Manual de Responsive Web Design

Daniel Martínez
Miguel Angel Alvarez

desarrolloweb.com

desarrolloweb.com/manuales/responsive-web-design.html

Introducción: Manual de Responsive Web Design

Este es un manual de Responsive Web Design, la técnica que permite hacer webs adaptables a las condiciones del ordenador o dispositivo que las está accediendo. También las conocemos como técnicas adaptativas, focalizadas principalmente a las dimensiones de la pantalla.

Responsive Web Design no es una única técnica, sino son un conjunto de ellas que nos sirven para hacer diseños adaptables al medio donde se van a consumir. Por tanto, este manual no tiene un principio y un fin que pudiera estar perfectamente definido, sino que se podría ampliar su contenido en función de las necesidades de los sitios web en la actualidad, experiencia de los escritores o las novedades en los estándares como HTML y CSS para facilitar la adaptabilidad del diseño web.

En Internet no hemos encontrado una información completa y ordenada de principio a fin, en español, sobre Responsive Web Design, por lo que nos hemos decidido a escribir este texto que esperamos sirva de referencia para quien comienza a interesarse por la adaptabilidad del diseño o incluso para quien ya aplica el "Responsive".

Encuentras este manual online en:
http://desarrolloweb.com/manuales/responsive-web-design.html

Autores del manual

Las siguientes personas han participado como autores escribiendo artículos de este manual.

Miguel Angel Alvarez

Miguel es fundador de DesarrolloWeb.com y la plataforma de formación online EscuelaIT. Comenzó en el mundo del desarrollo web en el año 1997, transformando su hobby en su trabajo.

Daniel Martínez

Diseñador gráfico convertido a web

Conceptos fundamentales del diseño responsive

Comenzamos por una serie de artículos que explican los motivos por los que hoy el diseño web debe de ser adaptable, analizando los conceptos más fundamentales. El objetivo es entender que debemos diseñar para todos los tipos de pantalla, pero también para todas las velocidades de conexión, todos los navegadores, etc.

Qué es Responsive Web Design

Comenzamos con una descripción general del término Responsive Web Design de modo que todos podamos entender de dónde viene y por qué es tan importante en el mundo actual del desarrollo.

Responsive Web Design es la técnica que nos permite crear sitios adaptables a las condiciones del ordenador o dispositivo desde donde se van a acceder, sobre todo en lo que tiene relación con la pantalla del sistema donde se están visualizando. Aunque el término es suficientemente general como para poder referirse a una adaptabilidad en torno a cualquier condición. En este artículo vamos a intentar aclarar el término para que los lectores sepan bien a qué nos referimos.

Antes de pensar en la web, podemos entender el término "responsive" en general. Es la característica de un sistema que tiene respuesta hacia el medio que le rodea. No es algo específico de la web, sino que lo encontramos en el día a día. Por ejemplo, una luz que se enciende cuando alguna persona pasa al lado, una puerta que se abre cuando nos situamos cerca para entrar o un semáforo que se pone en rojo cuando pasa un coche a velocidad mayor de la permitida. Todo ello son sistemas "responsive", que tienen una respuesta conforme al medio o las condiciones donde se encuentran.

La traducción podría ser "Responsivo", que sí se encuentra en el diccionario de la Real Academia Española y significa "Perteneciente o relativo a la respuesta". Sin embargo en términos de una página web sería más adecuado usar la palabra "Adaptable", pues la página es capaz de adaptarse al dispositivo u ordenador donde se encuentra. En fin, el término es suficientemente conocido en inglés como para usar la palabra "Responsive" y que todos los que vayan a leer este texto capten perfectamente a qué nos referimos, por lo

que preferimos no traducirlo.

Marco actual donde surge el Responsive Web Design

Tenemos que pensar en las condiciones actuales de la Web. Antes existían solo ordenadores donde se consumían los contenidos y sin embargo hoy podemos consultar Internet en cualquier tipo de dispositivo. Móviles, tablets, televisores, libros electrónicos, relojes e incluso neveras permiten acceder a la Web y sus servicios. Son múltiples medios con distintas características y los diseñadores web deben preocuparse porque los sitios se vean bien en cualquiera de ellos.

¿Versión para móviles?

Tradicionalmente en la web los sitios estaban optimizados para ordenadores de escritorio. Luego, al popularizarse el acceso a Internet en móviles, o al introducirse el mercado de los tablets, lo habitual era la existencia de diversas versiones de un mismo sitio, creadas y optimizadas específicamente para teléfonos, tabletas y ordenadores personales.

Aún seguimos viendo esa solución en la Internet de hoy e importantes sitios mantienen versiones distintas para su web. Esa forma de trabajar era mayoritaria hasta el año 2010 aproximadamente, sin embargo, no es práctica. Nos obligaba a diseñar una web varias veces y mantener layouts para diversos tipos de dispositivos. El trabajo por tanto se duplicaba o triplicaba y no solo el de creación del sitio, sino lo que es peor, el de mantenimiento. Si el dueño de una web decidía crear una nueva sección, o reformular una existente, estaba obligado a actualizar las webs de cada uno de los sistemas para los que había creado versiones distintas.

Pero el problema se agravó con la llegada de la Internet en la televisión, en el coche, etc. ¿Qué nos toca? ¿hacer una versión web para cada uno de los dispositivos nuevos que vayan apareciendo? El camino no va por ahí.

Para atender a las necesidades actuales y futuras, necesitamos crear sitios web que sean adaptables a cualquier medio donde queramos consumirlas, presentes y futuros. Tener una única web es la mejor situación y justamente la mejor ventaja que nos ofrece el "responsive".

Tecnología que surge para cubrir las necesidades de adaptabilidad

Aparentemente, el trabajo de adaptar una web no es tan complicado, pero hace falta tecnología que nos permita realizarlo. En la segunda y tercera generación de sitios y antes de la implantación de las CSS3, teníamos pocas herramientas para conseguir la deseada adaptabilidad. Existían los diseños fluidos y en DesarrolloWeb.com ya hablábamos de ellos en 2001, que en contraposición de los sistemas rígidos permitían tener layouts capaces de utilizar todo el ancho de la pantalla. Era lo más parecido a un sitio responsive de hoy.

Las páginas fluidas se construían con contenedores que tenían la anchura de la ventana donde estaban visualizándose. Sin embargo, estaban pensados para que una web se viese bien en pantallas de 800, 1024 o 1280 píxeles de ancho. No estaban ingenidados con las características de los móviles en la cabeza. Hacer que una tabla (en aquel momento aún se llevaba la maquetación con tablas y se estaba comenzando a ver la maquetación con CSS) o un elemento DIV tenga la anchura de la pantalla es muy sencillo, pues requiere solo un atributo de estilo, sin embargo, conseguir que tu diseño se vea bonito en cualquier dimensión ya no es tan fácil.

Sobre todo hay que pensar que con la llegada de los móviles el rango de dimensiones habituales de las pantallas es mucho más abarcante. Tenemos pantallas que van desde los 300 píxeles de ancho a los 2000 y pico. Hoy incluso hay pantallas que superan los 5000 píxeles de ancho. Si estiras un diseño concebido para 300 píxeles de ancho, para llevarlo a dimensiones del doble de anchura, el triple, o más, en la mayoría de los casos tendrás un diseño horrible. Como no había tecnología para poder crear sitios elásticos que se vieran bien en cualquier dimensión de pantalla, los autores de web establan obligados a crear diferentes versiones de las páginas.

En resumen, CSS3 vino para solucionar algunas de las necesidades actuales, por medio de nuevos atributos y construcciones capaces de responder al entorno donde se encuentran. Nos referimos sobre todo a las Media Queries o a atributos tan simples como max-width o min-width (aunque estos últimos pertenencen a CSS 2.1). Gracias a estas utilidades somos capaces de reaccionar ante distintas circunstancias como la anchura de la pantalla, ventana o contenedor donde están aquellos elementos a maquetar. Los veremos con calma a lo largo del Manual de Responsive Web Design.

Término por Ethan Marcotte

No podemos terminar este artículo sin dar crédito al diseñador que acuñó el término de "Responsive Web Design". Se trata de Ethan Marcotte que publicó un post en "List Apart" y un libro donde explicaba todas las circunstancias y técnicas relacionadas con el diseño adaptable. No es que sea el primer diseñador que pensó en la necesidad de adaptar el layout de las webs a las diferentes pantallas (de hecho los creadores del estándar CSS ya habían creado tecnologías para producir sitios adaptables), sino que fue el que usó ese término que hoy es tan popular.

En DesarrolloWeb.com somos de los pioneros en acercar el término en español a la comunidad, en diversos eventos en directo y conferencias. En EscuelaIT se ofreció el primer curso de Responsive Web Design en español.

En resumen y como conclusión podemos decir que "Responsive" engloba muchos campos, pensamientos, prácticas, tencologías y técnicas que pueden incluso crecer con el tiempo. A medida que aparecen nuevos sistemas y dispositivos singulares, surgen nuevas necesidades, a las que se da respuesta ya sea con técnicas reformuladas, usando tecnologías existentes, o con tecnologías y estándares nuevos. Todo ello lo iremos cubriendo en este manual.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *19/01/2015*
Disponible online en *http://desarrolloweb.com/articulos/que-es-responsive-web-design.html*

Diseño responsive es diseño web

La web solo se concibe si es adaptable, hablar de un diseño responsive es hablar de diseño web, no hay web si no hay responsive.

Este artículo pretende dar una visión personal, y no solamente mía, sino de la comunidad de DesarrolloWeb.com en general y por supuesto, de Daniel Martínez @Wakkos en particular. Adicionalmente queremos establecer una serie de consejos útiles para el diseño web en general que te servirán para hacer

diseños adaptables en particular.

Cuando comenzamos la corriente del "responsive" y gracias a Daniel realizamos los primeros eventos en directo en los que hace tres años ya lo veníamos diciendo.

"El diseño responsive es diseño web"

Ahora todos queremos hacer diseño Responsive y así es como debe de ser. Pero ojo, cuando haces un diseño responsive realmente no estás haciendo nada del otro mundo, sino simplemente lo correcto.

A donde voy es que hablar de responsive queda muy bonito, nos hace parecer más entendidos, pero realmente de lo que estamos hablando es de diseño web y nada más. La web es un medio donde participan todo tipo de terminales, dispositivos, mil tipos de pantallas, etc. Debemos pensar en todos ellos cuando estamos diseñando una web y no solamente en móviles o solamente en ordenadores de escritorio.

Es nuestro trabajo y responsabilidad como diseñadores. Es una manera de hacernos unfavor a nuestros proyectos, a nuestros clientes y por ende a los usuarios que visitan las webs que fabricamos.

Así pues, en adelante, cuando nos refiramos al "diseño responsive" (por nuestra parte a lo largo del Manual de Responsive) entendámoslo como una manera de decir que el layout de la página va a adaptarse a la ventana donde estamos visualizándola. Pero no perdamos la noción que es simplemente la manera de actuar que debería tener cualquier profesional que se dedique al desarrollo de sitios web.

"No se entiende el diseño web si no es responsive"

Consejos para que el diseño web adaptable sea más fácil de producir

Nuestra tarea a lo largo del Manual de Responsive es establecer las guías y explicar las técnicas para que cualquier persona pueda comenzar a implementar el responsive de una manera más sencilla, menos traumática.

No vamos a explicarte en este artículo las técnicas esenciales, sería imposible por la longitud del texto, pero sí darte algunos consejos que esperamos puedas seguir en adelante. Repara además que todas estas buenas prácticas son comúnes a cualquier trabajo de diseño web, no solamente a los diseños adaptables, por lo que servirán para cualquier diseñador.

No uses estilos in-line: Los estilos inline son aquellos que se colocan en el propio HTML, sobre todo en el atributo style de la etiqueta. También por supuesto es inadecuado usar cualquier atributo HTML que sirva para aplicar un formato, como align="center". Esos estilos inline van a ser un corsé que quizás no se adapte

a todas las circunstancias. Igual un texto que aparece centrado en cierta resolución y queda bien, resulta horrible en una resolución diferente y si tu etiqueta tiene marcado ese estilo inline no lo podrás cambiar cuando lo necesites, en tus media queries que veremos más adelante.

La web no necesariamente se debe ver igual en todos los dispositivos y navegadores: El responsive es justamente eso, adaptarse al medio, pero lo que quiero decir ahora es que una web no necesita verse igual en todos los clientes web. Debe poder leerse bien en todos y permitir a los usuarios consumir su contenido, pero no debemos empeñarnos en que se vea igual en Internet Explorer 8 que en Google Chrome. Entendamos que son navegadores diferentes y por tanto la web es normal que se vea diferente. No tratemos de marear la perdiz para conseguir un sin-sentido.

No diseñes para una plataforma: Cuando haces responsive o cuando haces web en general no se trata de diseñar u optimizar una web en un dispositivo, navegador o sistema operativo dado. Generalmente es una equivocación andar descubriendo si es Internet Explorer o si es Android o si es tal cosa o tal otra el sistema que nos visita. Si basas tu optimización en plataformas ¿Qué pasa cuando aparezca una nueva? o una versión nueva? vuelves a optimizar? Esto no quiere decir que para solucionar un problema puntual no puedas preguntarte si estás en tal sistema, algo que a veces puede salvarnos la vida, sino más bien decimos que no se debe asumir como una estrategia de diseño en general.

Usar el Javascript para lo que es: A veces tratamos de solucionar problemas con las herramientas que no son adecuadas y Javascript se presta a ello. Por ejemplo, una animación Javascript en vez de una animación CSS. O un diseño de filas en formato cebra, alternando colores. Si se puede hacer con CSS es preferible que apliques CSS, porque serás capaz de modificar esos estilos fácilmente dependiendo de las características del dispositivo que te visita, algo que no es tan rápido o tan sencillo si lo haces con Javascript. Eso no quita que se use Javascript en ciertas situaciones, sobre todo a modo de fallbacks, para facilitar una segunda vía en navegadores que no la soportan. Por ejemplo, hacer una animación con Javascript cuando el navegador del usuario no acepte las animaciones CSS. Aunque quizás antes nos debemos preguntar ¿realmente es necesaria esa animación en aquel sistema que no la soporta?

> **Nota:** Si te interesa saber la manera correcta de detectar las capacidades de los navegadores, deberías leer el Manual de Modernizr.

¿Ya te hemos dicho que no uses tablas para maquetar?: Creo que a día de hoy ya no es necesario, o no debería serlo, incidir en este tema. Pero por si las moscas, no cuesta nada volver a mencionarlo. No se debe maquetar con tablas. En vez de eso, usa contenedores como DIV, o si encajan todavía es mejor usar los semánticos como ARTICLE, SECTION, MAIN, etc. Es verdad que las tablas son muy cómodas para crear una estructura de columnas donde colocar contenido, pero no son nada flexibles para un diseño responsive. Todo lo que tengas en tablas lo vas a tener que cambiar tarde o temprano por otros tipos de contenedores, para luego mediante CSS definir la estructura de columnas, una o varias, según las dimensiones de la pantalla o el navegador. Así que conviene que cuanto antes te acostumbres a desechar las tablas, si es que todavía las usas. Esto no quiere decir que nunca más vas a poder usar la etiqueta TABLE, ni mucho menos. TABLE es ideal para mostrar información tabulada, por ejemplo un listado de registros, una tabla de características de producto, donde sabes seguro que siempre se van a visualizar los contenidos tabulados. En ese caso, hasta TABLE tiene un correcto valor semántico.

Nota: Ten en cuenta que incluso contenedores como DIV, o cualquier otra etiqueta que englobe contenido, puede tener un comportamiento de tabla, si es que lo necesitas para tu maquetación para hacer cosas como alineado vertical. En ese caso, puedes mediante CSS asignar ese comportamiento con display: table y compañía.

Unidades relativas: Si no lo haces ya, conviene que te vayas acostumbrando a usar unidades de CSS de las relativas, como %, em, rem. Esto te facilitará la asignación de espacios y tamaños más que las unidades absolutas como px, cm, pt… El motivo es porque en un diseño responsive no sabes el tamaño que vas a tener para desplegar un contenido. Por ejemplo, si asignas a una caja una anchura de 600px ¿qué crees que pasará si la ves en un dispositivo que tiene una pantalla de 320px de ancho?. Si en vez de ello hubieses definido esa anchura como 80%, no habrías tenido quizás problemas.

