alta resolución con el mismo nombre que la original precedida del texto "*@2x*" para denotar las variantes de alta resolución en el servidor.

Continuando con nuestro ejemplo el código de la imagen c es:

******

En pantallas de alta resolución *Retina.js* va a revisar en el servidor si existe una imagen alternativa en esta ruta:

img/bextlan@2x.png

e intercambiará el código *HTML* por:

******

Para utilizar *Retina.js* sólo hay que subir el script a nuestro servidor y llamarlo en la parte inferior del documento *HTML*, justamente antes del cierre de la etiqueta *body*, y *Retina.js* reemplazará automáticamente las imágenes por nosotros.

```
        <figure>
            <h2>Imagen c</h2>
            <img src="img/bextlan.png" />
            <figcaption>Imagen HTML de 512px de ancho por alto con
            soporte retina display visualizada a 256px con la
            librería JavaScript retina.js.</figcaption>
        </figure>
    </section>
    <footer>
        <small>
            Documento hecho con <i>Responsive Design</i> y la
            filosofía <i>Mobile First</i>.
        </small>
    </footer>
    <script src="js/retina.min.js"></script>
</body>
</html>
```

Fig. 2.42 Implementación de *Retina.js*.

Como se puede observar la solución ideal para el soporte a pantallas de alta resolución es utilizar esta librería.

X. Carga *responsible*.

En el diseño de un sitio web tradicional, no se necesitaba pensar en la carga de contenido *responsible*, al diseñarse para una sola resolución, el contenido siempre era el mismo, actualmente con la llegada de los dispositivos móviles, detenerse a pensar en que contenido se cargará en móviles y cuál en escritorio es fundamental, y es un paso que se debe

planear antes de comenzar a diseñar y programar el sitio. Veamos un documento (*carga-contenido.html*) de ejemplo:

Fig. 2.43 Carga de contenido *responsible* vista escritorio.

Fig. 2.44 Carga de contenido *responsible* vista móvil.

Como podemos observar en las figuras 2.43 y 2.44 tenemos un documento que en la vista de escritorio tiene un título, una imagen, un texto y un video, mientras que en la vista de móvil tenemos el mismo contenido, exceptuando el video, si analizamos el código *CSS* del ejemplo (*carga-contenido.css*) podemos ver que el responsive del documento se hizo con *Mobile First* y se puede apreciar que en los estilos generales se le aplico un *display:none* al video, lo que hace que en móviles no se vea el contenido y en la *media queries min-width:769px*, se le cambia el valor a *display:block*.

A nivel de presentación nuestro documento está bien, pues no visualiza el video en móviles y si en escritorio, pero es importante recordar que la regla *CSS display:none*, lo único que hace es ocultar el elemento de la visualización del navegador, eso significa que no lo mostrará, pero que si lo seguirá descargando, entonces el usuario que visite este documento desde su móvil estaría descargando los 6.99MB que pesa el video sin verlo.

Ocultar el contenido únicamente con *CSS* es una mala práctica de *responsive*, lo que se debe hacer es cargar el contenido de forma *responsible* con la ayuda de *JavaScript*.

Lo que se debe hacer es cargar o remover el contenido dinámicamente con *JavaScript* en el momento que el documento se carga en la ventana del navegador o cuando el usuario redimensione la misma.

Continuando con nuestro ejemplo, nuestro documento se maqueto con el enfoque *Mobile First*, eso significa que de inicio el video no debe de mostrarse en el documento *HTML* y que éste se mostrará hasta después de los 769px de anchura.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=2, user-scalable=yes" />
    <meta charset="UTF-8" />
    <title>Carga de Contenido Responsable</title>
    <meta name="description" content="En este documento explicaremos el concepto de la carga de contenido responsable" />
    <link rel="stylesheet" href="css/carga-contenido-mobile-first.css" />
</head>
<body>
    <header>
        <h1>Carga de Contenido Responsable</h1>
    </header>
    <section id="contenido">
        <h1>HTML5</h1>
        <img src="img/html5.png" />
        <p>
            <i>HTML5</i> define los actuales estándares de diseño y desarrollo web, tanto para equipos de escritorio, como para
            dispositivos móviles.
        </p>
        <div id="video"></div>
    </section>
    <footer>
        <small>
            Documento hecho con <i>Responsive Design</i> y la filosofía <i>Mobile First</i>.
        </small>
    </footer>
    <script src="js/carga-contenido-mobile-first.js"></script>
</body>
</html>
```

Fig. 2.45 Código *HTML* del documento *carga-contenido.html*.

Si vemos el código del documento *HTML*, podemos observar que después del párrafo que contiene el texto hay una etiqueta div vacía con el atributo id 'video'; se encuentra vacía por que el documento está pensado primero en el móvil, entonces el video sólo se cargará hasta resoluciones de más de 769px.

```javascript
function cargarContenido()
{
    var anchoPantalla = window.innerWidth;
    var video = '<video controls preload poster="img/web-hero.jpg"><
        source src="video/web-superhero.webm" type="video/webm" /><
        source src="video/web-superhero.mp4" type="video/mp4" /></
        video>';

    if(anchoPantalla>769)
    {
        document.getElementById("video").innerHTML = video;
    }
    else
    {
        document.getElementById("video").innerHTML = null;
    }
}