Volveremos seguramente a incidir sobre todos estos puntos en el futuro en el manual de Responsive Web Design.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *26/01/2015*
Disponible online en *http://desarrolloweb.com/articulos/diseno-responsive-web.html*

Los tres pilares de optimización del diseño responsive

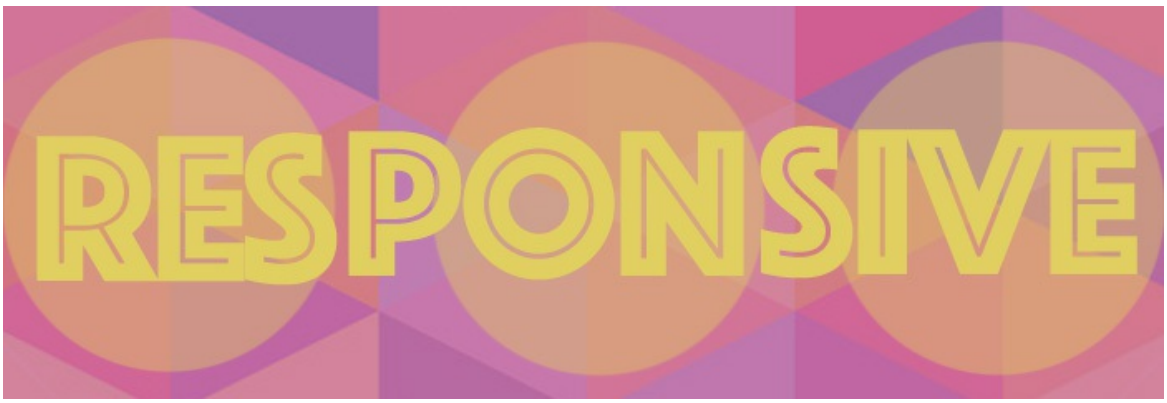
Cuando diseñamos una web debemos fijarnos en tres cosas fundamentales, diseño para todos los navegadores y sistemas, todas las resoluciones de pantalla y todas las velocidades de conexión.

Anteriormente explicamos que "diseño responsive" es diseño web, así que en líneas generales todo lo que vamos a comentar sirve también para el diseño web en general. Pues bien, cuando diseñamos una web, queremos que sea fácilmente usable y consumible y por ello debemos optimizarla en varios sentidos. El objetivo es el de siempre, que las personas aprovechen la página, tengan una experiencia de usuario grata y accedan a los contenidos que nosotros queremos.

Diseño responsive no es solamente hacer que una web se adapte tanto a las pantallas de ordenadores como a las de los móviles, que también lo es. En realidad hay mucho más por detrás y lo vamos a ver en forma de tres pilares fundamentales que describiremos a lo largo de este artículo. Los enumeramos a continuación:

Las técnicas responsive se focalizan en diseñar para:

- Todos los navegadores y sistemas
- Todas las resoluciones y tamaños de pantalla
- Todas las velocidades de conexión



Todos los navegadores y sistemas operativos

Los navegadores con los que se accede a Internet tienen diferencias los unos con los otros y debemos ser conscientes de ellas para que las páginas web se vean correctamente siempre. Hay multitud aspectos que deben ser tenidos en cuenta, pero en general nos debemos de asegurar que el contenido sea accesible en todos los browsers.

Pero las diferencias no solo están relacionadas con los navegadores, también con los sistemas operativos. Hubo un tiempo en el que las personas tenían mayoritariamente Windows y podíamos estar seguros que si nuestro sistema estaba optimizado para el SO de Microsoft estábamos alcanzando el noventa y tantos por ciento de los posibles usuarios o clientes. Sin embargo ese panorama ha cambiado mucho en la actualidad, ya que hoy el sistema Mac OSX es bastante popular, así como el sistema Linux.

Pero la cosa no acaba ahí, puesto que los teléfonos móviles o tabletas se usan para navegar y el uso de Internet desde dispositivos de movilidad representa ya la mayoría del tráfico en algunos países. Estos dispositivos tienen sus propios sistemas operativos, que también debemos tener en cuenta.

Todas las dimensiones / resoluciones de pantalla

Quizás sea éste el apartado que todos pensamos cuando nos hablan de "Responsive", diseñar una web que se adapta a la pantalla de ordenadores y dispositivos. No hace falta hablar mucho de ello, porque es la imagen mental que todos tenemos: Usar las media queries para que todos los ordenadores y móviles vean la web con los elementos dispuestos de manera que se facilite la lectura.

No queremos decir que este punto no tenga importancia, pero hay bastante más detrás. No obstante, la mayor parte de las técnicas que se aprenden en relación al lenguaje CSS van orientadas a este punto y es el que veremos con más detalle a lo largo del Manual de Responsive. Por todo ello, sobran las palabras de momento para describir este punto.

Todas las velocidades de conexión

Cuando pensamos en diseño móvil no debemos pensar solo en lo limitado del tamaño de la pantalla del dispositivo. No nos podemos olvidar que mucha gente navega en el móvil con conexiones de limitado ancho de banda, o incluso, aunque tengan ancho de banda amplio en muchos casos, casi siempre existen planos de datos que tienen limitada la cantidad de megas o gigas que se pueden consumir al mes.

Por tanto, hoy volvemos a encontrarnos en la necesidad de optimizar la web para que tenga menos peso y

sea rápida de transferir, como en la época del acceso a Internet en las líneas telefónicas con los módems de antaño. Debemos optimizar el código, HTML, CSS y Javascript, así como las imágenes y cualquier otro elemento que forme parte de la página. Pero no solo eso, también tenemos que "optimizar" el contenido. Aquí la palabra "optimizar" no estaría tan bien utilizada y aunque no deja de ser una optimización, se trata más que nada de elegir bien el contenido que se desea mostrar en cada momento.

Es más, sabiendo que las personas esperan de media tan solo 5 segundos para desistir en el acceso a una web, con mayor motivo debemos mantener acotada la cantidad de contenido a mostrar.

En este punto habría que señalar que la selección de un contenido correcto, conciso, claro, breve para que sea fácil de descargar, pero lo suficientemente extenso para que ayude correctamente a vender un producto, una marca o una persona, es una disciplina completa. Existen personas que se han especializado en estudiar el contenido y asegurarse que siempre sea el correcto e incluso hay personas que saben optimizarlo de cara a la venta, otras de cara a la comunicación corporativa, etc.

Además, cuando se trata de optimización de contenido muchas veces hay que tirar mano de programación informática. Existen técnicas puramente ligadas a la comunicación escrita, así como al aspecto estético, como la optimización de gráficos. Pero además hay técnicas de programación, como Ajax, que pueden ayudar a tener un contenido optimizado para una descarga de la página más breve.

También es importante la minimización del código Javascript o CSS. En resumen, existen muchos puntos interesantes, que veremos más adelante en otro artículo, para mejorar la carga de una web en términos de ajustar el contenido correctamente.

Conclusión

El diseño web necesita adaptarse a todas las condiciones donde puede ser consumido. No estamos hablando solamente de las pantallas, también del navegador o del sistema operativo del ordenador o dispositivo. Pero también el contenido debe ser pensado de manera que no se ofrezca información innecesaria que no agrega valor en la estrategia de comunicación. El contenido es el rey, pero también tiene peso, y no solo eso: mucho contenido puede despistar de lo realmente importante. En el caso de un comercio electrónico puede acabar provocando que el usuario se sature con muchas opciones ello derive en la pérdida de una venta.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *16/02/2015*
Disponible online en *http://desarrolloweb.com/articulos/tres-fundamentos-responsive-web-design.html*

Procedimientos habituales en el diseño adaptable

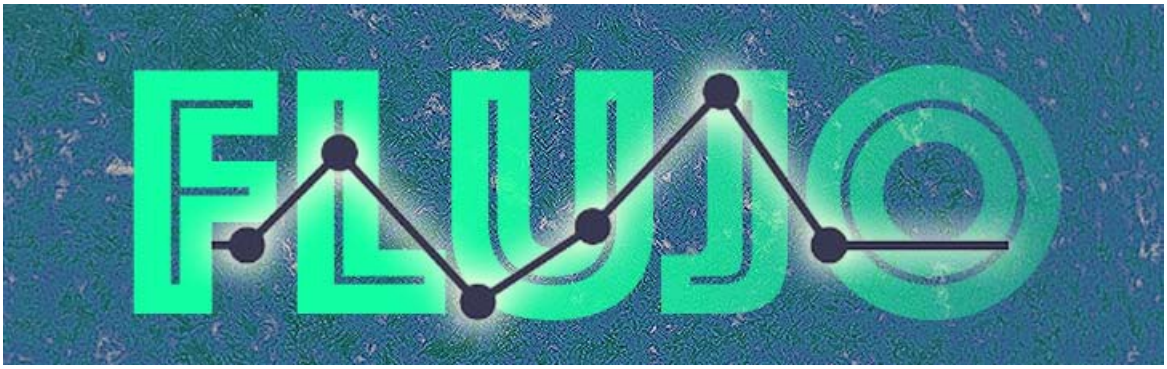
Ahora vamos a analizar los mecanismos o flujos de trabajo habituales cuando estamos implementando Responsive Web Design, indicando cuáles son los procedimientos a seguir y sus diversas variantes. El objetivo es entender qué pasos realiza un diseñador para crear un sitio web adaptable facilitando la labor de desarrollo y mejorando las prestaciones del diseño producido.

Flujo de desarrollo de un sitio responsive

Cómo se desarrolla un sitio adaptativo, aproximación de los procesos usados para la creación de un diseño Responsive Web Design.

En el Manual de Responsive hemos conocido muchas características deseables de los sitios de Internet de hoy en general y de los sitios adaptables en particular. Ahora vamos a comenzar con una aproximación al flujo de trabajo que se podría realizar para la creación de un "layout responsive". Claro que hay muchas técnicas que deberemos conocer para poder producirlos, ya en código, pero no nos vamos a meter todavía con explicaciones demasiado técnicas, sino a estudiar los posibles procesos de diseño.

Con el término "proceso de diseño" de momento nos referimos al procedimiento con el que se puede construir un sitio, desde su concepción hasta su ejecución. Aunque de momento nos quedaremos en las filosofías o estrategias posibles para solucionar los problemas de la web de hoy.



Problemática de la web de hoy

Ya hemos hablado de ella en artículos anteriores, por lo que no vamos a introducirnos de nuevo en las circunstancias de la web actual, solo cabe recordar que hoy al diseñar "páginas web" tenemos que conseguir que éstas sean medianamente inteligentes para poder adaptarse a cualquier tipo de situación.

Debe quedar claro que existen decenas, o cientos, de dispositivos que se conectan con un sitio web y que los desarrollos deben poder verse correctamente en todos ellos. Esa variedad de ordenadores y dispositivos

tienen unas características muy distintas y además un número siempre creciente.

Pasos para diseñar un sitio responsive

Deben de existir decenas de aproximaciones o flujos de trabajo para desarrollar un sitio "responsive" que tengan en cuenta la problemática actual descrita anteriormente. Aquí ya depende de cada diseñador y de sus costumbres, preferencias o formación. Vamos a introducir a continuación aproximaciones genéricas que son comúnmente aceptadas y aplicadas a nivel profesional y más adelante en futuros artículos nos dedicaremos a describirlas en detalle y apreciar las ventajas e inconvenientes de cada una.

> **Nota:** Suponemos que antes de comenzar se tiene claro el objetivo del sitio, se ha hecho un prototipo (wireframe), en papel o cualquier otra herramienta de prototipado y tanto el cliente como el desarrollador han llegado a un consenso sobre cómo debe ser la web y qué elementos debe contener. Este punto no es trivial y sobre él hablaremos más adelante, cuando expliquemos filosofías como "Mobile First", pero básicamente podemos adelantar que debemos ser bastante concisos con los contenidos que se deben mostrar, eliminando aquello que sea superfluo o innecesario. Explicaremos más adelante que un diseño responsive tenemos que comenzar por montar la web para que se vea bien en móviles y si cargamos excesivamente de contenido la página no nos va a caber todo en la pequeña pantalla de un dispositivo. Es una tendencia habitual que el cliente o el equipo de marketing quiera incorporar mucha información, pero hay que pensar que no siempre es necesario e incluso habrá ocasiones en la que demasiada información pueda ser contraproducente. Así que este proceso comienza por racionalizar el contenido que debe tener la web, algo que no vamos a tratar en este artículo.

1) Crear un HTML con el contenido que deseamos mostrar

En cualquiera de los caminos debemos partir de un mismo HTML. Como sabes, el contenido se escribe con HTML y debe ser común para todos los tipos de dispositivos e incluso para los ordenadores con pantallas enormes. El lenguaje CSS es el que nos permite aplicar un formato adecuado para la presentación de ese contenido y allí es donde podremos aplicar nuestras reglas, que permitirán que el diseño se adapte a cada tipo de sistema donde deba ser consumido el contenido.

> **Nota:** Desarrollar con HTML distinto para lo cada tipo de sistema, móvil, tablet, ordenador de escritorio, etc., dijimos que no era la idea detrás de las técnicas responsive. Lógicamente estamos hablando de un mismo HTML para un sitio web.

A ser posible, el HTML tiene que ser bastante semántico, algo que aporta el HTML5. Es algo deseable por diversos motivos como ya hemos hablado en decenas de artículos en DesarrolloWeb.com y programas en directo.

Como resultado de este punto debes tener un HTML sencillo, en el que habrás situado el contenido completo de tu sitio y aquellas interfaces para las funcionalidades necesarias. Me refiero principalmente a los bloques de contenido, puesto que lo más seguro es que hayas colocado textos falsos (los típicos lorem) e imágenes que posiblemente no tienen nada quever, con la intención de ocupar los espacios que vas a usar en tu diseño. Lo que tendrás también son las imágenes de cabecera o pie, como el logo de la empresa, iconos sociales, barras de navegación, etc. En resumen, todo lo que se ha acordado en el prototipo que deberías incluir en la página web a construir.

De momento, si tienes dudas sobre si un contenido debería estar o no en el diseño inicial, es mejor que no lo pongas. O si es un contenido que solo se va a ver en ordenadores de escritorio, tampoco lo pongas, puesto que nos vamos a centrar primero en encajar todo en las pantallas pequeñas.

2) Aplicar CSS

Hacer el HTML es la parte menos divertida. Con el CSS comenzará la fiesta y podrás empezar a disfrutar la mejor parte del diseño web. Como sabes, el CSS sirve para aplicar un formato a la página, tanto en lo que respecta a la posición de los elementos como al estilo o aspecto que deben tener. En este punto debes comenzar a aplicar los estilos necesarios a la página, poco a poco, para que comience a verse como tú querías.

A lo largo del Manual de Responsive de DesarrolloWeb.com iremos explicando cada una de las técnicas y herramientas que te ayudarán a construir un CSS optimizado que nos permita crear un layout adaptable. Existen mucha información que seguramente querrás saber, pero es imposible contarla toda en el mismo artículo. De momento vamos a describir las dificultades que encontrarás, pues no solo debes conocer a fondo el funcionamiento de las CSS sino también cómo los navegadores lo soportan, pues no todos entienden las mismas reglas de estilos.

Cada navegador entiende un conjunto de reglas de estilos y aquellas que no soporta, por desconocerlas, simplemente las ignora. Ahí es donde comienza la parte más compleja del diseño web y donde podremos aplicar principalmente dos flujos de trabajo diferentes. Pero antes de explicarlos queremos analizar los dos tipos de circunstancias en las que debemos encontrar soluciones.

A) Recursos de diseño estéticos

Son aquellos estilos que se agregan para mejorar la estética de un sitio, como fuentes tipográficas especiales, cajas con esquinas redondeadas, sombras en textos o cajas, etc. Este tipo de elementos no requieren una especial adaptabilidad. Son menos preocupantes, porque no van a afectar a la accesibilidad de un contenido. Son interesantes para que un sitio se vea especialmente bonito, pero si un navegador no puede renderizar ese tipo de estilos no representará un problema de gravedad. Por ejemplo, si Internet Explorer 8 no es capaz de mostrar un borde redondeado, no va a pasar absolutamente nada, puesto que los usuarios serán capaces de ver las cajas, aunque tengan esquinas con ángulos rectos.

> **Nota:** Aquí unos y otros pueden tener opiniones o necesidades distintas. Quizás nuestro cliente quiere que la web se vea igual en un navegador de última generación y en un navegador anticuado como IE8. Quizás nuestro jefe lo exija, pero en nuestra opinión (y la de la mayoría de los diseñadores más importantes del momento) tenemos otras cosas más importantes en las que centrarnos. Hemos hablado en decenas de programas en directo en nuestro canal de Youtube sobre estas circunstancias y nuestro objetivo no es discutir una vez más sobre ello. Aunque, de manera objetiva, lo más importante de una web es el contenido y lo que deben exigirnos los jefes, clientes, y nosotros mismos, es que el contenido sea accesible desde cualquier sistema, no tanto que se vea igual. Si hay un navegador que no soporta sombras, por ejemplo, no se acaba el mundo. Al contrario, deberíamos ser conscientes de las diferencias de cada uno y aceptarlas.

B) Diseño de layout

Son aquellos estilos que se encargan de presentar la información con una estructura definida, por columnas,

cabecera, pie, etc. El layout, al que nos referimos también como plantilla, nos permite jerarquizar la información y desplegarla de manera que el usuario la pueda entender mejor, pueda apreciar la importancia de cada cosa y centrarse en lo que nosotros preferimos que se aprecie más. Por ejemplo, si una web tiene una barra lateral donde hay banners y otras cosas con menor importancia (lo que conocemos como "aside") en todos los sistemas deberíamos poder remarcar la menor importancia de ese contenido. Si un navegador no es capaz de situar esa barra lateral en su lugar, por lo menos debemos centrarnos en que no aparezca antes que otras informaciones más importantes. O incluso más importante sería que, si un navegador no consigue respetar el diseño de layout, que por lo menos no aparezcan unos elementos encima de otros, dificultando la lectura del contenido.

En resumen y Como sabes, en el mundo de la web, a todos los niveles, **lo importante es el contenido. Si la estética de una web no es igual en todos los navegadores podemos seguir viviendo tranquilos. Sin embargo, si hay diferencias de layout éstas pueden provocar que el contenido no sea accesible**, o entendible, y eso sí representaría un problema mayor que deberíamos corregir.

Aproximaciones para resolver los problemas de compatibilidad del CSS

Hemos dicho que nos debemos preocupar más por los problemas de layout derivados de las diferencias de navegadores y su soporte con CSS, aunque estas técnicas también pueden ayudarte a encontrar vías de solución de problemas estéticos.

Ahora deberíamos introducir dos nuevos conceptos que son variaciones diferentes para resolver un mismo problema, las incompatibilidades en el CSS: Progressive Enhancement o Graceful Degradation. Sin embargo, son conceptos y técnicas lo suficientemente importantes para verlos de manera independiente, por lo que lo dejaremos para el próximo artículo. Aunque de momento cabe decir que estas técnicas no son algo específico de Responsive Web Design, sino que las aprendimos cuando estudiamos HTML5 en el diseño web.

Si quieres saber más te recomendamos ver el vídeo sobre HTML5 y compatibilidad con los distintos navegadores.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *02/02/2015*
Disponible online en *http://desarrolloweb.com/articulos/flujo-desarrollo-responsive-web-design.html*

Progressive Enhancement Vs Graceful Degradation

Dos técnicas, o flujos de trabajo, que nos ayudan a resolver los problemas de compatibilidad de los distintos navegadores frente al estándar CSS, explicadas en el marco del diseño adaptable.

En el artículo anterior estuvimos analizando el Flujo de desarrollo de un diseño responsive y aunque quedó claro que es tema amplio, motivo de estudio en diferentes artículos que vamos a abordar durante el Manual de Responsive web design, comentamos que existían dos variantes comúnmente usadas para solucionar los problemas de compatibilidad entre los navegadores.

En el presente artículo vamos a expolar tanto la aproximación Progressive Enhancement como Graceful

Degradation, aunque antes debe quedar claro que se trata de dos técnicas que no se usan solamente en el diseño responsive, sino que las conocimos ya cuando hace años comenzamos a explicar HTML5 en DesarrolloWeb.com.

Básicamente, lo que hacemos con estas dos aproximaciones, es solucionar las diferencias entre navegadores cuando interpretan un mismo código. Como sabes, a lo largo de los años se han ido presentando diferentes versiones de navegadores que son capaces de entender un conjunto diferente de etiquetas HTML, atributos de estilos CSS y Javascript. Ante las diferencias de navegadores, sus versiones y los sistemas operativos donde funcionan, y con la intención de hacer un contenido accesible para todos ellos, nos vemos en la necesidad de aplicar estas técnicas, que nos permiten que un contenido se vea correctamente en todos los sistemas posibles.

Ambas técnicas te pueden ayudar, incluso puede que prefieras en un mismo proyecto aplicarlas a la vez para solucionar diferentes problemas. Todo depende de tus preferencias o las necesidades del desarrollo. Ahondaremos sobre este punto al final del artículo. Pero comencemos describiendo las aproximaciones motivo de estudio de este texto.

> **Nota:** Como decimos, Progressive Enhancement y Graceful Degradation son aproximaciones para solucionar problemas de compatibilidad entre HTML, CSS y Javascript (que son los lenguajes que entienden los navegadores). No obstante, en el marco del diseño adaptativo, nos interesa más que nada el CSS y por ello es en términos de CSS que vamos a conocer estas filosofías.

Progresive Enchancement

Se diseña un sitio web centrándose en los sistemas más anticuados, y por tanto con menos soporte de las CSS. El CSS que utilizaremos será el básico que está disponible en esos viejos sistemas y simplemente nos preocuparemos porque el contenido sea accesible. A partir de ahí iremos implementando más complejidad al diseño, aplicando CSS que soportan solo los navegadores estándar actuales. Ese CSS no será soportado por sistemas antiguos, por lo que simplemente no les afectará. Como último paso aplicaremos CSS que se pueda ver en las versiones punteras de los navegadores, quizás sean las versiones experimentales que soportan un CSS que quizás todavía no forma parte del estándar actualmente recomendado para su implementación por la W3C.

En esta aproximación partimos de un mismo HTML y vamos agregando capas al desarrollo, preparadas para los distintos navegadores del mercado. A medida que nos centramos en un conjunto de navegadores más avanzado, vamos agregando nuevas capas que aprovechan las características más novedosas y que

simplemente son ignoradas por los navegadores más antiguos. Esta técnica lo que produce es una "mejora progresiva" de los sitios web, visible en los navegadores que la soporten, manteniendo un producto de base accesible para los navegadores menos actuales.

Por concretarlo con un ejemplo, podrías comenzar en una primera etapa por estilizar tu HTML con CSS compatible con todos los navegadores históricos, probando el desarrollo en Internet Explorer 8, por ejemplo. Una vez tengas tu web modesta pero correcta, pasarás a un navegador más actual, por ejemplo Internet Explorer 9 o superior (IE 10, 11 o el que sea que desees focalizarte) o incluso en versiones actuales de Firefox o Chrome, aplicando reglas de estilos que sabes que no están soportadas en IE8 pero que sí forman parte del estándar presente. Después de ese segundo paso tendrás una web bonita, que se comportará fantásticamente en navegadores que soportan el estándar actual. Podrás ir un poco más lejos, en una tercera etapa, si lo deseas y aplicarás estilos CSS experimentales, que solo están disponibles en determinados navegadores más punteros, quizás utilizando los prefijos propietarios de las reglas de estilo que todavía no están en el estándar recomendado.

Graceful degradation

Esta aproximación recorre más o menos el camino en sentido inverso. Básicamente desarrollas el CSS de tu web (recuerda que el HTML es el mismo siempre) atendiendo a las características más actuales de los navegadores. Te centras en hacer un sitio web tal como a ti te gustaría que se viera, con todos los detalles de diseño que has pensado aplicar para tu asombrosa web. Durante esa primera etapa probarás tu web en un navegador actual, aplicando todo tipo de reglas, aunque sepas que algunas no se van a poder ver en navegadores o sistemas operativos anticuados.

Una vez estás satisfecho con el diseño que te ha salido y has visto que en las versiones de navegadores modernos se ve tal como deseabas, a continuación pruebas el sitio web en navegadores menos avanzados y realizas las tareas que sean necesarias para que el sitio se vea decentemente.

Esas tareas de adaptación de la web para sistemas históricos en ocasiones incluyen la instalación de algún script Javascript que permita que los navegadores antiguos entiendan características de las CSS que no son soportadas de manera nativa. Quizás incluso definirás estilos alternativos que deben aplicarse solo a los navegadores que no entienden los estilos que tú querías aplicar en un principio.

> **Nota:** Los scripts Javascript que te sirven para suplir las carencias de los navegadores antiguos se llaman "Polyfills" (rellena-huecos) y la estrategia correcta, para su instalación únicamente en navegadores donde se necesiten, pasa por usar una librería como Modernizr, de la que hemos hablado ya en el Manual de Modernizr.

Como decimos, comienzas desarrollando tu web y probándola en un navegador actual, Chrome o Firefox son los principales candidatos, puesto que se actualizan automáticamente y por lo tanto es de suponer que tengas su última versión, pero también podrías hacerlo en Internet Explorer en versión 10, 11 o la más moderna que encuentres en este momento. Luego la vas probando en navegadores antiguos y vas solventando los problemas de visualización que vayas encontrando. Puedes probarla en una segunda etapa en Internet Explorer 9 y en una tercera en Internet Explorer 8 y así hacia abajo hasta donde quieras llegar.

¿Cuál es la mejor?

Los diseñadores más destacados apoyan más la aproximación basada en Progressive Enhancement porque opinan que los flujos de trabajo están más optimizados. Hay una lectura de estas dos prácticas en contraposición que es interesante y que debemos pensar.

Progressive Enhancement parte de la base de un diseño básico y accesible. A medida que se va disponiendo de tecnología superior somos capaces de añadir capas por encima al producto anterior, con nuevos estilos que solo se verán en los navegadores más modernos y avanzados.

Por su parte, Graceful Degradation ya comienza con un nivel de complejidad grande y el proceso de conseguir un diseño que se vea correctamente en navegadores antiguos se realiza muchas veces en base a parchear el desarrollo de manera que se adapte también a navegadores antiguos.

Bajo ese prisma podríamos decir que Progressive Enhancement mira hacia adelante en el flujo de desarrollo, mientras que Graceful Degradation empieza delante y mira hacia atrás. Teóricamente, cuando saquen en el futuro nuevos clientes web con nuevas características sería más fácilmente incorporadas en el diseño con una aproximación Progressive Enhancement, añadiendo simplemente una capa por encima a nuestro CSS. Por tanto, mirar hacia el futuro nunca está de más.

> **Nota:** Técnicamente y en términos de diseño CSS añadir una capa por encima no es más que añadir al final del código CSS nuevos estilos. Al estar al final del código CSS, y debido a la cascada, esa capa de estilos nuevos mandarán sobre los anteriores, aunque solo los interpretarán los navegadores nuevos, que los conocen. Los antiguos simplemente los ignorarán.

Algunos diseñadores son más radicales en este sentido y se oponen al graceful degradation bajo el argumento que los profesionales no deberíamos perder tiempo en solucionar los problemas derivados de navegadores antiguos, aplicando parches o "polyfills", porque eso provoca que los usuarios de navegadores obsoletos no se preocupen por actualizar sus sistemas.

Mi visión personal

Entre las dos aproximaciones comentadas, Progressive Enhancement o Graceful Degradation, en mi opinión personal no hay una que sea la mejor o la más adecuada. Insisto que esto ya es una opinión y que unos diseñadores o desarrolladores pueden pensar de una manera o de otra. Incluso puede haber profesionales que mezclen una y otra filosofía llegando a nuevos flujos de trabajo perfectamente válidos.

En mi caso, cuando desarrollo una web, hoy prefiero centrarme en navegadores actuales, diseñando como realmente quiero que se consuma la web y olvidándome de los navegadores antiguos. Aplico el CSS que realmente quiero que tenga la web y luego, cuando ya la tengo lista, la pruebo en Internet Explorer, en versiones antiguas, modificando aquellas cosas que son necesarias para que se pueda consumir correctamente, ya sean Javascripts o estilos CSS alternativos. Osea, prefiero implementar Graceful Degradation.

> **Nota:** Para colocar estilos alternativos en navegadores antiguos tienes la posibilidad de usar Modernizr, que trabajando de la manera correcta no te dará ningún problema de machacar estilos que realmente quieras para los navegadores avanzados.

Yo personalmente disfruto más diseñando una web pensando en un navegador actual y luego me tomo la tarea no tan agradecida de "apagar fuegos", solventando los problemas de visualización de navegadores antiguos. Para que nos entendamos, es como comenzar la comida por los postres y luego comerte los platos que te gustan menos, quizás no sea lo mejor, pero es como más me gusta trabajar.

Además, no veo problema en mezclar ambas aproximaciones. Puedes haber creado el diseño de un sitio web siguiendo la filosofía Graceful Degradation y dos años después aplicarle por encima una capa nueva, con CSS pensado para navegadores, o versiones, que hayan aparecido recientemente, lo que sería Progressive Enhancement.

¿Y los tamaños de pantalla? ¿estas técnicas no aplican?

Como te habrás dado cuenta, todavía no nos estamos preocupando de los diferentes tamaños de pantalla, ya habrá tiempo de hablar de ellos en el Manual de Responsive de DesarrolloWeb.com. De momento esta aproximación queda más que nada en lo que sería un flujo de trabajo comúnmente aceptado.

No obstante, algunos autores también identifican las técnicas Progressive Enhancement / Graceful Degradation en relación al diseño adaptable. Lo hacen en el sentido de diseño "Mobile First" o "Desktop First", osea, si se diseña pensando en el móvil al principio y luego en ordenadores de escritorio, o bien se diseña primeramente para pantallas grandes y se va adaptando a pantallas pequeñas. En este caso "Mobile First" sería más próxima a la filosofía Progressive Enhancement y "Desktop first" más próxima a Graceful Degradation.

Sin embargo, no veo mucha relación de estos términos en este sentido, del diseño para distintos tamaños de pantallas. En este área no encuentro mucha discusión y hay un consenso prácticamente total en que la aproximación adecuada es "mobile first", de la que hablaremos más adelante con detalle.

Conclusión

Como consideración final insistir en que no todo es blanco o negro, cada diseñador o desarrollador debe conocer ambas aproximaciones para decidir cuál es la que se adapta a su modo de pensar o de actuar. Incluso puede que en determinados proyectos o aplicaciones web sea mejor una aproximación que la otra. Lo que todos coinciden, o al menos la mayoría de los grandes diseñadores, es que debemos centrarnos en adaptar los sitios para solventar aquellos problemas derivados del layout y no tanto adaptar los recursos estéticos para que una web se vea igual en todo tipo de navegadores.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *08/02/2015*
Disponible online en *http://desarrolloweb.com/articulos/progressive-enhancement-graceful-degradation.html*

Mobile First

Mobile First es una filosofía, una manera de encarar el trabajo y una forma de facilitarnos la labor durante el diseño responsive, comenzando siempre por los dispositivos, con pantallas menores.

"Mobile First" es en realidad un concepto bastante simple: diseñar pensando en los móviles primero. Así se resume este artículo que viene a explicar los motivos por los que debemos comenzar siempre centrándonos en los dispositivos con pantallas más pequeñas y generalmente con menor ancho de banda disponible para la navegación.

Cuando diseñas un sitio web responsivo debes tener en la cabeza una enorme cantidad de contextos donde tu contenido va a ser consumido. Para no perderse y para al final conseguir una solución optimizada para todo el mundo debemos aplicar la filosofía Mobile First.

A lo largo de todo el Manual de Responsive Web Design vamos dando pinceladas que nos orientan hacia la idea del mobile first. Desde la elección del contenido, quedándose con la información más esencial y rechazando lo superfluo, hasta las etapas de la ejecución de un diseño, se hacen con esta filosofía en la cabeza.

Contenido

En un móvil no te cabe toda la información del mundo y no tiene sentido que un usuario deba hacer un scroll casi infinito hasta encontrar aquello que resulta de importancia. En el artículo de los tres pilares fundamentales del diseño responsive ya hablamos bastante acerca de esto y explicamos muchos motivos por los que no debemos concentrar en una página mucha cantidad de información.

Ahora, bajo la filosofía de Mobile First agregaremos la tarea de realizar una "especificación del contenido" de una página, que contiene simplemente un listado (en palabras) de todos los contenidos que queremos englobar dentro de una página, ya sea la home o una de las páginas interiores del sitio. Esa relación de contenido tenemos que pensarla con el móvil en la cabeza y su pantalla de dimensiones reducidas, que no es capaz de albergar una cantidad enorme de información.

> **Nota:** La especificación del contenido no se refiere a nada gráfico, ni siquiera a esquemas de información, jerarquías, secciones del sitio en el árbol de contenido o site maps. Por supuesto tampoco a wireframes o prototipos donde se distribuya la información. Es simplemente un listado de elementos que podríamos escribir con letras en un block de notas. Es un inventario de elementos, algo como logotipo, buscador, navegador con las principales secciones, imagen con el producto principal o los 3 productos principales, etc.