window.addEventListener("load",cargarContenido);
window.addEventListener("resize",cargarContenido);
```

Fig. 2.46 Código *JavaScript* para cargar el contenido de forma *responsible*.

Si analizamos el *script* (*carga-contenido.js*) que hace cargar el contenido de forma responsable, podemos ver que se ha creado una función llamada *cargarContenido*, misma que se ejecuta en la carga del documento y en el redimensionamiento de la ventana del navegador (los *addEventListener*).

Dentro de la función se crea una variable para obtener el ancho de la pantalla (*var anchoPantalla*) y otra donde se guarda el código *HTML* que hace visualizar el video (*var video*). Posteriormente se ejecuta una condición que pregunta si la variable que guarda el ancho de la pantalla es mayor a 769px (*anchoPantalla>769*) entonces modifica el contenido *HTML* de la *div* con el *id video* y le agrega el contenido de la variable que guarda el video; en caso contrario, es decir, si el ancho de la pantalla es menor o igual a 769 entonces modifica el contenido *HTML* de la *div* con el *id* video a *null* lo que hace que el código del video se borré y por ende no se cargue en el documento.

Si abrimos el inspector de elementos del navegador podemos ver el código *HTML* en vivo y nos daremos cuenta que en una resolución superior a 769px la *div* con el *id video* carga el código del video mientras que en resoluciones menores la *div* se mantiene vacía (ver las figuras 2.47 y 2.48).

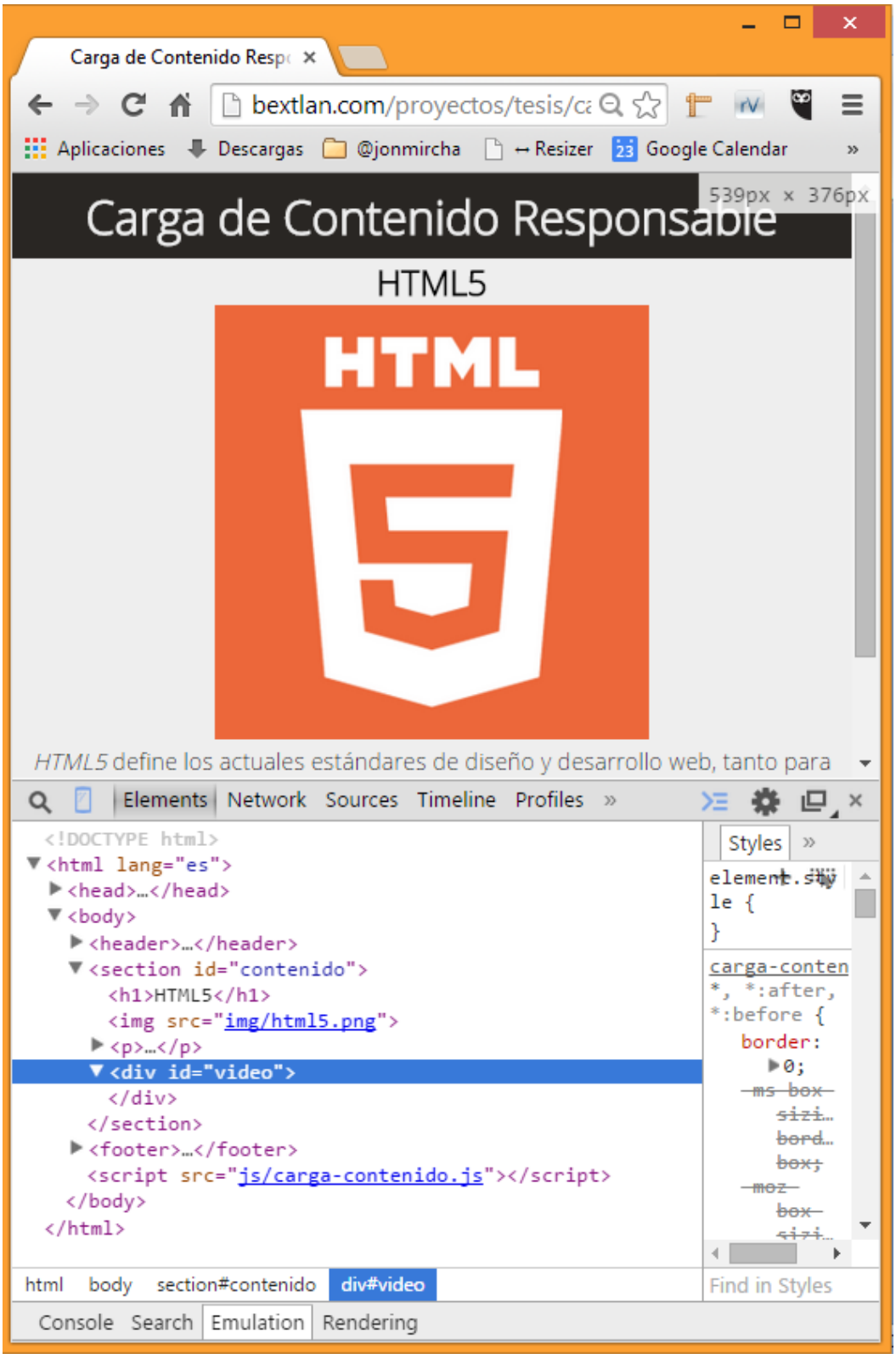

Fig. 2.47 Prueba con el inspector de elementos a 539px, la *div id video* permanece vacía.

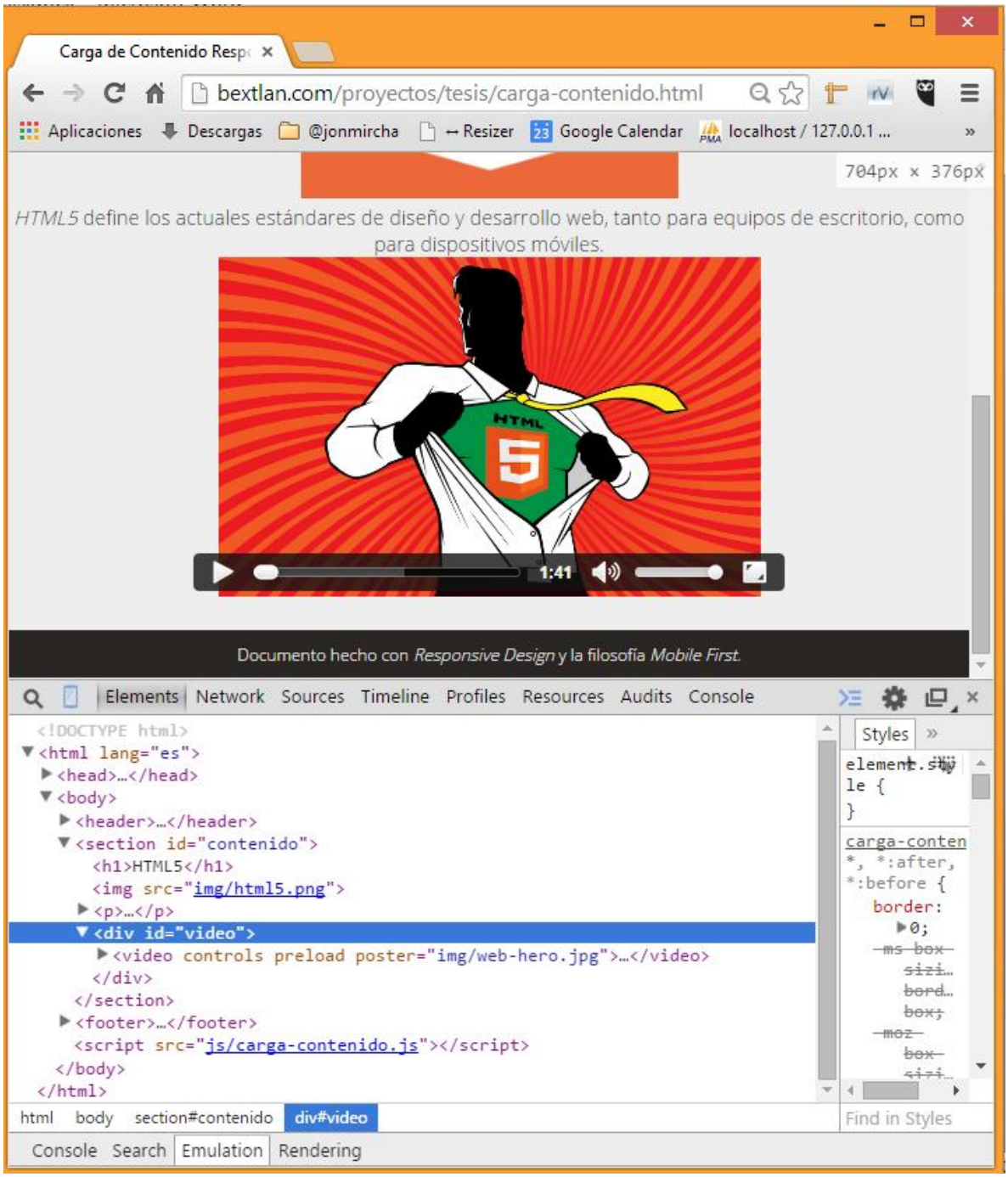

Fig. 2.48 Prueba con el inspector de elementos a 704px, la *div id video* permanece contiene el código *HTML* que inserta el video.

Finalmente para concluir con este punto es importante destacar que dependiendo del enfoque responsive con el que se trabaje, será la forma en cómo cargar el contenido de forma *responsible*.

En nuestro ejemplo hemos trabajado con el enfoque *Mobile First* (que es el recomendable), por tal razón se ha mantenido vacío el elemento *HTML* donde se muestra el video y sólo hasta la resolución que se debe mostrar, es que se agrega dinámicamente; si el usuario lector decidiera trabajar con el enfoque *Desktop First* entonces inicialmente el contenido tendría que cargarse en el documento *HTML* y posteriormente en la resolución donde ya no se deba mostrar, removerlo dinámicamente con *JavaScript* (ver las figuras 2.49 y 2.50).

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=2, user-scalable=yes" />
    <meta charset="UTF-8" />
    <title>Carga de Contenido Responsable</title>
    <meta name="description" content="En este documento explicaremos el concepto de la carga de contenido responsable" />
    <link rel="stylesheet" href="css/carga-contenido.css" />
</head>
<body>
    <header>
        <h1>Carga de Contenido Responsable</h1>
    </header>
    <section id="contenido">
        <h1>HTML5</h1>
        <img src="img/html5.png" />
        <p>
            <i>HTML5</i> define los actuales estándares de diseño y desarrollo web, tanto para equipos de escritorio, como
            para dispositivos móviles.
        </p>
        <div id="video">
            <video controls preload poster="img/web-hero.jpg">
                <source src="video/web-superhero.webm" type="video/webm" />
                <source src="video/web-superhero.mp4" type="video/mp4" />
            </video>
        </div>
    </section>
    <footer>
        <small>
            Documento hecho con <i>Responsive Design</i> y la filosofía <i>Desktop First</i>.
        </small>
    </footer>
    <script src="js/carga-contenido.js"></script>
</body>
</html>
```

Fig. 2.49 Código *HTML* del documento *carga-contenido.html* con enfoque *Desktop First*.

```
function cargarContenido()
{
    var anchoPantalla = window.innerWidth;
    var video = '<video controls preload poster="img/web-hero.jpg"><
        source src="video/web-superhero.webm" type="video/webm" /><
        source src="video/web-superhero.mp4" type="video/mp4" /></
        video>';

    if(anchoPantalla<769)
    {
        document.getElementById("video").innerHTML = null;
    }
    else
    {
        document.getElementById("video").innerHTML = video;
    }
}

window.addEventListener("load",cargarContenido);
window.addEventListener("resize",cargarContenido);
```

Fig. 2.50 Código *JavaScript* para cargar el contenido de forma *responsible* con enfoque *Desktop First*.

Responsible Responsive Design.

Con los 10 consejos mencionados en el apartado anterior, el *Responsable Responsive Design* podría definirse como:

La capacidad que tiene un sitio web para ser cargado y visualizado correctamente en el dispositivo que lo consuma, adaptándose a las características de éste (tamaño de pantalla, resolución de píxeles, etc.). No se trata exclusivamente de reacomodar la información sino de presentar y cargar el contenido de manera óptima para el dispositivo que el usuario este usando.

Capítulo III: CASO PRÁCTICO.

Responsible Responsive Design en el sitio bextlan.com v3.1.

> "Bextlan.com v3.1 | lugar de... bits, vectores y pixeles. Startup que enseña *HTML*5, tecnologías web y difunde la cultura de los medios digitales a través de cursos de pago en línea y videotutoriales gratuitos. De México con amor, para el resto del mundo."
>
> (MirCha, 2014).

En este apartado, aplicaremos de forma práctica los 10 tips de *Responsible Responsive Design* vistos en el capítulo anterior, al sitio bextlan.com versión 3.1.

Nota: Es importante mencionar que para que el usuario lector pueda comprobar por sí mismo la aplicación de los *tips* que se mostraran en las figuras de este capítulo, puede acceder a la hoja de estilos del sitio en línea desde este enlace (*bextlan.css*), al *script* de programación desde este otro (*bextlan.js*) y finalmente para acceder al código *HTML* de cada sección, simplemente dar clic derecho sobre el navegador y activar la opción ver código fuente de la página, dicha opción le abrirá el código *HTML* de la página en una nueva pestaña (ver figuras 3.1 y 3.2).