Por supuesto, esa definición de elementos la debemos hacer con el cliente y seguramente saldrán muchas cosas y quizás algunas sean prescindibles. En la cabecera en un móvil igual no cabe el logotipo + navegador

+ formulario de login + buscador. Entonces debemos intentar reducir todo aquello que es superfluo o prescindible, eliminándolo del listado. Quizás lo podemos sustituir por un icono de "opciones" y, una vez pulsado que se vean aquellas cosas que no entran en la página para móviles.

Diseño

La etapa creativa, cuando se diseña la interfaz, ya sea en papel durante el prototipado o en HTML + CSS en la ejecución, también se hace comenzando por aquello que veríamos cuando la web se consulta desde un móvil.

Resulta mucho más sencillo diseñar una web para un móvil, que tiene muy pocos elementos, que para un ordenador de escritorio donde tenemos espacio para colocar muchas otras cosas. Por ello es una buena idea que, cuando diseñemos, nos pongamos delante de un espacio reducido y veamos qué cosas son las que entran en el diseño y qué cosas pueden ser prescindibles.

Si estás ejecutando el diseño en HTML + CSS comenzarás reduciendo la ventana del navegador a las dimensiones de anchura que pueda tener un móvil, quizás unos 320 píxeles de ancho (ahora no vamos a discutir sobre dimensiones), para acomodar los elementos de tu inventario de contenido. Quizás te des cuenta que necesitas prescindir de nuevos elementos que son difíciles de acomodar y que no eran tan esenciales después de todo.

> **Nota:** También puedes conectar el móvil al ordenador y ver directamente en tu dispositivo cómo se está visualizando la web. No cabe duda que el resultado y la experiencia de navegación es mucho más fiel cuando ves lo que estás desarrollando directamente en un dispositivo que tiene las dimensiones de anchura aproximadas a las que estás focalizando tu diseño. Existen varias herramientas que te pueden ayudar a hacer de puente entre el móvil y el ordenador, de modo que puedas ver en vivo todos los cambios que estás haciendo en tus CSS o en tu HTML. Ahora no es el momento de explicarlas, pero no os preocupéis pues tendremos tiempo para ello en un futuro artículo en DesarrolloWeb.com.

Una vez tengas el diseño que se vea bien en una ventana con anchura reducida, entonces irás estirando la ventana hasta que te des cuenta que tu diseño comienza a verse feo. Entonces es el momento de poner un "breakpoint" (punto de ruptura) y definir media queries para pantallas que tengan dimensiones mayores que esa anchura.

Nota: Hablaremos más adelante sobre los breakpoints y las media queries. De momento nos quedamos más en la parte de conceptos y flujos de trabajo y en poco tiempo podremos ver cómo se implementan en código.

Mobile First es una cuestión de comodidad, pues tu diseño irá de menos a más y por tanto gradualmente irás acomodando las cosas y adaptando el layout a las necesidades de cada anchura de pantalla. Pero no solo eso, también es sentido común, pues es mucho más fácil comenzar por una pantalla pequeña donde aparecen pocos elementos y pocos detalles e ir agregando cosas a medida que van cabiendo en el layout. Es decir, resulta más sencillo tener pocos elementos e ir agregando más cosas a medida que van cabiendo, que empezar por el diseño para pantallas grandes y luego ir quitando cosas cuando no te caben.

Es más fácil diseñar en pequeñito al principio. Cuando tienes más espacio verás que tus cosas siguen cabiendo. Si diseñas en grande, cuando tengas menos espacio las cosas no te entrarán y seguramente tendrás

que hacer muchos más esfuerzos para acomodarlas. En este punto todos los diseñadores están de acuerdo, así que no lo pienses mucho, comienza diseñando para móviles.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *23/02/2015*
Disponible online en *http://desarrolloweb.com/articulos/mobile-first-responsive.html*

MediaQueries

Cuando pensamos en diseño Responsive inmediatamente se nos vienen a la cabeza las mediaqueries CSS3, pues éstas son las herramientas fundamentales para especificar estilos con respecto a diversas condiciones del cliente web que está visitando el sitio. En los siguientes artículos te explicamos con detalle qué son las Mediaqueries y te ofrecemos información valiosa sobre cómo aplicarlas, creando los breakpoints más adecuados para tus layouts.

CSS Media Queries

Las Media Queries, incorporadas en CSS3, son las primeras construcciones del lenguaje CSS que nos permiten definir estilos condicionales, aplicables únicamente en determinadas situaciones.

Las Media Queries son, en una traducción rápida, consultas de las características del medio donde se está visualizando una web y nos sirven para definir estilos condicionales, que solo se aplicarán en caso que esa consulta del medio sea satisfactoria.

Son una de las herramientas fundamentales para implementar el "Responsive Web Design" y han llegado de la mano de las CSS3, convirtiéndose en un aliado fundamental de cualquier diseñador web.

En la mayoría de los casos las Media Queries sirven para definir estilos diferentes para distintos tamaños de la pantalla. Son sencillas de entender y aplicar, aunque el estándar es bastante sofisticado, con diversas posibilidades. Existen muchos usos, algunos no tan habituales en el mundo del diseño actual, pero que podrán tener su protagonismo en algún momento.

> **Nota:** A modo de curiosidad, pues realmente no se usa mucho en el diseño actual, las media queries pueden decirnos si el dispositivo tiene una relación de pantalla 4:3 o 16/9, o una profundidad de color dada, por ejemplo. En este artículo nos encargaremos de conocer los casos más comunes de uso y más adelante nos adentraremos en estas otras características adicionales.

Paso al frente de las CSS

Las Media Queries representan una evolución importante de CSS, puesto que suponen la primera estructura condicional en el lenguaje. Sabemos que CSS no es un lenguaje de programación, ni tiene motivos para parecerse, pero existen muchas cosas que se implementan en los lenguajes de programación y que nos vendrían bien en el desarrollo con CSS, como es el caso de los condicionales.

"Si ocurre ésto, entonces haz tal cosa"

Una construcción condicional como esta es tan útil y básica que, aunque CSS no sea un lenguaje de programación, necesita incorporarlas. Ejemplos de casos en los que nos vendría bien un condicional:

- Si la pantalla del usuario tiene estas características, entonces aplica estos estilos
- Si se imprime el documento en la impresora, aplica estos estilos.
- Si la pantalla del dispositivo tiene estas dimensiones y además está situado en posición horizontal (landscape), entonces aplica este CSS.

Situaciones como esas son básicas y los diseñadores pueden resolverlas usando las Media Queries. Son un paso al frente en cuanto a la separación entre el contenido y su representación, puesto que nos facilitan que el contenido de una página pueda adaptarse al medio donde se está reproduciendo, sólo a través de la definición de estilos, sin tener que tocar el HTML para nada.

Alternativas para implementar Media Queries

Aunque la utilidad es novedosa, la sintaxis es parecida a lo que ya conocemos de las CSS, por lo que nos resultará bien sencilla.

Para producir Media Queries debemos tener siempre en mente la expresión condicional, aquella que debe cumplirse para que se apliquen ciertos estilos. Además la expresión condicional puede tener incluso varias condiciones, usando operadores lógicos como "and" para combinarlas. De modo que las circunstancias que se deban cumplir para aplicar unas reglas CSS sean de lo más variadas.

Alternativa 1: La primera alternativa de las Media Queries es a través del atributo media de la etiqueta LINK. Como sabemos, esa etiqueta es la que se usa para enlazar una hoja de estilo con un documento HTML. En ese enlace podemos especificar condicionales que deben cumplirse para que los estilos enlazados se apliquen. Por ejemplo, que se esté imprimiendo un documento o que la pantalla tenga cierta anchura mínima.

Recordamos, la etiqueta LINK tiene esta forma:

```
<link rel="stylesheet" href="estilos-generales.css">
```

Pues ahora simplemente le podemos agregar el atributo "media" indicando la condición que se debe cumplir para que estos estilos se apliquen:

```
<link rel="stylesheet" href="estilo-imprimir.css" media="print">
```

Este atributo media="print" quiere decir que los estilos deben aplicarse sólo cuando la página se están mostrando para la impresión.

> **Nota:** Este uso ya lo vimos anteriormente en el artículo Estilos específicos para impresión.

```
<link rel="stylesheet" href="estilo-pantallas-grandes.css" media="(min-width:1200px)">
```

Este uso será seguramente más novedoso para ti. Quiere decir que esos estilos deben aplicarse sólo cuando la pantalla del usuario (En caso de ordenadores de escritorio, la ventana del navegador) tenga una anchura mínima de 1200 píxeles.

El problema de escribir tus Media Queries así es que tienes archivos de CSS separados. Es decir, aquellos estilos para impresión o para pantallas de 1200px están en archivos independientes, lo que es sencillo de administrar para nosotros, pero una mala práctica en términos de optimización de la web, puesto que se tienen que realizarse varias solicitudes al servidor distintas para traerse los CSS. En la práctica ralentiza la carga de la página en relación a hacer una única solicitud de un archivo CSS que contenga todo el código de los estilos.

Alternativa 2: Este método que vamos a ver ahora es más interesante y es el que se usa habitualmente a nivel profesional. Consiste en incorporar los estilos en una construcción @media donde se apliquen entre llaves todos los estilos que queremos para una consulta de medio dada.

```
@media (min-width: 500px) {
    h1{
        margin: 1%;
    }
    .estiloresponsive{
        float: right;
        padding-left: 15px;
    }
}
```

Como puedes ver, tenemos la sentencia @media en la que podemos indicar entre paréntesis las condiciones que deben cumplirse para que se aplique esta media query. En este caso será para pantallas que tengan una anchura mínima de 500 píxeles.

Luego entre llaves colocamos todas las reglas y atributos de estilos CSS que necesitemos aplicar en esta situación. Las reglas de estilos son las mismas que pondrías fuera de la estructura condicional. Cuando la sentencia entre paréntesis se evalúe como verdadera, se aplicarán todas ellas.

Operadores lógicos para las Media Queries

Para combinar diversas condiciones podemos usar los operadores lógicos, de una manera similar a como se usan en lenguajes de programación. Los que tenemos disponibles son:

*** Operador and:**

*** Operador not:**

*** Operador only:**

*** Operador or:**

```
@media (max-width: 600px) and (orientation: landscape) {
    h1{
        color: red;
    }
}
```

Esta regla se aplicaría para pantallas con una anchura máxima de 600 píxeles y cuando la orientación está en horizontal.

> **Nota:** Ten en cuenta que la mayoría de smartphones simulan tamaños de pantalla mayores, haciendo una especie de dimensiones virtuales que faciliten la lectura de webs que no están diseñadas para Responsive Web Design. Por ello, lo más seguro es que tengas que poner el "viewport" en el documento HTML a algo como:
>
> ```
> <meta name="viewport" content="width=device-width, initial-scale=1.0" />
> ```
>
> Sin ese viewport tu móvil simularía que tiene unas dimensiones de unos 980 píxeles, cuando quizás la pantalla solo tenga una anchura real de 320 o 500 píxeles. Este asunto ya se explicó con detalle en el artículo sobre el viewport.

Para ver en funcionamiento la parte del orientation (landscape o portrait), tienes que usar un móvil o tablet y cambiar la posición de la pantalla, para que esté en horizontal o vertical.

> **Nota:** Si tienes un ordenador que no reconozca el cambio de orientación la posición siempre será considerada será landscape.

```
@media tv and (min-width: 1200px){
    h1{
        margin: 10%;
    }
}
```

Esta regla aplicaría en dispositivos de tipo televisión y cuya resolución minima de anchura sea de 1200 píxeles.

```
@media (min-width: 600px),  handheld and (orientation: portrait) {
    h1{
        color: green;
    }
}
```

Este mediaquery servirá para pantallas de minimas dimensiones 600 píxel y también para todos aquellos dispositivos handheld que estén en posición vertical.

> **Nota:** Handheld es un término inglés que sirve para especificar aquellos dispositivos que son de mano. Pequeños ordenadores que se llevan en la mano, como los palm o agendas electrónicas que aparecieron antes de popularizarse los smartphones o tablets. Por mis pruebas handheld no aplica a los móviles o tablets.

Las Media Queries tienen muchas más posibilidades que no cubrimos con este artículo. Las veremos más adelante, pero sin duda con lo que ahora sabes podrás resolver el 98% de las necesidades que te puedas encontrar en el mundo del diseño web.

Si te interesa saber más sobre este tema, en este otro artículo de DesarrolloWeb.com te contamos la estrategia para aplicar las Media Queries en Responsive Web Design.

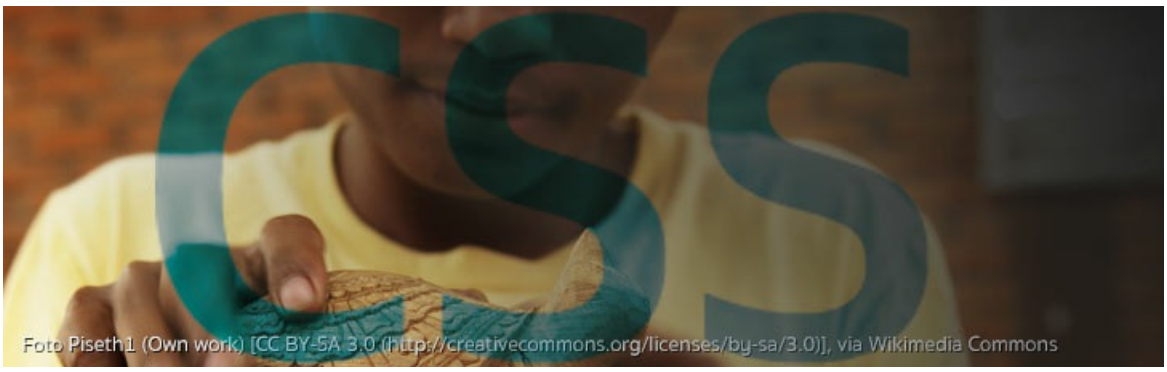
Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *07/01/2015*
Disponible online en *http://desarrolloweb.com/articulos/css-media-queries.html*

Guías de uso de las Media Queries en Responsive Web Design

Vamos a conocer muchas de las técnicas relacionadas con las media queries que puedes aplicar para sitios que se adapten a todo tipo de pantallas y medios.

En el Manual de Responsive Web Design de DesarrolloWeb.com hemos abordado ya muchos asuntos importantes y en concreto en el caso de las Media Queries las hemos analizado desde un punto de vista técnico, centrándonos en la sintaxis de las CSS para Media Queries. Ha sido genial, porque ya sabemos muchas cosas que podemos realizar gracias a las Media Queries y cómo implementarlas a nivel de código CSS.

No obstante sobre todo para las personas que se están iniciando hay muchas cosas que les arrojarán dudas o problemas y que vamos a intentar resolver en el presente artículo. Por tanto, estás ante un texto menos técnico pero lleno de consejos y guías para que puedas aplicarlo en tus diseños adaptables.



Configurar un buen Viewport

El viewport es una definición o declaración sobre diferentes parámetros de la visualización de una página web, que afecta sobre todo a los dispositivos. Se trata de una etiqueta META a través de la cual se puede definir si el usuario puede o no hacer zoom en la página, qué zoom inicial debe aplicarse al entrar, la anchura que debe simular la pantalla del dispositivo. Es algo largo de contar, si se quiere explicar bien, por lo que te recomendamos leer el artículo dedicado al Viewport para encontrar la información detallada.

De momento te puedes quedar con que el Viewport sirve para que la página no se muestre en pequeñito en el dispositivo (Habrás apreciado que cuando entras en un sitio que no es "responsive" se ve todo muy pequeño). Cuando trabajas con un sitio adaptable tienes que obligar al dispositivo a no simular que tiene más anchura de pantalla de la que realmente tiene y para ello tienes que definir un viewport adecuado.

Es un fallo común cuando estás empezando es no tener un viewport definido. El síntoma es que tu página no te está aceptando las media queries que estás definiendo. Si te pasa eso no te vuelvas loco y define un viewport adecuado. Consulta el artículo mencionado para encontrar la información completa sobre cómo se define.

El orden de colocación de los mediaqueries

Orden de las mediaqueries es importante y por tanto debes tener cuidado con ello. Para entender cómo ordenar tus media queries tienes que tener en cuenta dos cosas:

1. La cascada que, a nivel de CSS, afecta a todas las reglas de estilo 2. La regla comentada anteriormente que llamamos Mobile First

> **Nota:** Aunque es un conocimiento básico, lo aclaramos por si acaso. La cascada en CSS es aquella manera de procesar los estilos que produce que se puedan sobrescribir. Osea, si en un punto de tu hoja de estilos defines un CSS para un elemento y más adelante en el código CSS lo sobreescribes colocando cualquier otro valor en ese estilo, el que se tiene en cuenta es el que se definió por último lugar (a menos que uses "important" en la primera regla). En resumen, si dos valores en atributos de CSS se contradicen, por la regla de la cascada, el que se tiene en cuenta es el que se colocó por último lugar.