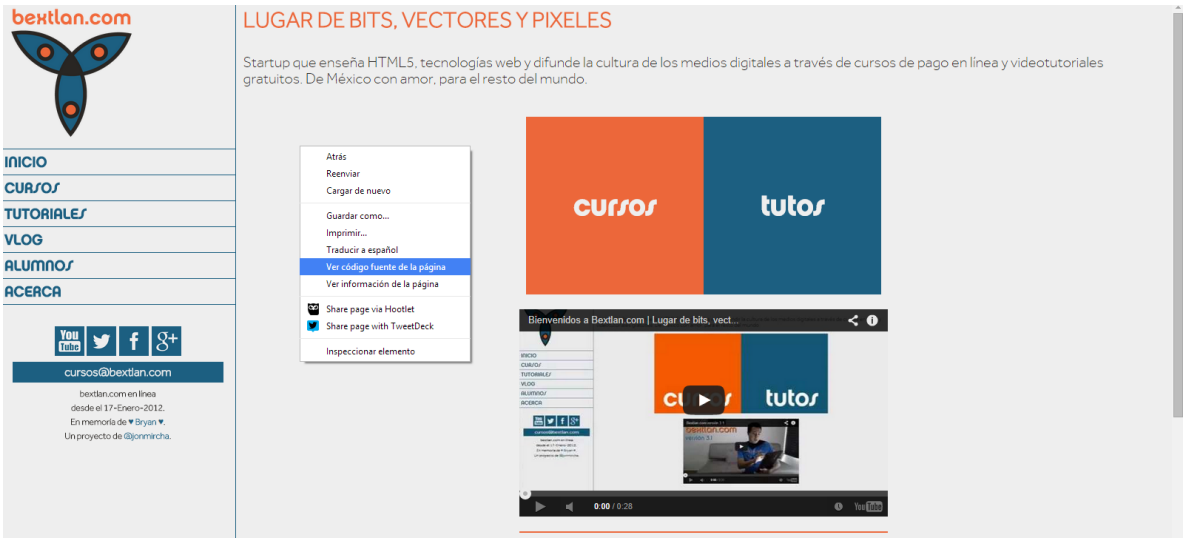

Fig. 3.1 Clic derecho opción ver código fuente de la página.

```
<!DOCTYPE html>
<html lang="es">
<head>
        <base href="http://bextlan.com" />
        <meta name="viewport" content="width=device-width,initial-scale=1,maximum-scale=1,user-scalable=no" />
        <meta charset="UTF-8" />
        <title>Bextlan.com v3.1 | lugar de... bits, vectores y pixeles.</title>
        <meta name="description" content="Startup que enseña HTML5, tecnologías web y difunde la cultura de los medios digitales a través de cursos de pago en línea y videotutoriales gratuitos. De México con amor, para el resto del mundo." />
        <meta property="fb:page_id" content="267632876623807"/>
        <meta property="og:title" content="Bextlan.com v3.1 | lugar de... bits, vectores y pixeles."/>
        <meta property="og:type" content="website"/>
        <meta property="og:url" content="http://bextlan.com/"/>
        <meta property="og:image" content="http://bextlan.com/img/bextlan.png"/>
        <meta property="og:site_name" content="http://bextlan.com/"/>
        <meta property="og:description" content="Startup que enseña HTML5, tecnologías web y difunde la cultura de los medios digitales a través de cursos de pago en línea y videotutoriales gratuitos. De México con amor, para el resto del mundo."/>
        <meta property="fb:admins" content="100000269464371"/>
        <meta name="apple-mobile-web-app-capable" content="yes" />
        <meta name="apple-mobile-web-app-status-bar-style" content="default" />
        <link rel="apple-touch-startup-image" href="http://bextlan.com/img/bextlan.png" />
        <link rel="shortcut icon" type="image/svg+xml" href="http://bextlan.com/svg/bextlan.svg" />
        <link rel="shortcut icon" type="image/x-icon" href="http://bextlan.com/img/favicon.ico" />
        <link rel="shortcut icon" type="image/png" href="http://bextlan.com/img/apple-touch-icon.png" />
        <link rel="apple-touch-icon" href="http://bextlan.com/img/apple-touch-icon.png" />
        <link rel="author" type="text/plain" href="http://bextlan.com/humans.txt"/>
        <link rel="sitemap" type="application/xml" title="Sitemap" href="http://bextlan.com/sitemap.xml"/>
        <link rel="stylesheet" href="css/bextlan.css" /></head>
<body class="fade-in">
        <input type="checkbox" id="logo-btn" />
        <label id="logo" for="logo-btn">
                <h1>bextlan.com</h1>
        </label>
        <section id="panel">
                <nav id="menu">
                        <ul>
                                <li><a href="http://bextlan.com">Inicio</a></li>
                                <li><a href="cursos">Cursos</a></li>
                                <li><a href="tutoriales">Tutoriales</a></li>
                                <li><a href="vlog">Vlog</a></li>
                                <li><a href="alumnos">Alumnos</a></li>
                                <li><a href="acerca">Acerca</a></li>
                        </ul>
                </nav>
```

Fig. 3.2 Código fuente de la página.

Aclarando lo anterior, comencemos con la aplicación de los 10 *tips* sobre *Responsible Responsive Design*:

I. Estructura de documento *responsive*.

En la figura 3.3 podemos apreciar que el sitio tiene el *DOCTYPE HTML* que indica que es un tipo de documento *HTML5*, la etiqueta *HTML* con su atributo *lang* especificando el idioma del documento, en este caso '*es*'. También tiene la etiqueta *meta viewport*, encargada del correcto funcionamiento del *responsive web design* y la etiqueta *meta charset*

especificando el juego de caracteres que tendrá el documento, en este caso '*utf-8*'.

```
<!DOCTYPE html>
<html lang="es">
<head>
		<base href="http://bextlan.com" />
	<meta name="viewport" content="width=device-width,initial-scale=1,maximum-scale=1,user-scalable=no" />
	<meta charset="UTF-8" />
	<title>Bextlan.com v3.1 | lugar de... bits, vectores y pixeles.</title>
	<meta name="description" content="Startup que enseña HTML5, tecnologías web y difunde la cultura de los medios digitales a través de cursos de pago en línea y videotutoriales gratuitos. De México con amor, para el resto del mundo." />
```

Fig. 3.3 Estructura de documento *responsive*.

En la figura 3.4 podemos apreciar la hoja de estilos en cascada antes del cierre de la etiqueta *head*.

```
	<link rel="author" type="text/plain" href="http://bextlan.com/humans.txt"/>
	<link rel="sitemap" type="application/xml" title="Sitemap" href="http://bextlan.com/sitemap.xml"/>
	<link rel="stylesheet" href="css/bextlan.css" /></head>
<body class="fade-in">
	<input type="checkbox" id="logo-btn" />
	<label id="logo" for="logo-btn">
		<h1>bextlan.com</h1>
	</label>
```

Fig. 3.4 Estructura de documento *responsive*.

En la figura 3.5 podemos apreciar los *scripts* enlazados antes del cierre de la etiqueta *body*.

```
	<a id="subir" href="#" title="subir"></a>
		<script src="//ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
	<script src="js/bextlan.js"></script>
	<script>
		var _gaq = _gaq || [];
		_gaq.push(['_setAccount', 'UA-28315572-1']);
		_gaq.push(['_setDomainName', 'bextlan.com']);
		_gaq.push(['_trackPageview']);
		(function(){
			var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;
			ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.google-analytics.com/ga.js';
			var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);
		})();
	</script></body>
</html>
```

Fig. 3.5 Estructura de documento *responsive*.

II. Configuración de la etiqueta *meta viewport*.

En la figura 3.6 podemos apreciar la etiqueta *viewport* configurada para la correcta visualización del contenido en cualquier dispositivo.

```
	<meta name="viewport" content="width=device-width,initial-scale=1,maximum-scale=1,user-scalable=no" />
	<meta charset="UTF-8" />
	<title>Bextlan.com v3.1 | lugar de... bits, vectores y pixeles.</title>
```

Fig. 3.6 Configuración del *viewport*.

III. Maquetación fluida.

En la figura 3.7 podemos observar elementos contenedores del sitio maquetados con unidades relativas (*em's* y porcentajes), permitiendo que el diseño fluya de forma natural con la resolución del dispositivo.

```
83  body
84  {
85          min-height: 100%;
86          overflow-x: hidden;
87          overflow-y: scroll;
88          position: absolute;
89          width: 100%;
90  }
112 #contenido
113 {
114         left: 5em;
115         padding: .5em;
116         position: relative;
117         text-align: center;
118         width: 70%;
119 }
155 #jonmircha-empresas a
156 {
157         display: inline-block;
158         padding: .5em;
159         vertical-align: middle;
160         width: 40%;
161 }
```

Fig. 3.7 Maquetación fluida.

IV. Objetos flexibles.

En el sitio bextlan hay 2 tipos de objetos: las imágenes y los iframes donde se embeben los videos de *Youtube*. En la figura 3.8 se puede observar que a ambos elementos se les ha aplicado la regla *CSS max-width:100%;* haciéndolos flexibles con respecto de su contenedor padre.

```
95
96  h5 { margin-bottom: 1em; }
97
98  hr { border: solid thin #EC673A; margin: 1em auto; }
99
100 iframe, img { max-width: 100%; }
101
102 li { list-style: none;  }
103
104 p { margin-bottom: 1em; }
105
```

Fig. 3.8 Objetos flexibles.

V. *Media Queries*.

VI. Puntos de interrupción (*breakpoints*).

VII. Enfoques responsive: *Mobile First vs Desktop First*.

VIII. Pruebas r*esponsive*.

En la figura 3.9 se puede apreciar el uso de las media queries y los puntos de interrupción propuestos por Marcotte en orden ascendente de la menor a la mayor y validando el ancho de la pantalla con *min-width* lo que nos indica que el enfoque con el que se realizo el *responsive* es *Mobile First*. También se puede ver que la primer línea de código *CSS* dentro de cada media quieres es un cambio de fondo que esta comentado a los enlaces que se encuentran dentro de un elemento con la clase contacto (*.contacto a*), lo que indica que se uso la técnica de

cambio de color mencionada en esta investigación para hacer las pruebas *responsive* en cada punto de interrupción.

```
@media screen and (min-width: 321px){
        .contacto a     { /* background: magenta; */ width: 80%; }

        #contenido { width: 75%; }

@media screen and (min-width: 481px){
        .contacto a     { /* background: cyan; */ font-size: 1.5rem; }

        #categoria ul>li { display: inline-block; width: 48%; }

        #contenido { width: 84%; }
@media screen and (min-width: 601px){
        .contacto a     { /* background: blue; */ width: 60%; }

        #inicio-lista h3
        {
                font-size: 2em;
                height: 6em;
                line-height: 6em;
                width: 6em;
        }
@media screen and (min-width: 769px){
        .contacto a     { /* background: green; */ width: 50%; }

        #categoria ul>li { width: 30%; }

        #contenido { left: 7em; }
@media screen and (min-width: 1025px){
        .contacto a     { /* background: red; */ padding: .5em; }

        body { font-size: 1.25rem; }

        #categoria ul>li { width: 24%; }

        #contenido { left: 9em; }
@media screen and (min-width: 1201px){

        .contacto a     { /* background: #1C5F81; */ width: 30%; }

        #categoria ul>li { width: 19%; }

        #contenido { left: 12em; }
```

Fig. 3.9 *Media queries*, puntos de interrupción, enfoque *Mobile First* y pruebas *responsive*.

IX. Contenido *responsible*.

En la figura 3.10 se puede apreciar la invocación de las tipografías utilizadas en el sitio, así como la asignación del tamaño de fuente en la etiqueta *body* del documento.

```
@font-face{
    font-family: "chaletcito";
    font-style: normal;
    font-weight: normal;
    src: url("../fonts/Chalet.eot");
    src: url("../fonts/Chalet.eot?#iefix") format("embedded-opentype"),
         url("../fonts/Chalet.ttf") format("truetype"),
         url("../fonts/Chalet.svg#Chalet") format("svg"),
         url("../fonts/Chalet.woff") format("woff");
}

@font-face{
        font-family: "effra";
        font-style: normal;
        font-weight: normal;
        src: url("../fonts/EffraLight.eot");
        src: url("../fonts/EffraLight.eot#iefix") format("embedded-opentype"),
                 url("../fonts/EffraLight.ttf") format("truetype"),
                 url("../fonts/EffraLight.svg") format("svg"),
                 url("../fonts/EffraLight.woff") format("woff");
}

@font-face{
        font-family: "flaticon";
        font-style: normal;
        font-weight: normal;
        src: url("../fonts/Flaticon.eot");
        src: url("../fonts/Flaticon.eot#iefix") format("embedded-opentype"),
                 url("../fonts/Flaticon.ttf") format("truetype"),
                 url("../fonts/Flaticon.svg") format("svg"),
                 url("../fonts/Flaticon.woff") format("woff");
}
body, html
{
         background: #EEE;
         color: #292623;
         font-family: "effra", sans-serif;
         font-size: 1rem;
         font-size: 16px;
         text-align: center;
         text-rendering: optimizeLegibility;
}
```

Fig. 3.10 Declaración de Tipografías y asignación de tamaño de fuente en el *body*.

En la figura 3.11 se puede apreciar que a los elementos textuales del sitio se le está aplicando el tamaño de fuente en unidades relativas (*em's*).

```
127 #inicio-lista h3
128 {
129         background: #EC673A;
130         color: #EEE;
131         font-family: "chaletcito", sans-serif;
132         font-size: 2em;
133         height: 4em;
134         line-height: 4em;
135         margin: auto;
136         text-rendering: optimizeLegibility;
137         width: 4em;
138 }
490         #jonmircha>p { font-size: 1.75rem; padding-left: .5em; width: 60%; }
491
492         #jonmircha-empresas a { width: 22%; }
493
494         #logo { height: 2em; width: 2em; }
495
496         #logo:before { top: 1em; width: .33em; }
497
498         #menu a { padding: 1em; }
499
500         #regresar { text-align: left; }
501
502         .boton-alumno b { font-size: 1.5em; margin: 0; width: 50%; }
```

Fig. 3.11 Asignación de tamaño de fuente en *em's*.

Para terminar con este apartado es importante mencionar que para dar soporte a pantallas *Retina* no se ha utilizado ni la librería *Retina.js* ni la *media queries min-device-pixel-ratio*, al ser bextlan un sitio con pocas imágenes y la mayoría de ellas siendo trazos vectoriales, exceptuando las fotos de la sección de alumnos, lo que se hizo fue optimizar las fotografías e imágenes a una calidad web de 72dpi con un tamaño real de 300px de anchura por 300px de altura y con *CSS* se le aplicó a todas las imágenes un tamaño de 150px x 150px, es decir, la mitad. Con esto se ha logrado dar soporte a pantallas *Retina* evitando tener dos archivos de cada imagen, y como se optimizaron con una calidad para web, el peso de las imágenes oscila entre los 5KB y 30KB para imágenes

resultantes de gráficos vectoriales (como las de las secciones de tutoriales y cursos), así como entre 30KB y 60KB para las fotografías (como las de las secciones de alumnos).