Así pues, el orden que se usa normalmente para las media quieries (MQ) es:

1. Colocar todos los estilos a aplicar de manera global, fuera de cualquier MQ. Los estilos que coloques sin

MQ serán como "MQ globales" que afectarán a todos los dispositivos, con todas las anchuras de pantalla. Por la regla del Mobile First, las configuraciones que pondrás en tu CSS global (ausencia de MQ) serán las que se apliquen a los móviles, osea, a los dispositivos con menor anchura de pantalla. (Ver aclaración más abajo) 2. Luego colocarás, como primer MQ, el estilo necesario para los dispositivos de la siguiente anchura de pantalla que necesites. 3. Irás colocando los MQ en orden de anchuras de pantalla ascendentes. Terminarás siempre por el MQ que coloques para las anchuras de pantalla mayores, ordenadores con pantallas de alta resolución.

> **Nota:** Esta guía, basada en la filosofía Mobile First, es la que pone en práctica la mayoría de los diseñadores, por considerarse más útil y versátil. Lógicamente tú podrías aplicar la filosofía que te venga en gana, siempre y cuando sigas la regla de la cascada, que es la que verdaderamente te marca el lenguaje CSS.

Rangos de media queries

Puedes perfectamente especificar rangos de media queries. Para ello usas min-width y max-width a la vez. Quizás no es algo muy habitual, pero lo mencionamos por si tuvieras necesidad de ello. La sintaxis sería la siguiente:

```
@media (min-width: 700px) and (max-width: 800px){
    .lateral{
        width: 33%;
        float: right;
        background-color: #6ee;
    }
}
```

Esos estilos para la clase "lateral" los visualizarías en anchuras de pantalla que cumplan las dos condiciones a la vez, osea, que su anchura mínima sean 700 píxeles y las anchura máxima sea 800 píxeles. Es decir, el rango de 101 píxeles que van desde los 700 hasta los 800 pixel.

Anchura de la pantalla del dispositivo

Hasta ahora hemos definido los media queries con respecto a una anchura de la pantalla que puedas tener en un momento dado, pero también puedes definirlas para la anchura total de la pantalla, independientemente de la configuración actual. Lo consigues con max-device-width y min-device-width y no afectan a la anchura de la pantalla actual, sino a la del dispositivo.

> **Nota:** Recuerda que cuando decimos anchura generalmente queremos referirnos a anchura o altura. Generalmente las media queries se aplican a la anchura, pero también las puedes aplicar a la altura, min-height, max-height. Recuerda también que con media queries puedes especificar si se aplican a dispositivos que estén posicionados en horizontal o en vertical (landscape o portrait). Todo eso se mencionó en el anterior artículo de las media queries.

Aclaración sobre las "Media Queries globales"

Antes hemos comentado acerca de "media queries globales". Este es un concepto "verdad a medias", puesto que no existen mediaqueries globales, que afecten a todas las resoluciones pantalla y todos los tipos de dispositivos. Pero si lo analizas más de cerca observarás que, si no existe ninguna media querie definida en tus CSS, entonces tus reglas de estilo estarán siendo aplicadas de manera global. Dicho de otro modo, cuando no defines ningún media querie en realidad estás definiendo estilos CSS que afectan a todos los posibles estados, pantallas, medios, dispositivos, etc.

Esa ausencia de Media Queries es justamente tu declaración "global", aplicable en todos los contextos, que afecta a todos los sistemas.

Conclusión

Con todo este conocimiento ya estás en condiciones de ponerte manos a la obra y aplicar media queries a tu diseño web y adaptarlo a tus necesidades. Hay muchas cosas por delante que conocer para aprovechar las posibilidades del Responsive Web Design, pero estamos seguros que podrás aplicar un montón de utilidades para adaptar todo tipo de sitios a todo tipo de pantallas. Seguro que disfrutarás de la experiencia. Seguiremos trabajando para llevarte muchos otros conocimientos y técnicas esenciales para el diseño web.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *04/03/2015*
Disponible online en *http://desarrolloweb.com/articulos/guias-uso-media-queries-rwd.html*

Breakpoints para diseño Responsive

Los puntos o medidas de anchura donde se pueden crear saltos en el diseño Responsive, llamados comúnmente breakpoints, a partir de donde aplicar las media queries para Responsive Web Desing.

Responsive Web Design son muchas cosas, es optimización para tres elementos fundamentales, navegadores y sistemas operativos, velocidades y anchuras de pantalla. En este caso vamos a tratar sobre la optimización para pantallas, los tipos de pantallas y los saltos que el diseño debe pegar para ir adaptándose.

Observarás que un diseño adaptable es responsivo a los cambios de la ventana del navegador. A medida que la hacemos mayor o menor la plantilla de la página va cambiando para adaptarse mejor a las nuevas dimensiones. Al estirar la pantalla observarás que, de una única columna en definiciones pequeñas, pasa a ser de dos columnas en una anchura de pantalla mayor, luego pasa a tener tres columnas cuando amplias todavía más el espacio y así indefinidamente, tal como el diseñador haya definido.

A esos saltos en los que un diseño cambia de un layout a otro, a medida que va agrandándose la ventana del navegador les llamamos breakpoints y se producen gracias a las media queries. En este artículo hablaremos de los breakpoints y sobre cuáles son los puntos ideales en los que realizarlos.

Los breakpoints no deben orientarse a dispositivos

Este es el punto más importante del artículo, puesto que generalmente las personas con menos experiencia en Responsive Web Design tienden a asumir que existen unas medidas estándar en las que se deben aplicar los breakpoints. Podría haberlas conforme a las dimensiones de diversos dispositivos, como iPhone, iPad, ordenadores con pantallas estándar y luego hablaremos de ellas, pero la realidad o la recomendación va por otro lado.

Como vimos en los capítulos de introducción en el Manual de Responsive, uno de los motivos por los que existe esta filosofía de diseño es el de no crear versiones distintas para distintos tipos de sistemas. No creamos una versión para móviles, otra para tabletas y otra para ordenadores. Eso debe haber quedado claro. Pues de la misma manera, no creamos breakpoints orientados a dispositivos. El motivo es el mismo: es imposible que puedas generar tantos saltos como tipos de pantallas y sus resoluciones, etc.

Si te empeñases en crear breakpoints con las dimensiones de dispositivos en la cabeza, vas a tener que crear saltos para cientos de dispositivos. Cuando saquen un nuevo modelo o una nueva marca seguro que no quieres actualizar tus saltos para que la web se vea bien en ellos. Es imposible alcanzar a todos los dispositivos, todas las marcas y modelos. Es imposible que conozcas todas las dimensiones de pantalla, las existentes hoy y las que puedan existir mañana.

Cómo saber el lugar donde colocar un breakpoint

Si no haces breakpoints orientados a dispositivos, ¿cómo saber dónde colocarlos? La respuesta te la da tu diseño. En otras palabras, breakpoint debe estar orientado al diseño particular de tu web.

Para saber dónde colocar los saltos con las mediaqueries debes estirar la ventana del navegador, partiendo de la ventana con dimensiones reducidas (mobile first) vas estirando la anchura hasta que comienzas a apreciar que tu diseño está viéndose peor. No es que los usuarios se dediquen a cambiar las dimensiones de la ventana del navegador, es algo que hacemos los diseñadores, para imitar rápidamente distintos tamaños de pantalla de móviles o tablets.

Toma cualquier página web adaptable, como la de DesarrolloWeb.com, y ponla en dimensiones de anchura pequeñas. Entonces agranda la ventana y apreciarás que en determinado momento surge mucho espacio vacío o que las columnas comienzan a ser demasiado grandes. Entonces, en ese momento, poco antes o poco después, verás que "zas", cambia el layout.

Osea, vas haciendo la página más ancha y cuando tu diseño empieza a empeorar, o los espacios comienzan a estar peor distribuidos, entonces es el momento adecuado de colocar un breakpoint.

Breakpoints grandes y pequeños

La regla de oro es no limitarnos a nosotros mismos a un número de breakpoints determinado. No debes escatimar a la hora de hacer breakpoints. No los veas como puntos donde cambias de layout, orientado a móviles, tablets o monitores de ordenador de alta definición. Como hemos dicho, no es ese el objetivo, crear versiones del diseño, aunque mantengas un único HTML. A veces un breakpoint puede ser simplemente cambiar la alineación de un elemento, la medida de un texto en un titular, etc. Los puntos donde colocas media queries pueden ser para cualquier cosa que necesites. A veces es algo tan simple como una imagen, que a partir de determinadas dimensiones, quieres que flote (float) para que el texto alrededor la rodee. O un aumento minúsculo en la fuente de la página. No importa que tan pequeño es el cambio, si ves que beneficia al aspecto de tu página, es motivo para crear un nuevo "media queries".

Por tanto, hay tanto breakpoints grandes, que te cambiarán el layout de la página y por tanto afectan de manera más radical al diseño y otros pequeños puntos de ruptura intermedios, que realizarán pequeños ajustes para solucionar detalles que merezca la pena retocar. Por todo eso, volvemos a remarcar el punto anterior, es importante que los breakpoints los hagas en función de tu diseño, apreciando el momento adecuado para producirlos, agrandando y empequeñeciendo la ventana del navegador para decidir cómo y cuándo.

Medidas estándar para media queries

Hay diseñadores que, a pesar de todo lo dicho anteriormente, aún se preguntan si existen medidas estándar para una página, en los que crear sus breakpoints. No podemos darte esas medidas, entre otras cosas porque no pretendemos contradecirnos a nosotros mismos, al menos no en este artículo. Pero no desesperes porque realmente no las necesitas.

Volvemos a insistir, las veces que haga falta, que las mediaqueries no se deben definir para optimizar un diseño para iPad o iPhone, etc. sino que deben ser aquellas que produzcan una adaptabilidad mejor del diseño. Y eso solo lo consigues conociendo tu propio diseño y no conociendo los soportes desde donde lo van a acceder. Recuerda además que cualquier colección de dispositivos que puedas tener siempre va ser parcial, puesto que es imposible reunir todos los tamaños de pantalla y resoluciones en ningún documento. Básicamente además porque necesitaríamos estar actualizando esa lista casi diariamente.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *05/05/2015*
Disponible online en *http://desarrolloweb.com/articulos/breakpoints-responsive-web-design.html*

Medidas estándar de pantallas para tus media queries

Listado de pantallas y viewports comunes de dispositivos y ordenadores de escritorio para hacernos una idea de hacia dónde podrían apuntar nuestras media queries CSS.

He pensado bastante antes de redactar este artículo, analizando hasta qué punto debería o no escribirlo.

Como sabemos, los breakpoints de un diseño Responsive Web Design para crear tus MediaQueries no deben orientarse a tamaños de pantalla estándar, sino debe ser algo específico para tu propio diseño web. Sin embargo, muchas personas quieren saber, al menos a modo orientativo, qué medidas de pantallas habituales se pueden encontrar en los dispositivos.

Lo que no quiero que se piense es que este punto contradice todo lo dicho anteriormente en el Manual de Responsive Web Design. Por eso es bueno insistir que estas medidas son solamente a modo orientativo, para aquellas personas que tienen curiosidad sobre las dimensiones de los dispositivos "estándar".

Recordando que cualquier colección de medidas para media queries está siempre desactualizada e incompleta, porque es imposible albergar todas y cada una de las medidas de pantalla que hay en el mercado de ordenadores y dispositivos y además, cuando se publique o lea este texto, quizás ya hayan aparecido nuevos dispositivos con dimensiones diferentes a las que se mencionan. Así que toma estas medidas de viewport como una referencia únicamente orientativa y sobre todo parcial. Entiende además que no pretendemos que diseñes para ninguna de estas dimensiones, puesto que sería un error.

Nota: Muy importante, lo que se muestra a continuación es el tamaño del Viewport. Las medidas en centímetros o pulgadas de la pantalla no tienen que ver con las medidas que el teléfono tiene en píxeles. Pero además, el móvil puede tener una resolución en píxeles diferente que su viewport, porque muchos móviles usan varios píxeles de pantalla para representar un píxel de imagen, debido a su "device pixel ratio" o el ratio de pixels que el dispositivo usa para mostrar un pixel de computable en medidas de viewport. Por ejemplo, la pantalla del Nexus 5 es Full HD, lo que implica una resolución de 1080 x 1920 pixels. Sin embargo, al mostrar una página web el Nexus 5 tiene un viewport de 360 x 598. Todo esto es un poco de lío, pero es porque las densidades en píxles de las pantallas ahora son mayores y para representar un píxel usan varios puntos.

Medidas de teléfonos móviles

La gama iPhone tiene medidas que son factibles de recopilar, puesto que los modelos que han aparecido en el mercado son limitados. Sin embargo, la gama Android debe tener decenas de miles de tipos de pantallas distintas, por lo que simplemente podemos hacer un resumen de modelos más típicos.

iPhone 4 y 4S: 320 x 480 iPhone 5 y 5S: 320 x 568 iPhone 6: 375 x 667 iPhone 6+: 414 x 736 Nexus 4: 384 x 598 Nexus 5: 360 x 598 Galaxy S3, S4, S5: 360 x 640 HTC One: 360 x 640

Medidas de tablets

Pasa un poco lo mismo, podemos indicar medidas de viewport para los tablets de Apple, pero es imposible

dar todas las medidas que encontrarás en un Android. Recuerda que, así como los móviles, puedes situar la pantalla en vertical u horizontal, así que es un poco lo mismo que digamos que un iPad tiene unas dimensiones de 768 x 1024 o de 1024 x 768.

Además, por la densidad de píxeles de las pantallas, aunque iPad 1, 2, 3 y 4, incluso el iPad Mini, tengan el mismo viewport, los píxeles de resolución de sus pantallas son diferentes, lo que cambia es el pixel aspect ratio. En este caso concreto, los iPad Mini y los iPad 1 y 2 tienen un device-pixel-ratio de 1 mientras que iPad 3 en adelante tiene un device-pixel-ratio de 2.

iPad todos los modelos, así como iPad Mini: 1024 x 768 Galaxy Tab 2 y 3 (7.0 pulgadas): 600 x 1024 Galaxy Tab 2 y 3 (10.1 pulgadas): 800 x 1280 Nexus 7: 603 x 966 Nexus 10: 800 x 1280 Microsoft Surface W8 RT: 768 x 1366 Microsoft Surface W8 Pro: 720 x 1280

Ordenadores de escritorio

En el caso de los ordenadores de escritorio contamos con las resoluciones de pantalla típicas de toda la vida, aunque también entra en juego recientemente la densidad de píxeles que aumenta en ordenadores modernos, como las pantallas Retina en los Mac.

Pantallas pequeñas (usadas por ejemplo en netbooks): 1024 x 600 Pantallas medianas: 1280 x 720 / 1280 x 800 Pantallas grandes: ancho superior a 1400 píxeles, ejemplo 1400 x 900 o 1600 x 1200.

Relojes

Aquí ya la cosa se pone complicada, puesto que el viewport es muy reducido, por lo que la experiencia de usuario debe ajustarse mucho, pero eso es otro tema.

Apple Watch: 42mm de anchura de pantalla, viewport de 256px (calculado en relación al tamaño de pantalla del iPhone) 360 Moto Watch: 218 x 281

Conclusión

De momento eso es todo. No quiero acabar sin volver a insistir en que esta recopilación no pretende ser completa, ya que las Medidas estándar para breakpoints no existen, pues depende del diseño de tu página.

Hay dos referencias que me parecen muy útiles y que he utilizado como documentación para construir este listado. Por una parte el sitio web Viewport Sizes, que tiene una estupenda referencia de una cantidad enorme de dispositivos. Por otra parte el artículo Media Queries for Standard Devices de CSS-Tricks, que te viene con un código de los media-queries que deberías usar para definir estilos para un dispositivo, lo que nos parece un poco error por los motivos que ya hemos comentado, pero que podrían ser de utilidad a los interesados (solo por capricho) de aplicar estilos específicos para un modelo particular.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *30/06/2015*
Disponible online en *http://desarrolloweb.com/articulos/medidas-estandar-pantallas-media-queries.html*

Todo sobre unidades de medida para el diseño adaptable

CSS dispone de un gran catálogo de unidades de medida que se pueden adaptar a diversas necesidades. En lo que respecta a Responsive Web Design ¿Cuáles son las unidades de medida más interesantes? ¿En qué casos debemos usar cada una?

Unidades de medida en CSS: relativas y absolutas

Discusión sobre las unidades de medida CSS, distinas unidades con sus explicaciones y la diferenciación entre unidades relativas y absolutas.

Las unidades de medida son algo que viene de CSS, tienen que ver con el lenguaje de las hojas de estilo y no especialmente con las técnicas englobadas en "responsive". Por tanto, ya han sido objeto de estudio en otros artículos de DesarrolloWeb.com. Si quieres saber lo básico conviene leer el artículo de unidades de medida CSS http://www.desarrolloweb.com/articulos/185.php si no sabéis a qué nos referimos.

Sin embargo en Responsive Web Design debemos usar unidades de medida con mayor cuidado, si cabe, para sacar lo mejor de cada una. Quizás habéis escuchado que en líneas generales se debe intentar usar unidades de medida relativas y, aunque es cierto en muchas circunstancias, no siempre es correcto ni por tanto lo más adecuado.

Las unidades tienen diversas características que debemos conocer, para usarlas con criterio y escoger las más adecuadas en cada caso. De eso es lo que trata este artículo. Además, también queremos actualizar la información de DesarrolloWeb.com, puesto que en los últimos años han aparecido nuevas unidades, en nuevas especificaciones de CSS, de las que no hemos hablado en artículos.

Nota: En programas en vivo de los que emitimos por hangout todas las semanas sí hemos ido actualizando la información acerca de unidades de medida CSS. Por ejemplo te recomendamos el #designIO: Unidades de medida CSS.

Medidas fijas y medidas relativas

Lo primero que tenemos que saber distinguir es entre unidades relativas y unidades fijas. Por si acaso, vamos a describirlas y constatar cuáles son las unidades CSS más habituales de cada tipo.

Unidades fijas:

Son aquellas que especifican una medida en términos absolutos, sin tener en cuenta el contexto donde se están aplicando. Por ejemplo 300px (300 píxeles) es algo bastante fijo, que tendrá un valor independientemente de dónde se haya definido. 300px son siempre eso, 300px, independientemente de si tu contenedor tiene una anchura de 2000px o de 500px.

> **Nota:** En muchos de los casos cuando decimos 300px en CSS corresponde con 300 puntos en la pantalla, pero no es siempre así pues habría que discutir acerca de pantallas con mayor densidad de píxeles, como las pantallas retina. En determinadas pantallas un px de CSS no tiene siempre que corresponder con un punto en la pantalla, pero no nos vamos a meter todavía en ello.

Las siguientes unidades son unidades fijas:

px: Píxeles in: Pulgadas (1 in es igual a 96px) pt: Puntos (1 pt es igual a 1/72 in) cm: Centímetros mm: Milímetros pc: Picas

Unidades relativas:

Las unidades relativas de CSS son aquellas que tienen en cuenta el contexto donde se encuentran. Son relativas a las dimensiones del contenedor donde se han definido. Por ejemplo %, es una unidad relativa, puesto que 30% de ancho no será lo mismo para un elemento situado dentro de un contenedor de 2000px de anchura o sobre un contenedor de 1000px de anchura.

%: porcentaje em: Altura de la fuente rem: Root-em vw: Viewport width vh: Viewport Height vmin: Viewport menor, entre altura o anchura vmax: Viewport mayor, entre altura o anchura ex: anchura de la fuente para la letra "x" ch: la anchura del carácter "0" (cero)

> **Nota:** Sigue leyendo para encontrar explicaciones detalladas de estas medidas.

Unidades relativas ¿a qué?

Si sabes la respuesta o al menos te has hecho esa pregunta te puedes dar por satisfecho pues es una buena reflexión. Cada unidad puede ser relativa a una cosa distinta, o incluso medidas como el porcentaje pueden ser relativas a varias cosas.

El porcentaje es relativo a la propia medida heredada. Por ejemplo, si estamos definiendo tamaños de

fuentes y decimos que es 150%, entonces esa medida es relativa al propio tamaño de la fuente del elemento que hereda del contenedor (una vez y media el tamaño de la fuente que habría en el elemento antes de aplicarle el estilo). Pero la misma unidad de porcentaje, aplicada a la propiedad width de una caja será relativa al tamaño del contenedor en su anchura.

Por su parte, la unidad em es relativa al tamaño del texto del contenedor donde está aquel elemento donde se le aplica esa unidad. Y siempre es relativo al tamaño del texto, aunque se aplique en una propiedad como width que afecta a la anchura del elemento.

> **Nota:** "em" es una unidad que equivale al tamaño del texto en un contenedor. 1em es el tamaño de la fuente definido en un contenedor. El tamaño del texto predeterminado en un navegador es de 16 píxeles, por tanto y a no ser que lo cambiemos, 1em en el body será equivalente a 16px.

Por ejemplo, si definimos que un texto sea 2em para calcular el tamaño efectivo habrá que saber cuál es el tamaño del texto en el contenedor y multiplicarlo por dos. Igualmente, si estamos definiendo una anchura y le asignamos 10em, para calcular la anchura efectiva en píxeles tendremos que saber cuál es el tamaño de la fuente del contenedor.

Por ejemplo, tenemos un elemento DIV que es hijo directo de BODY. Si no se ha modificado el tamaño del texto del BODY es equivalente a 16px. Si ese DIV tiene un width de 10em, la anchura efectiva de elemento será de 10 x 16 píxeles = 160px.

Lo interesante aquí es que, para el caso de las dimensiones de los elementos, podemos especificar las anchuras o alturas relativas tanto en em como en %. Si usas em serán relativas al tamaño de texto y si usas % serán relativas al tamaño del contenedor, llegando a fórmulas distintas:

Valor % = (dimensión objetivo / dimensión del contenedor) x 100 Valor em = (dimensión objetivo / fuente contenedor)

Dicho con palabras: Si quieres que un elemento tenga 200 píxeles de ancho y las dimensiones del contenedor en un momento dado son de 1000 píxeles, el valor en % será de 200/1000 = 0.2 * 100 = 20%

Si quieres que un elemento tenga 200 píxeles y el tamaño de la fuente en su contenedor es de 16 píxeles, en em tendrías que darle el valor 200 / 16 = 12.5em

Las fórmulas es solo cuestión de aplicarlas, lo importante que deberías quedarte en este punto es que, dependiendo de la unidad, éstas son relativas a unas cosas y a veces a otras. ¿Cuál usar? La que te venga bien. Generalmente para anchuras relativas de elementos te será más fácil distribuir los espacios usando %, pero quieres tener un elemento que tenga altura aproximada del texto, puedes usar em. Por ejemplo, un span que sea un poco más alto que el texto que tiene dentro.

A continuación tienes un cuadro resumen de las unidades relativas, con respecto a la medida a la que son relativas. %: Porcentaje, puede ser relativo frente a varios elementos, si trabajamos con fuentes es relativo a la fuente, pero si lo aplicamos a width es relativo a la anchura del contenedor, por poner dos ejemplos.

em: Relativa al tamaño de la fuente del elemento actual rem: Relativa al tamaño de la fuente del elemento raíz (HTML) vw: Viewport Width, relativa al tamaño del viewport, sería el 1% de la anchura del viewport

vh: Viewport Height, relativa al tamaño del viewport, sería el 1% de la altura del viewport vmin: Relativa al tamaño del viewport, el valor mínimo entre su altura y altura vmax: Relativa al tamaño del viewport, el valor máximo entre su altura y altura ex: Relativa a la fuente definida en el contenedor, tamaño de la letra "x" en la anchura ch: Relativa a la fuente, igual que ex pero con la anchura del caracter 0 (cero)

> **Nota:** Para aclarar vmin y vmax pensemos en un Viewport de ejemplo 320 x 480 (Si lo tenemos en vertical "portrait", sería 480 x 320 si lo tenemos en horizontal "landscape"). Si lo tenemos en Portrait el vw será equivalente a 3.2px y vh sería 4.8px. En cualquiera de los casos, tanto horizontal como vertical vmin sería el menor entre vw y vh que sería equivalente a 3.2px y vmax sería equivalente a 4.8px.

En el siguiente artículo seguiremos hablando de unidades de medida, explicando algunas unidades no tan usadas y valorando la conveniencia de las unidades relativas y absolutas en diversas situaciones.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *10/08/2015*
Disponible online en *http://desarrolloweb.com/articulos/unidades-medida-css-relativas-absolutas.html*

Unidades de medida CSS más adecuadas para el Responsive Web Design

Explicaciones sobre las mejores unidades de medida y discusión sobre cuándo debemos usar unas u otras para sacar lo mejor de las hojas de estilo y el diseño adaptable.

En el anterior capítulo del Manual de Responsive Web Design estuvimos haciendo una primera aproximación a las unidades de medida de CSS y explicando los grandes grupos de unidades, que serían las unidades de medida relativas y absolutas.

Este texto viene a continuar con lo que veníamos hablando, haciendo un especial hincapié en explicar unidades de medida no tan utilizadas en la actualidad, pero que tienen un comportamiento muy adecuado para los sitios adaptables.

Comenzaremos ofreciendo una explicación sobre las unidades"em", "rem", así como "vw" y "vh".

Unidades "em"

En el artículo anterior ya explicamos muchas cosas sobre esta una unidad de medida CSS, así que no nos vamos a repetir. Solo recordar que em corresponde con la medida actual de la fuente en un contenedor. Por ejemplo, si has modificado el tamaño de la fuente en un elemento y le has asignado font-size: 20px, entonces dentro de ese contenedor (y sus hijos si no redefinimos ese tamaño, 1em será igual a 20px.

Además comentamos que el tamaño de fuente predeterminado en el navegador es usualmente de 16px, por lo que, si no lo has alterado en algún lugar de tu CSS, 1em equivale a 16px. Pero ojo, tómalo solo como una referencia, ya que ese tamaño definido por defecto es algo propio del navegador y que podría cambiar entre marcas de navegadores, versiones, modelos, sistemas operativos o configuraciones del usuario.

Unidades "rem"

Ahora que has entendido "em", puedes entender la unidad "rem", que es relativa al tamaño de la fuente como em, pero en este caso tiene en cuenta el tamaño de la fuente del elemento HTML y no la heredada o definida en el elemento donde se encuentre aquello cuyas dimensiones estás definiendo.

El problema de rem es la compatibilidad con navegadores antiguos, IE8 o anteriores, puesto que es una unidad definida en nuevas versiones de CSS. Por ese motivo, si quieres usar rem y asegurar la compatibilidad hacia atrás, deberías definir primero las medidas en cualquier unidad compatible con todos los navegadores, como px o em y luego definir las medidas con rem.

> **Nota:** Como ves el trabajo de usar rem se duplica por tener que especificar las unidades dos veces. Pero esto lo podrías solucionar de una manera práctica y elegante con preprocesadores de CSS, de los que no hablaremos ahora.

La unidad rem es una excelente manera de definir tamaños de fuentes, pues suele ser mucho más fácil de organizar y te evita sorpresas o dudas habituales al usar em, ya que rem siempre tendrá en cuenta una única medida para calcular el tamaño final de la fuente en cualquier lugar.

Unidades vw y vh

Estas también son unas unidades de reciente aparición en el estándar CSS. Su nombre, vw y vh corresponden con las siglas "viewport width" y "viewport height". Podríamos decir que se corresponde con la anchura y la altura de la pantalla del dispositivo, pero no necesariamente corresponde justamente con las dimensiones reales, puesto que a veces el viewport puede simular que el dispositivo tiene unas dimensiones diferentes de los píxeles de la pantalla. Esto se entiende mejor si conoces el concepto de viewport que hemos explicado anteriormente.

Estas unidades son relativas, aunque en este caso lo son a la pantalla del dispositivo. 1vw es la centésima parte de la anchura del viewport del dispositivo, de modo que 100vw sería la anchura total. Del mismo modo 1vh sería la centésima parte de la altura del viewport.

En la práctica es especialmente útil para conseguir que un elemento ocupe toda la altura disponible en la pantalla de un dispositivo. Como sabemos un height: 100% tiene un comportamiento muy errático, no funcionando en la mayoría de los casos para los objetivos deseados. Por ello podemos usar el vh como unidad, como sigue:

```
.lateral{
    width: 200px;
    height: 100vh;
    background-color: #4ff;
}
```

La parte menos interesante es que no funciona en pantallas de ordenadores, por lo que tendrás que aplicar estilos alternativos si quieres que un elemento ocupa toda la anchura de la pantalla.

> **Nota:** Tenemos un artículo completo donde se explican estas unidades Viewport Width y Viewport Height.

Cuáles son mejores unidades CSS, fijas o relativas?

Otra de las preguntas típicas que nos podemos hacer. Entonces, después de todo, ¿Cuáles son las unidades recomendadas, las fijas o las relativas? Pues me temo que no hay una respuesta exacta que sirva en todas las situaciones. Debemos conocer las características de cada una para saber cuál usar.

No obstante, debido al desarrollo responsive y la imposibilidad de conocer de antemano las dimensiones exactas de la pantalla, generalmente usarás unidades de medida relativas. Pero repito que no lo podemos tomar como una regla general.

Cuando definas una anchura de un elemento que forma parte de un layout usarás habitualmente las unidades relativas, para que al agrandar o disminuir el espacio siga adaptándose bien y se distribuyan correctamente las cajas para las dimensiones actuales del navegador. Esto en lo que tiene que ver con la anchura, puesto que alturas relativas no se suelen trabajar, ya que sabemos que la altura se calcula en función del contenido que haya que colocar en los contenedores.

En la definición de tamaños de las fuentes también es muy útil trabajar con unidades relativas, dado que así todos los tamaños del texto se adaptarán al elemento raíz. Experimentarás que es muy útil, llegada a una anchura dada de pantalla, aumentar (o disminuir) todos los tamaños de los textos de la página, para facilitar la lectura. Si has usado siempre fuentes con tamaños relativos sería muy fácil de conseguir, cambiando simplemente el valor del elemento BODY o HTML. En el siguiente artículo vamos a ampliar el uso adecuado de las unidades CSS para fuentes en diseños adaptables.

Pero a veces hay elementos que por necesidad tienen dimensiones fijas, como imágenes que quizás no deseamos que se estiren o se encojan dependiendo de la pantalla, columnas que deben tener siempre unas mismas dimensiones fijas, elementos de interfaz gráfica que no deseamos que alteren su forma, etc.

En general como ves, encontrarás más casos en los que aplicar unidades relativas, aunque no significa que debamos considerarlo como una regla general de obligado cumplimiento, pues habrá muchas ocasiones donde serán necesarias las unidades fijas.

> Este artículo es obra de *Miguel Angel Alvarez*

Fue publicado por primera vez en *31/08/2015*
Disponible online en *http://desarrolloweb.com/articulos/unidades-css-adecuadas-diseno-adaptable.html*

Unidades de medida CSS para fuentes Responsive Web Design

Las mejores unidades de medida de CSS para trabajar con tamaños para tipografías en Responsive Web Design, ventajas, inconvenientes y soluciones a la unidad rem.

Las unidades de CSS no tienen mucho misterio, ya que están documentadas en cientos de sitios web, sin ir más lejos en DesarrolloWeb.com tenemos varios artículos sobre el tema. Pero en la práctica surgen dudas sobre estos asuntos porque, a veces, las personas no tenemos claro cuándo utilizar em o rem, porcentaje, etc. cuando se trata de dar tamaño a los textos.

Sobre este asunto, Daniel Martínez estuvo exponiendo las respuestas a los problemas más típicos a la hora de asignar tamaños a las fuentes, siempre con unidades de medida para las hojas de estilo que se adapten al dispositivo, para el Responsive Web Design.

Nota: este artículo es un extrato parcial del programa completo, emitido en directo que realizamos en DesarrolloWeb.com, #designIO sobre Unidades de medida CSS.

Las mejores unidades de medida para los textos en CSS

Como debemos de saber, cuando trabajamos con diseños adaptables, debemos usar siempre medidas relativas, para que éstas se adapten a las dimensiones de las pantallas donde pueda verse el sitio web. Esto nos obliga a descartar px, cm, pero nos quedan varias otras donde elegir.

Básicamente, la mejor unidad para trabajar textos adaptables en CSS es el rem, "la unidad perfecta". Esta unidad de medida quiere decir "root em", y es aquella que nos remite al tamaño de fuente que tenemos en la raíz. Por ejemplo, si nosotros tomamos el BODY y tiene un *fontsize* de 100% (lo que corresponderá generalmente con unos 16 píxeles), sabemos que siempre rem será equivalente a esos 16 píxeles, lo pongas donde lo pongas.