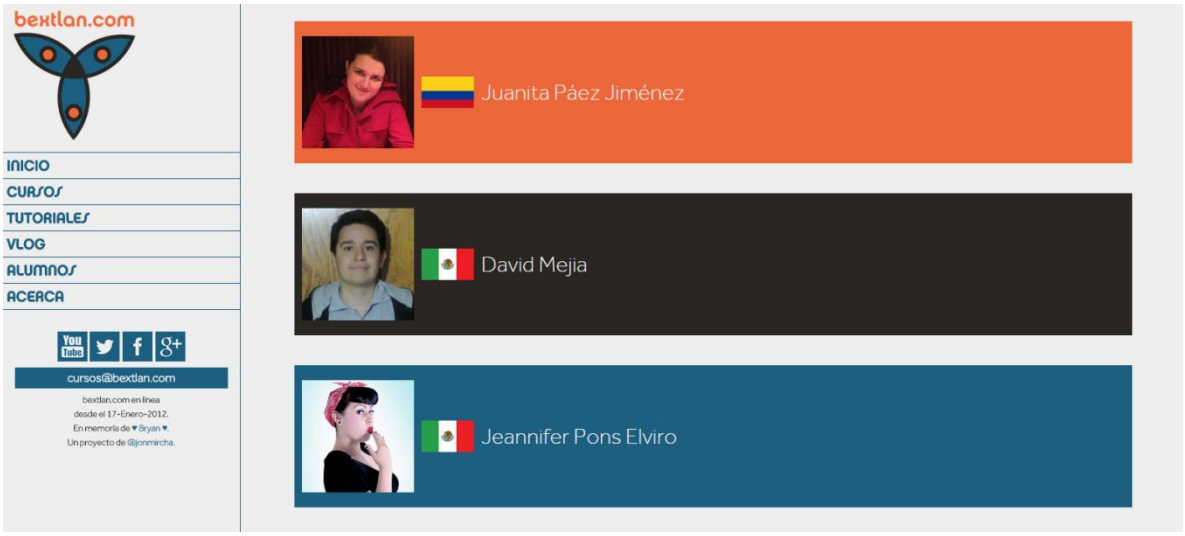

Fig. 3.12 Imágenes de la sección alumnos de bextlan vista desde un equipo con pantalla normal.

Fig. 3.13 Imágenes de la sección tutoriales de bextlan vista desde un equipo con pantalla normal.

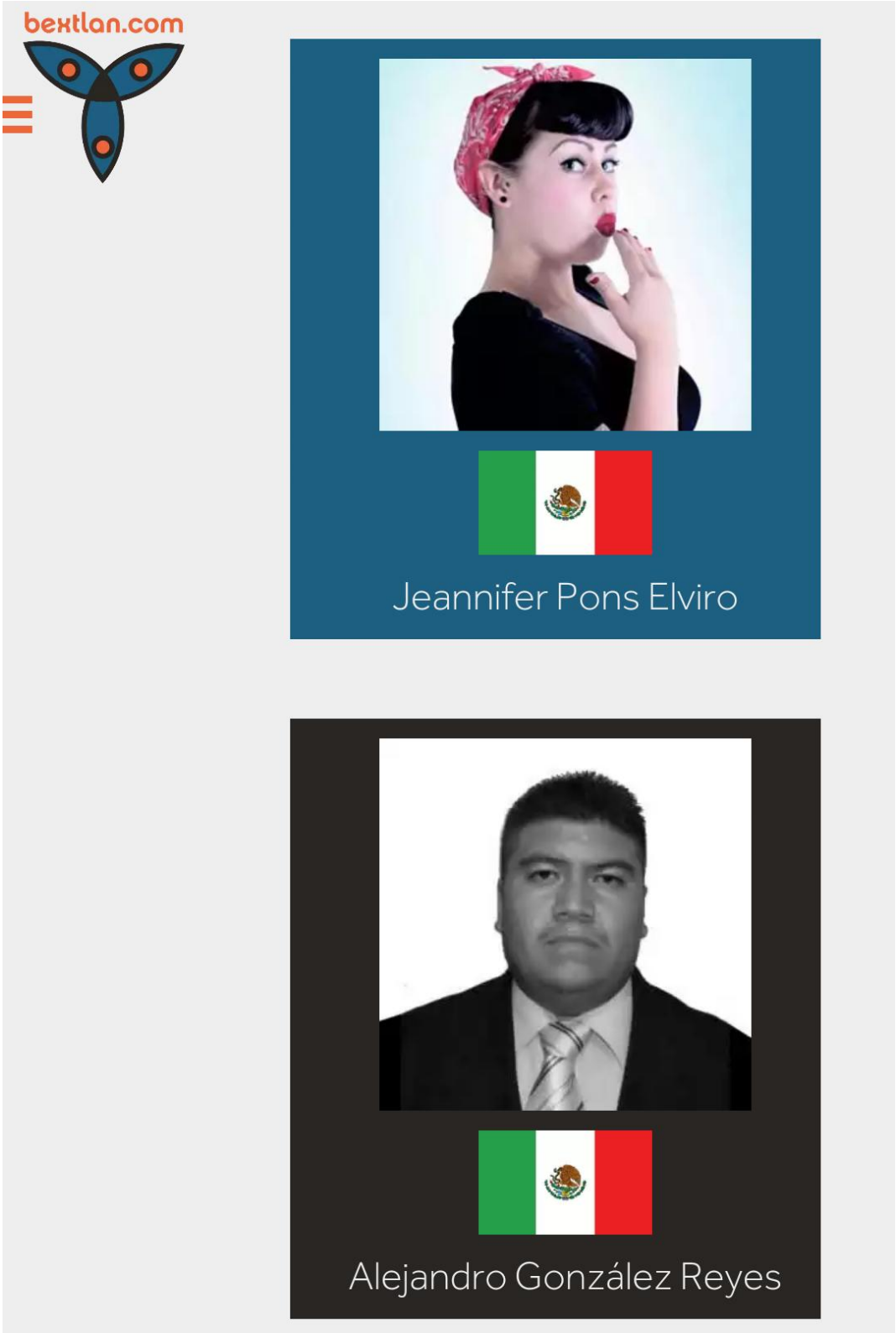

Fig. 3.14 Imágenes de la sección alumnos de bextlan vista desde un equipo con pantalla *Retina*.

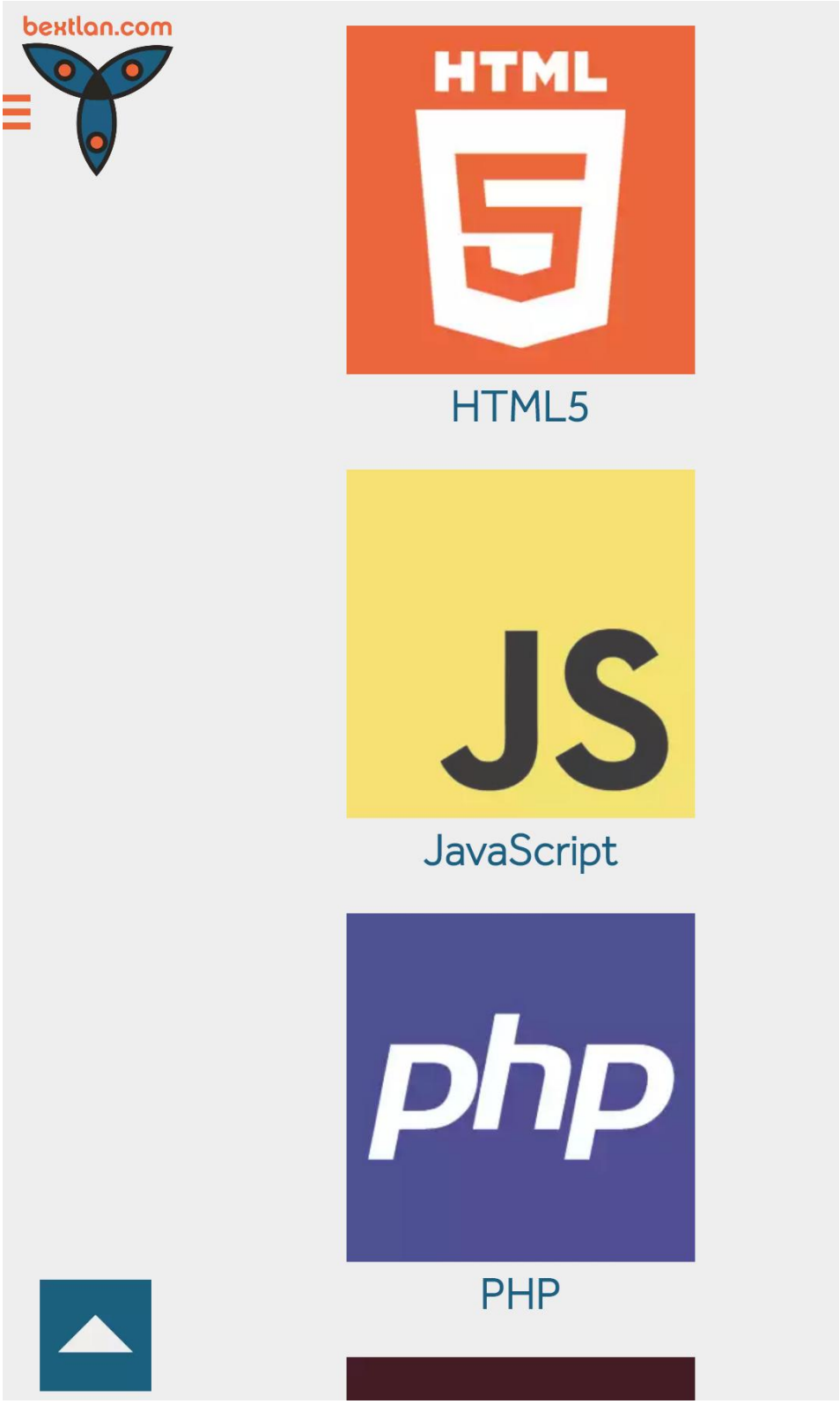

Fig. 3.15 Imágenes de la sección tutoriales de bextlan vista desde un equipo con pantalla *Retina*.

```
105
106 #Bryan { width: 100%; }
107
108 #categoria figure>img, .imagen { height: 150px; width: 150px; }
109
110 #categoria ul>li { padding-bottom: 1em; text-align: center; }
111
```

Fig. 3.16 Código *CSS* donde se aplica el tamaño de imagen a la mitad del real (150px x 150px).

X. Carga *responsible*.

Para el sitio bextlan no se utilizo el *tip* de carga *responsible*, pues todo el contenido que se ve en un equipo de escritorio es el mismo que se visualiza en un móvil, lo único que va cambiando es la distribución de la maquetación del contenido.

Conclusiones.

Teniendo en cuenta lo que se ha descrito un sitio web o aplicación diseñada responsablemente probablemente requerirá más tiempo y recursos que un proceso de desarrollo tradicional, al menos inicialmente. La inversión adicional producirá mejores resultados y ahorros a largo plazo y los beneficios serán mayores que cualquier impacto negativo.

Con la penetración profunda y creciente de los dispositivos móviles la web se está consumiendo en múltiples plataformas. Los diseñadores web tienen que responder a este nuevo entorno. Es mucho más eficiente en el largo plazo construir una base de código para servir a múltiples dispositivos que construir sitios independientes separados cada uno con una base diferente.

El entorno móvil en constante movimiento requerirá la revisión de las técnicas en curso y volver a la prueba contra los nuevos dispositivos y cambios de las plataformas.

Si bien a primera vista puede parecer que el *Responsive Web Design* genera sitios y aplicaciones más grandes y complejas, pero en realidad el responsive con un enfoque responsable puede ayudar a normalizar y regularizar lo que se está convirtiendo para muchas empresas en un enfoque caótico el soporte de sus sitios y aplicaciones web a los dispositivos móviles.

Un enfoque explícito en la experiencia móvil debe permitir una mejor asignación de recursos y prioridades para satisfacer mejor las necesidades de los usuarios, tal vez incluso restar importancia a la experiencia de escritorio que ha dominado el desarrollo durante la mayor parte de la era de *Internet*.

Para terminar con las conclusiones de esta investigación algunos puntos para considerar sobre el Responsible Responsive Design:

- Sólo se tiene que mantener un único sitio web.
- Aborda una amplia variedad de dispositivos: teléfonos, tabletas, computadoras de escritorio, etc.
- No asuma que los usuarios móviles quieran menos contenido y tampoco que quieren el mismo que en el escritorio, cada sitio requerirá su análisis de contenido.
- Comience el proceso de diseño con el móvil.
- Haga la vista móvil predeterminada.
- No oculte contenido, cárguelo o descárguelo dependiendo el caso.
- Pensar el texto en función de su contenedor.
- Usar fuentes seguras para los dispositivos móviles.
- Adaptar las imágenes para los dispositivos y sus resoluciones.
- Finalmente hacer pruebas en varios dispositivos.

Fuentes de consulta.