Por qué rem es más útil que em

Hasta hace poco, estoy seguro de que muchos de los desarrolladores venían utilizando unidades de medida en em, pero ¿cuál era el problema con estos em? pues que el em depende del tamaño de la fuente del contenedor, o del tamaño de fuente declarado en alguno de los contenedores padre de éste.

La unidad em es relativa al padre, que si tiene a su vez otra unidad en em, se hace relativa al padre y así, en la jerarquía de elementos hasta llegar al elemento raíz, el BODY. Esto, que puede resultar fácil de entender, en la práctica a veces nos puede dar algún quebradero de cabeza al calcular qué medida es exactamente un em en un contenedor, porque dicho tamaño depende de una innumerable cantidad de contenedores que pueden estar anidados, hasta llegar al contenedor donde estamos indicando las medidas. De este modo, para cambiar el tamaño de la fuente de ese elemento, tenemos que tener en cuenta muchas otras medidas que pueda haberse dado a cualquiera de los padres en la página.

Calculando un valor en em

Veamos esto en números para darnos cuenta de cómo se debe calcular el tamaño de una fuente usando em, para hacerla corresponder al tamaño de píxeles deseado.

Imaginemos el tamaño base de la fuente es de 16 píxeles. El tamaño de un H1 imaginemos que lo queremos de 24 píxeles, que sería 1.5em. ¿Por qué? pues tenemos que aplicar la regla "target/context" el target es la medida que le queremos dar al H1, 24px y el contexto es la medida del contenedor donde estamos 16px. De modo que la medida que le tenemos que poner al H1 sería de 24/16 = 1.5em.

Pero ahora imaginemos que dentro del H1 tenemos un *link* y queremos darle un tamaño de letra menor, digamos 12px. Ahora ya no podemos dividir 12/16 para saber el tamaño de esa medida en em, ahora tendríamos que tener en cuenta el tamaño del H1 que habíamos quedado que era 1.5em. Si no sabemos que eso equivale en este caso a 24px, primero deberíamos de calcular el tamaño del H1 para llegar a los 24px que tenía. Target sería 12px y context 24px, de modo que 12/24 = 0.5em el tamaño en em que deberíamos poner al enlace.

Pero la cosa se puede complicar todo lo que queramos. Ahora podemos imaginar que dentro del link hay un SPAN, al que queremos darle otro tamaño de fuente distinto. Si queremos acertar con el tamaño deseado tendremos que tener en cuenta el tamaño del BODY, el definido luego para el H1, el definido para el enlace y finalmente calcular el tamaño del SPAN.

Pensando en otros dispositivos

Pensemos además en diferentes dispositivos. El tamaño de una fuente en un móvil podría ser si se desea un poquito menor que el tamaño de fuente en un Ipad.

> **Nota:** Aquí podríamos pensar que al ser el móvil de menor tamaño que el Ipad, la fuente ya de por si va a verse menor, pero debemos tener en cuenta que un móvil siempre lo tendremos más pegado a la cara, con lo que realmente admite un tamaño de fuente menor.

Si queremos entonces reducir la fuente podríamos asignar al BODY, por medio de "mediaqueries" para pantallas pequeñas, un tamaño de 80% en el tamaño de la fuente. Como todas las fuentes están definidas con medidas relativas, encontraremos que se reducen en toda la página, pero en la práctica observaremos

que se nos empieza a descuadrar, que los tamaños no casan.

Estos son los típicos problemas que evitamos con rem.

Cómo funcionan las unidades Rem

Rem no funciona de manera relativa a su contenedor, sino de manera relativa al tamaño definido en la raíz. De esa forma, cuando quiero calcular una medida de un elemento, no tengo que pensar en el tamaño de su padre o en el tamaño del padre del padre.

La regla del "target/context" sigue valiendo perfectamente, pero puedo hacer la cuenta siempre con respecto al tamaño definido en la raíz.

Si en el elemento BODY teníamos 16px de tamaño y quiero 24px en el H1, tengo que calcular 24/16 = 1.5rem. Si luego en el enlace que había dentro del H1 quiero que sea 12px, tengo que calcular como context 16px (que era el equivalente al 100% definido como tamaño de la raíz) y no el tamaño del H1. O sea, en este caso sería 12/16 = 0.75rem.

Compatibilidad con las unidades de medida Rem y fallback

Infelizmente, existe un problema con las unidades de medida especificadas en rem y no es otro que Internet Explorer 8 y por supuesto, las versiones anteriores a éste. Los IE antiguos no son compatibles con las unidades rem.

La solución es sencilla y se realiza por lo que se denomina fallback, que se basa en especificar código alternativo en caso de que no funcione el original. En este caso el fallback es tan sencillo como especificar las medidas en dos unidades.

```
fontsize: 16px;

fontsize: 1rem;
```

Los navegadores antiguos que no entienden rem se quedarán con fontsize 16px (porque ignoran la segunda línea, ya que no son compatibles con esa unidad). Los que entienden rem simplemente leen la segunda línea y sobreescriben el valor especificado en la primera.

> **Nota:** ¿Por qué nos podemos permitir escribir en píxeles esa medida fallback? porque nos resulta cómodo utilizarlos, ya que no tenemos que hacer ningún cálculo. Como esa medida solo la van a interpretar en los IE 8 o inferior, y son ordenadores de escritorio, no nos molesta.

Ciertamente nos da un poco más de trabajo especificar por partida doble todas las medidas, pero aquí encontramos una de las ventajas de usar los llamados preprocesadores CSS. Éstos tienen la posibilidad de escribir un mixin por medio del cual podemos indicar en nuestros CSS, siempre medidas en píxeles y luego que el propio preprocesador sea el que la convierta a rem, mediante la regla target/context y escriba las dos unidades de medida, tanto en rem para los navegadores modernos y píxeles como fallback para los Internet

Explorer antiguos.

Conclusión

De momento, con estas informaciónes esperamos que tengas suficiente para empezar a experimentar con los rem y asignar de manera más sencilla los tamaños de las fuentes de tus sitios web.

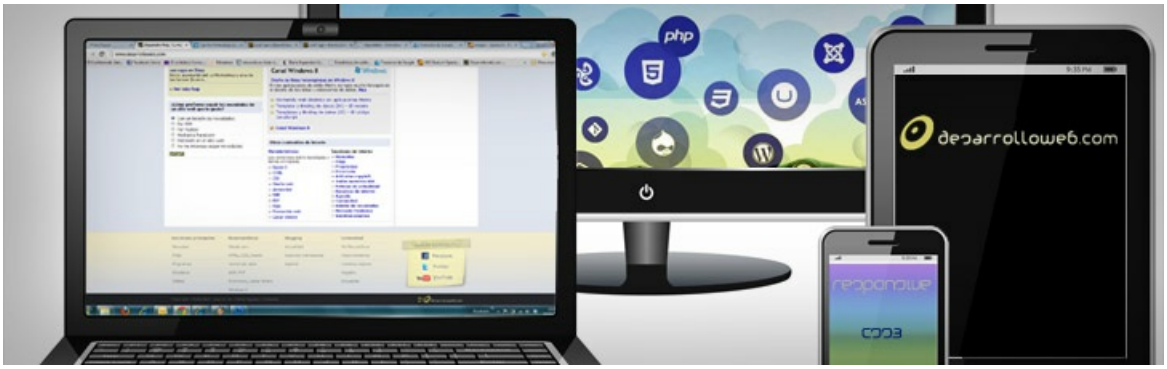
Si te ha interesado este artículo debes saber que simplemente te hemos resumido los primeros 14 minutos del programa. Estuvimos hablando sobre muchas otras medidas y unidades en CSS, no solamente para las fuentes, sino para otros tipos de atributos, como altura, anchura, etc. Todo eso lo encontrarás en el #designIO sobre unidades de medida CSS.

Este artículo es obra de *Daniel Martínez*
Fue publicado por primera vez en *10/07/2013*
Disponible online en *http://desarrolloweb.com/articulos/unidades-medida-css-responsive.html*

Unidades CSS viewportwidth y viewportheight

Nuevas unidades de CSS3: viewportwidth y viewportheight. Cómo podemos hacer un útil uso de ellas al aplicarlas a tamaños de las tipografías.

En este artículo vamos a hablaros de dos unidades nuevas de CSS, que tienen especial utilidad en el diseño adaptable (conocido como "responsive") y que vienen a solucionar algunas necesidades a la hora de definir tamaños en dispositivos, sobre todo útiles para las fuentes (o tipografías). Se trata de las unidades CSS3 "viewportwidth" y "viewportheight", que son relativas a la anchura y altura del dispositivo, respectivamente.



Estas unidades son una opción interesante para definir alturas y anchuras en dispositivos. Las unidades son abrevidadas con las siglas "vw" y "vh". Su medida es equivalente a un centésimo de la anchura o altura del dispositivo donde se está visualizando la web. Por ejemplo, si deseamos asignarle a un elemento una altura igual al tamaño completo del dispositivo, le aplicaremos el valor 100vp.

Por expresarlo mediante porcentajes, 1vw = 1% de la anchura del dispositivo. Por su parte, 1vh = 1% de la altura del dispositivo.

Nota: Para entender mejor estas medidas, si es que no lo sabes ya, te interesa saber lo que es el "Viewport". En resumen, viewport es igual a las dimensiones de la pantalla de tu móvil o tableta, mientras que en ordenadores de escritorio serían las dimensiones de la ventana del navegador.

La pega del viewportheight, bastante importante, es que solo funciona en dispositivos móviles y no en ordenadores de escritorio. Pero aun así merece la pena conocer y aplicar cuando realmente le podamos sacar partido, siempre en teléfonos y tabletas, porque en *desktop* tiene unos comportamientos un poco erráticos.

Uso de viewportheight para definir alturas de una manera cómoda

Las alturas en CSS siempre son un quebradero de cabeza cuando se quiere usar medidas relativas, con unidades como "%" de CSS. Seguramente que si tenemos algo de experiencia sobre maquetación web se sabe de lo que estamos hablando. Para tener las alturas "controladas" con unidades relativas de CSS tenemos que darle dimensiones de altura a todos los contenedores. O sea, definir las dimensiones de height de todos los elementos de la jerarquía del documento hasta llegar al elemento que realmente queramos definir su altura con un valor porcentual. La opción para alturas es el viewportheight

El correspondiente "compañero" de Viewportheight para las anchuras, viewportwidth, no es tan crucial para definir las dimensiones width, debido a que los comportamientos de unidades CSS para las anchuras no nos dan ningún problema. Pero no conviene perderlo de vista, especialmente pensando en las tipografías.

Medidas "viewport" para tipografías

Estas medidas, tanto viewportheight como viewportwidth, tienen un comportamiento especial que merece la pena mencionar. ¡Se puede usar en la tipografía! Esto quiere decir que podemos asignarle medidas en viewportwidth a un texto y éste siempre te medirá lo mismo en todas las pantallas, relativamente a las dimensiones de la pantalla que tengamos. Para que nos enteremos, si definimos el tamaño de una fuente con viewportwidth, y si en un dispositivo vemos que ocupa un 50% de la anchura de su pantalla, podemos tener certeza de que, en todos los dispositivos ocupará más o menos ese mismo porcentaje de anchura de la pantalla, aunque estemos trabajando con una resolución diferente.

Lo asombroso de todo esto es que el tamaño de la letra es "responsive", o en castellano "adaptable", sin necesidad de utilizar ningún *script* para conseguirlo o de otras técnicas como *mediaqueries* que se podrían aplicar con CSS para intentar conseguir ese efecto de fluidez.

Os animamos a probar estas medidas de CSS para sacar vuestras conclusiones. Su uso no difiere de cualquier otro que puedas estar trabajando en tus hojas de estilo.

```
body{
fontsize: 2.5vw
}
h1{
fontsize: 4.8vh
}
```

Es una maravilla que podemos usar y al menos en dispositivos nos dará la tranquilidad de que esas fuentes se van a ver igual de grandes, relativamente al tamaño de la pantalla donde se esté mostrando la página.

Nota: Esta información la hemos extractado del evento transmitido en directo Unidades de medida CSS #designIO, realizado en DesarrolloWeb.com con la participación de Daniel Martínez @Wakkos y Daniel Ruiz @MrViSiOn2, que nos ofrecieron mucha información sobre unidades de medida en hojas de estilo en cascada y con la mente en el diseño adaptable. Si te interesan las medidas CSS más adecuadas para diseños adaptables te recomendamos leer también el artículo Unidades de medida CSS para fuentes Responsive Web Design.

Este artículo es obra de *Daniel Martínez*
Fue publicado por primera vez en *16/09/2013*
Disponible online en *http://desarrolloweb.com/articulos/unidades-css-viewportwidth-viewportheight.html*

Practicando con CSS para Responsive Web Design

Por fin aterrizamos al código, comenzando a ver técnicas, estilos y recursos prácticos para poder componer diseños adaptables. Pero ojo, todo lo que ya conoces de las CSS está disponible y es aplicable en principio al diseño responsive, por lo que nos limitaremos a ver las cosas que representan una novedad en los diseños adaptables. Para complementar esta información te sugerimos también que busques en otros manuales de CSS en DesarrolloWeb.com.

Estilos CSS básicos para Responsive Web Design

Estilos, atributos y valores de CSS que resultan esenciales para el Responsive Web Design y que existen desde las versiones antiguas del estándar de las hojas de estilo.

Cuando pensamos en CSS para Responsive Web Design nos vienen a la cabeza multitud de herramientas disponibles en el lenguaje. Quizás aquello que más resulta representativo sean las Media Queries y es cierto que éstas son las que nos permiten hacer la mayor parte de la "magia" para conseguir sitios web que se adaptan a todo tipo de pantallas. Sin embargo, el mundo de las CSS es enorme y debemos aplicar muchas otras técnicas y estilos para conseguir producir sitios responsivos. En este artículo vamos a repasar unas cuantas.

No pretendemos ofrecer técnicas en orden de importancia o en el orden que las debes de conocer, tampoco explicar acerca de flujos de trabajo, pues de todo eso ya se habla en detalle en otra serie de artículos del Manual de Responsive. Más bien se trata de exponer una lista de propiedades de estilo, a modo de cajón desastre.

Como casi siempre que hablamos de diseño responsive, los consejos que encontrarás en este artículo se pueden aplicar, y se deben aplicar, para un mejor diseño web en general, así que las debes tener presentes incluso si no estás haciendo sitios adaptables.

Variaciones del modelo de caja entre los navegadores

Una de las primeras cosas que debemos entender, y aprender, cuando nos incursionamos en el mundo de las CSS es el modelo de caja. Supongo que si estás leyendo este texto algunas nociones debes de tener. Lo que no sé si sabes son las diferencias que existen entre los navegadores, principalmente las versiones antiguas de Internet Explorer.

El atributo box-sizing, aparecido en la especificación CSS3, nos permite indicar cómo deseamos que se interprete el modelo de caja, evitando que cada navegador lo entienda a su modo.

En concreto nos sirve para alterar el modo en el que los navegadores calculan las dimensiones de una caja (un elemento de la página en términos generales), es decir, su altura y anchura. El problema es que unos navegadores para hacer ese cálculo tienen en cuenta el padding y otros no, así como las dimensiones del borde. Si no definimos un box-sizing de manera global para todo el sitio web, corremos un serio riesgo que las páginas del sitio se vean de manera diferente en cada navegador, lo que generalmente no es deseable.

Obviamente, estas diferencias son muy importantes a la hora de maquetar. Básicamente porque si no las tienes en cuenta puede que tengas que rehacer mucho trabajo cuando pruebas tu página en otros navegadores. Esto es porque generalmente vamos maquetando y probando la página en un navegador y luego, cuando probamos la página en otro navegador que no entiende el modelo de caja de la misma manera, vemos que muchas cosas quedan fuera del lugar donde les corresponde.

No es nuestro objetivo explicar detenidamente las distintas alternativas de box-sizing, pero sí te indicamos cuál es la recomendación:

```
box-sizing: border-box
```

Ese atributo lo tendrás que especificar para todos los elementos de la página, así que puedes usar el selector * para seleccionarlos a todos.

Las posibilidades de display table para maquetación

Ese titular no significa que estemos apoyando la maquetación con tablas, ni mucho menos. Una cosa es el display:table de CSS y otra bien distinta las tablas de HTML realizadas con la etiqueta TABLE. La recomendación sigue siendo la misma, usar TABLE para mostrar información tabulada, sin embargo display: table nos sirve para facilitar las posibilidades de las tablas tradicionales en el uso de HTML semánticamente correcto.

Como esto ya lo hemos explicado en otras ocasiones no tiene sentido que lo volvamos a repetir aquí, simplemente recomendar la lectura del artículo sobre display table.