- *Brendan Eich on JavaScript Taking Both the High and Low Roads.* O'Reilly. 2014. *Youtube*. 12-03-14. Web. 01-08-14. <URL>.
- Wroblewski Luke. *Mobile First*. New York: A Book Apart, 2011. Impreso.
- Siwicki, Bill. "It's official: Mobile devices surpass PCs in online retail". *Internet Retailer.* 01-10-13. Web. 01-08-14. <URL>.
- Stat Counter. *Plataform Comparison World Wide from 2010 to 2014.* 2014. Gráfica. *Global Stats.* Web. 01-08-14. <URL>.
- Nielsen, Jacob. "Mobile Usability Update". *Nielsen Norman Group*. 26-09-11. Web. 01-08-14. <URL>.
- MacNaught, Stacey. "What's Happening to Your Mobile Traffic?". Texto. *Tecmark.* Tecmark, 26-09-11. Web. 05-08.14. <URL>.
- Stat Counter. *Mobile vs Desktop Comparison World Wide 2011.* 2011. Gráfica. *Global Stats.* Web. 05-08-14. <URL>.
- Parkes, Sarah. "ITU sees 5 billion mobile subscriptions globally in 2010". *International Telecommunication Union.* 15-02-10. Web. 05-08-14. <URL>.
- Vega John Freddy, Van Der Henst Christian. *Guía HTML5. El presente de la web.* Latinoamérica: CC BY-NC-SA 3.0, 2011. *Mejorando.la*. Web. 01-08-14. <URL>.
- World Wide Web Consortium. "HTL5 Logo". *World Wide Web Consortium.*15-07-11. Web. 05-08-14. <URL>.
- World Wide Web Consortium. "*HTML*5". *World Wide Web Consortium*. 31-07-2014. Web. 10-08-14. <URL>.

- World Wide Web Consortium. "*HTML* 5.1". *World Wide Web Consortium*. 17-06-14. Web. 10-08-14. <URL>.
- Vega John Freddy. "2013, el año de Javascript". Texto. *Cristalab*. Cristalab, 14-01-13. Web. 18-08-14. <URL>.
- Eguíluz Pérez Javier. *Introducción a JavaScript.* España: CC BY-NC-SA 3.0, 2009. *Librosweb.es*. Web. 01-08-14. <URL>.
- Eguíluz Pérez Javier. *Introducción a AJAX.* España: CC BY-NC-SA 3.0, 2009. *Librosweb.es*. Web. 01-08-14. <URL>.
- ECMA International. "Introducción a JSON". *ECMA International*. Web. 18-08-14. <URL>.
- Eich, Brendan. "The State of JavaScript". *Brendan Eich*. 2012. Web. 20-08-14. <URL>.
- Carey, Lucy. "Top programming languages list of 2013 throws up surprise winner". Texto. *Jaxenter*. S&S Media Group, 07-01-14. Web. 20-08-14. <URL>.
- Tiobe Company. "TIOBE Index for September 2014". *Tiobe Company. Web. 01-09-14.* <URL>.
- Allsopp, John. "A Dao of Web Design". Texto. *A List Apart*. A List Apart, 07-04-00. Web. 02-09-14. <URL>.
- Marcotte, Ethan. "Responsive Web Design". Texto. *A List Apart*. A List Apart, 25-05-10. Web. 02-09-14. <URL>.
- Marcotte Ethan. *Responsive Web Design.* New York: A Book Apart, 2011. Impreso.
- Keith Jeremy. *HTML5 for Web Designers.* New York: A Book Apart, 2010. Impreso.

- World Wide Web Consortium. "Media queries". *World Wide Web Consortium*. 04-04-01. Web. 12-08-14. <URL>.
- World Wide Web Consortium. "Mobile Web Best Practices 1.0". *World Wide Web Consortium*. 17-10-05. Web. 12-08-14. <URL>.
- Ewer, Tom. "5 Reasons Why Responsive Design Is Not Worth It". Texto. *ManageWP Blog.* ManageWP Blog, 31-05-12. Web. 20-08-14. <URL>.
- Bushell, David. "Responsive Mistakes: Less is More". Texto. *David Bushell*. David Bushell, 09-01-13. Web. 08-09-14. <URL>.
- Chan, Josh. "Responsive Web Design is Not the Future". Texto. *Six Revisions*. Six Revisions, 01-04-13. Web. 08-09-14. <URL>.
- Whalen, Brandon. "5 Reasons Why Responsive Web Design Sucks for Ecommerce Businesses". Texto. *Room 214.* Room 214, 06-09-13. Web. 08-09-14. <URL>.
- Young, James. "The 8 biggest responsive web design problems (and how to avoid them)". Texto. *Creative Bloq*. Creative Bloq, 17-12-13. Web. 08-09-14- <URL>.
- Avery, Justin. "Responsive Web Design". Responsive Design.is. 01-03-13. Web. 08-09-14. <URL>.
- Jehl, Scott. "Responsable Responsive Design". *A Book Apart*. 2014. Web. 02-09-14. <URL>.
- Eguíluz Pérez, Javier. *Introducción a CSS.* España: CC BY-NC-SA 3.0, 2009. *Librosweb.es*. Web. 01-08-14. <URL>.
- Eguíluz Pérez, Javier. *CSS Avanzado.* España: CC BY-NC-SA 3.0, 2009. *Librosweb.es*. Web. 01-08-14. <URL>.

- Eguíluz Pérez, Javier. "Posicionamiento Flotante"*. Librosweb.es*. 2009. Web. 01-08-14. <URL>.
- Eguíluz Pérez, Javier. "Propiedad Display"*. Librosweb.es*. 2009. Web. 01-08-14. <URL>.
- *Curso Práctico de HTML5 - Cap 03 Responsive Design con HTML5 y CSS3.* Bextlan. 2012. *Youtube*. 29-09-12. Web. 08-09-14. <URL>.
- Brunildo. "Display:inline-table"*. Brunildo*. Web. 01-08-14. <URL>.
- *CSS* Tricks. "A Complete Guide to Flexbox". *CSS Tricks*. 02-09-14. Web. 08-09-14. <URL>.
- World Wide Web Consortium. "Cascading Style Sheets Level 2 Revision 1 (*CSS* 2.1) Specification". *World Wide Web Consortium*. 07-06-11. Web. 12-08-14. <URL>.
- World Wide Web Consortium. "Media queries". *World Wide Web Consortium*. 19-06-12. Web. 20-08-14. <URL>.
- World Wide Web Consortium. "Graceful degradation versus progressive enhancement". *World Wide Web Consortium*. 11-08-11. Web. 20-08-14. <URL>.
- Acces Sites. "Graceful degradation and Progressive enhancement". *Acess Sites. 02-02-07. Web 20-08-14.* <URL>.
- *Taller: Responsive Web Design con HTML5*. Codejobs. 2012. *Youtube.* 08-10-12. Web. 08-09-14. <URL>.
- Google. "Google Fonts". *Google.* 09-12-11. Web. 10-09-14. <URL>.
- Cray, Brian. "PX to EM". *Brain Cray.* Web. 10-09-14. <URL>.
- Imulus. "Retina.js". *Imulus.* 2014. Web. 10-09-14. <URL>.