Maquetación Flexbox

Otra de las cosas nuevas que nos trae CSS y que sirven para maquetar de manera mucho más versátil, rápida y evitando muchos de los problemas habituales del posicionamiento de elementos, es usar las técnicas Flexbox.

Flexbox está con nosotros desde hace tiempo, el problema es que navegadores muy antiguos no lo soportan todavía. En concreto y como suele ocurrir, Internet Explorer es el que da los problemas, puesto que sólo se

encuentra disponible Flexbox a partir de la versión 10. Aunque dicho sea de paso, IE10 todavía te obliga a trabajar con prefijos propietarios "ms-" y tiene un soporte parcial. El estándar de maquetación Flexbox lo disfrutarás desde Internet Explorer 11 y de manera global en todos los otros navegadores del mercado. Si ese no representa un problema para ti, lo mejor para maquetar es aprovechar Flexbox porque te ofrece unas posibilidades increíbles hasta el momento.

Para saber más de Flexbox te recomiendo verte una clase en directo en la que explicamos las posibilidades de este estándar.

Atributos min-width y max-width / min-height y max-height

Estos atributos están disponibles desde hace mucho tiempo en las CSS así que deberías conocerlos. Son especialmente útiles en los diseños responsive porque los contenedores se estiran y se encogen, para adaptarse a las dimensiones de las ventanas o pantallas. Generalmente es el comportamiento deseado, lo que ocurre es que en muchos casos no deseas que ese encoger o agrandar llegue hasta ciertos límites.

Estás diseñando una web y muchas veces no deseas que la anchura llegue al límite posible de un monitor de alta resolución, por ejemplo 2.500 píxeles. No digo que no se deba hacer, es una decisión de diseño.

```
.dimensiones-maximas{
    max-width: 1600px;
}
```

A veces se trata de una columna que no debe medir más que unas dimensiones de anchura, quieres que al hacer grande la ventana se estire, pero que no llegue a pasar de un valor determinado. O una imagen que se adapta al ancho de un contenedor pero no quieres que llegue a ser tan grande que se sobrepase su resolución.

```
.img-banner-aside{
    width: 100%;
    max-width: 300px;
}
```

Con alturas pasa un poco lo mismo, aunque en este caso no se agrandan debido a cambios en las dimensiones de la pantalla o ventana, sino debido al contenido que tienen. A veces tienes un listado de artículos, por ejemplo. El propio navegador hará que las cajas donde los coloques sean de la altura necesaria para que quepa la descripción del artículo, pero unos tienen más texto que otros y por tanto puede ocurrir que las alturas queden descompensadas. Muchas veces no significará un problema, sobre todo cuando están uno debajo del otro, pero si queda uno al lado del otro y las cajas tienen un color de fondo puede que el efecto sea más feo (Ese problema se soluciona con Flexbox o con display table). Incluso, aunque estén uno debajo del otro, a veces si unos tienen texto muy pequeño, la caja queda con tan poca altura que se ve ridícula en comparación con otras y por ello quieres que tenga una altura mínima.

```
.article-home{
    background-color: #ddd;
    min-height: 225px;
```

```
}
```

En definitiva, en todos esos casos y muchos más que te puedas encontrar, conviene tener a mano los atributos max-width y min-width, así como max-height y min-height.

Conclusión

En este artículo hemos comentado diversos criterios de CSS y herramientas del lenguaje que se usan mucho para los diseños responsive. Por supuesto no son los únicos! solo unos cuantos que debes usar sí o sí, y sin llegar a la complejidad y potencia de las Media Queries que te solucionan los problemas de una manera diferente.

En el próximo artículo tenemos que hablarte de otro elemento fundamental cuando trabajas con diseños responsive. No es un CSS, sino una etiqueta META que es necesaria para que tu página se adapte completamente a la pantalla de un móvil: el Viewport.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *13/11/2015*
Disponible online en *http://desarrolloweb.com/articulos/estilos-css-rwd.html*

Etiqueta meta Viewport

El Viewport es una de las etiquetas más representativas de la web móvil, que nos permite configurar cómo debe interpretar una página el navegador web para móviles.

Inicialmente creada por Apple para definir diversas directrices sobre cómo el iPhone debe renderizar un documento web, el viewport es una etiqueta META que se ha convertido en un estándar en el momento actual. La mayoría de dispositivos móviles la soportan en la mayor gama de sistemas operativos, lo que la convierte en un integrante imprescindible para cualquier página que esté pensada para verse también en dispositivos en movilidad.

El viewport es uno de esos conceptos que cuesta más de explicar que de entender, o por lo menos a mí me lo parece. Quizás tampoco sea el interlocutor más adecuado para hablar de esta etiqueta META, pero alguien la tiene que tratar con el detalle que se merece en el Manual para el diseño web en móviles. Así que ¡vamos manos a la obra!

Básicamente, sirve para definir qué área de pantalla está disponible al renderizar un documento, cuál es nivel de escalado que puede realizar el usuario, así como si el navegador debe mostrarla con algún *zoom* inicial. Todo ello se indica a través de varios parámetros en la propia etiqueta META, como veremos en el presente artículo.

Entender el viewport

Comencemos por entender qué es el viewport. Y como veremos, se trata de un concepto bien sencillo, pues corresponde con el área que está disponible en la pantalla del navegador.

Aunque el viewport es algo propio de navegadores para móviles y cobra sentido justamente cuando estemos pensando en estos dispositivos con pantallas reducidas, podemos entenderlo primero en el contexto de un navegador de escritorio.

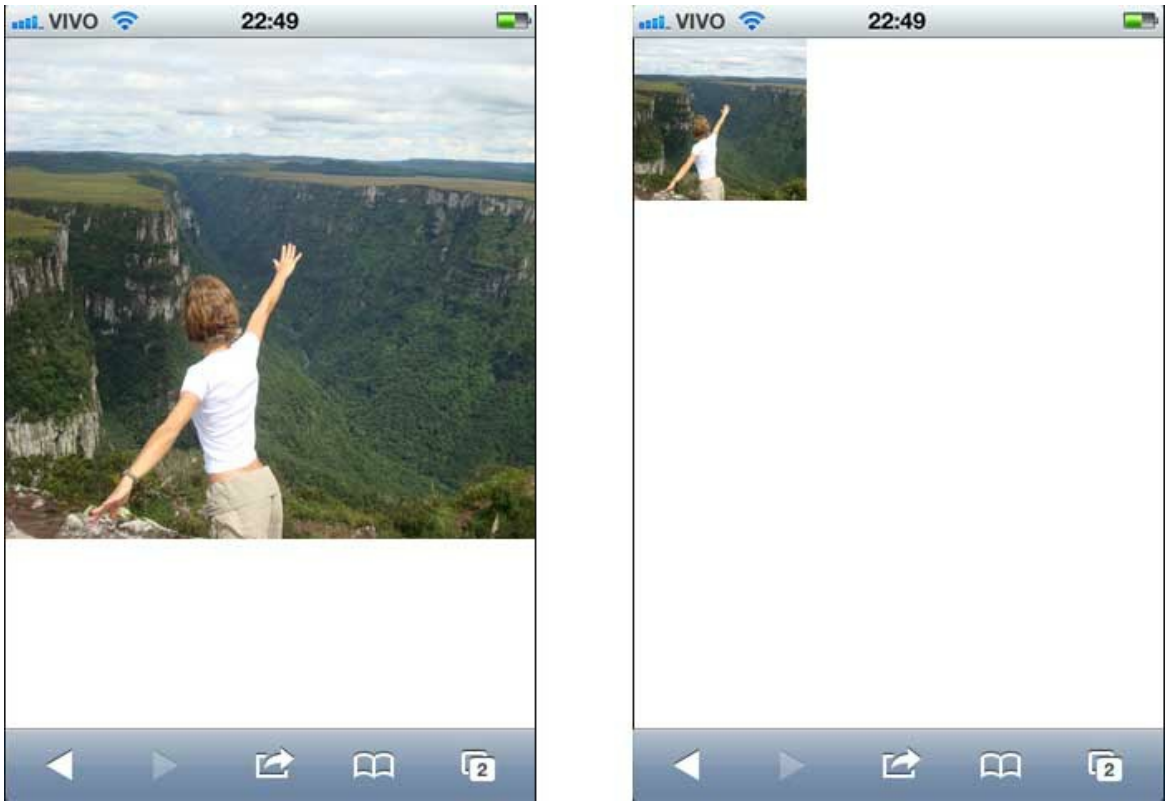
El viewport en un navegador en cualquier ordenador con sistemas tradicionales es igual al área de la ventana, o mejor dicho, al área disponible para renderizar el documento web (o sea, le restamos toda la interfaz del navegador, como botones, barra de direcciones, barra de menús, barras de desplazamiento, etc.) Dicho de otro modo, es el área útil donde se mostrará la página web.

¿Sencillo? Espero que sí. Pero ahora pensemos en móvil. Quizás sepas que cuando se ven las páginas en una dispositivo a menudo se reducen los contenidos, para conseguir que se ajusten al reducido espacio de la ventana del navegador. Es decir, para mostrar toda la página en el espacio disponible de la pantalla del dispositivo, se hace un escalado de la web, de modo que se ve todo en pequeñito.

Esta es la manera en la que se vería el sitio actual de DesarrolloWeb.com que está optimizado para ordenadores de escritorio y no tiene definida ninguna etiqueta viewport.

Bien, pues el viewport cuando estamos hablando de dispositivos móviles, no corresponde al tamaño real de la pantalla en píxeles, sino al espacio que la pantalla está emulando que tiene. Por ejemplo, en un iPhone, aunque la pantalla en vertical tiene unas dimensiones de 320 píxeles, en realidad el dispositivo está emulando tener 980 píxeles. Esto hace que ciertas páginas web (optimizadas para navegadores de escritorio) quepan en una pantalla de 320 píxeles, porque en realidad el Safari para iOS está emulando tener un espacio de 980 píxeles.

Pues bien, el viewport en estos casos es el espacio que el dispositivo emula tener, no la resolución real en píxeles que tiene la pantalla. Lo interesante en este caso es que los desarrolladores somos capaces de alterar el viewport que viene configurado en el navegador, algo que resulta totalmente necesario si queremos que nuestra página se vea correctamente en dispositivos de movilidad.



Esta imagen tiene la misma foto que se muestra en la pantalla de un iPhone. Supongamos que la foto mide 320 píxeles de ancho. En la parte de la derecha tendríamos la foto a tamaño real, que es como se vería si tuviéramos un viewport configurado a 320 píxeles de ancho. Pero al verla en un iPhone con un viewport configurado a 980 píxeles de ancho, la imagen se verá bastante más pequeña.

Configurar la etiqueta Viewport

Cuando Safari de iOS renderiza un documento web, hace un escalado de los contenidos para que las páginas diseñadas para sistemas de escritorio se vean más o menos bien en un teléfono móvil. Como pudiste apreciar en la primera imagen de este artículo, al verse el sitio web de DesarrolloWeb.com, los contenidos aparecían muy pequeñitos y ello nos obliga a hacer zoom para poder leerlos. Sin embargo, nosotros podemos configurar el viewport para decirle a Safari, o cualquier otro navegador para móviles, qué debe hacer en este sentido.

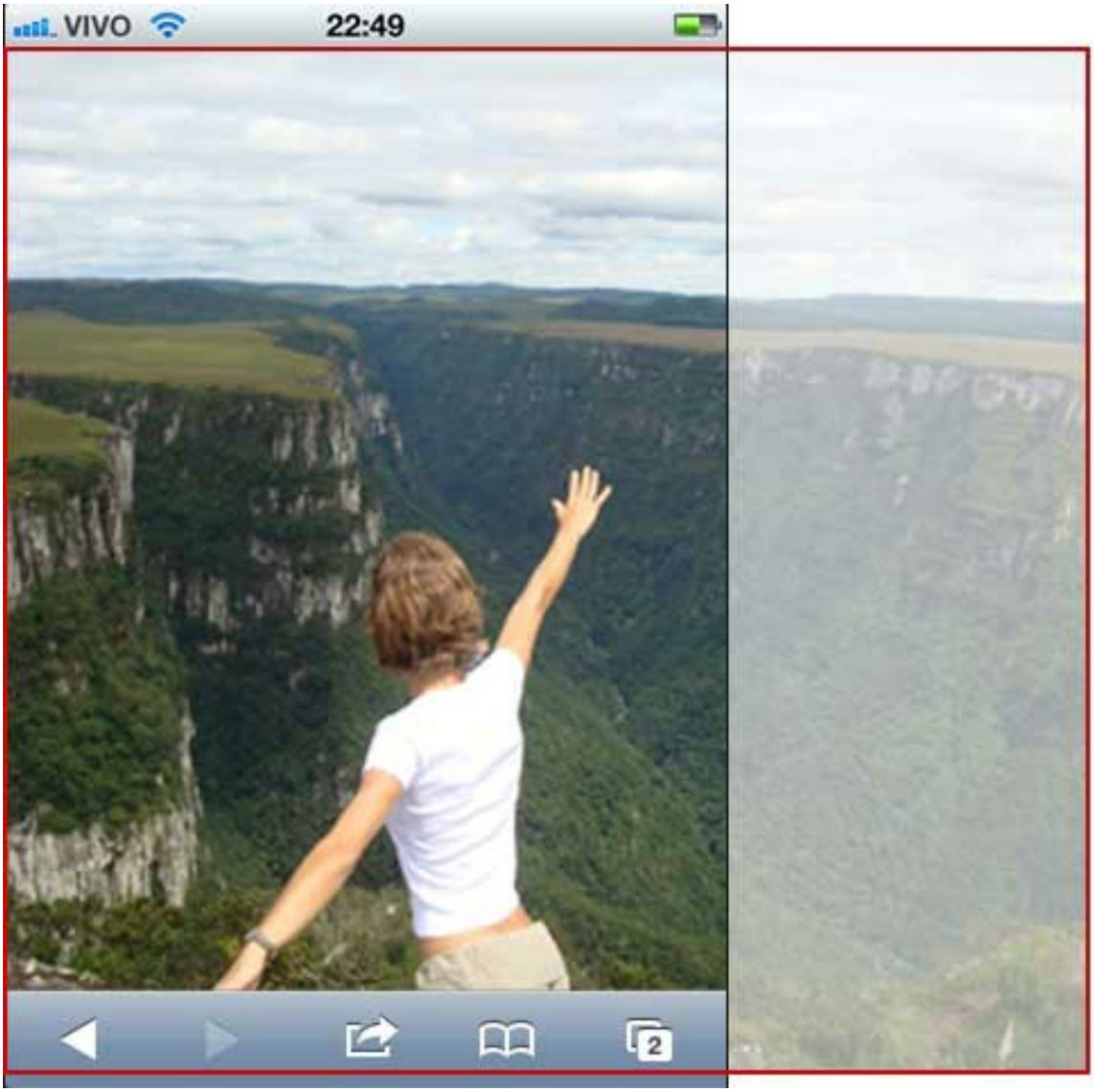
Nota: Insistimos en que, cuando hablamos de Safari, en realidad nos referimos a cualquier navegador para móviles, ya sean en dispositivos iOS, Android, Blackberry, etc., ya que la etiqueta viewport es soportada actualmente por la mayor gama de sistemas para móviles. En realidad, decimos Safari solamente porque fue el primero en utilizarla.

Es altamente recomendable que se altere la etiqueta viewport para conseguir que tu navegador se comporte como tú deseas, especialmente en el caso de las páginas que estamos diseñando para verse correctamente en pantallas pequeñas. Para ello disponemos de los siguientes parámetros en la etiqueta META.

- Width: anchura virtual (emulada) de la pantalla o anchura del viewport.
- Height: altura virtual de la pantalla o anchura del viewport.
- Initial-scale: la escala inicial del documento.
- Minimum-scale: la escala mínima que se puede poner en el documento.
- Maximum-scale: la escala máxima configurable en el documento.
- User-scalable: si se permite o no al usuario hacer zoom.

Como puedes ver, en la META viewport no se indica simplemente las dimensiones de la pantalla emulada, sino también el nivel de zoom que se puede estar configurando inicialmente y el nivel de zoom que se permitiría tener.

Nota: recordar que el usuario puede hacer zoom en la página web, con el gesto típico en la pantalla táctil. Al hacer zoom, realmente no estaría cambiando el viewport, sino la escala con la que se visualiza el documento.



En esta tercera imagen tenemos un viewport de 320 píxeles, sin embargo hemos definido un "initial-scale" de 1.5, por eso la imagen que medía realmente 320 píxeles de ancho no cabe en la pantalla del iPhone.

Un ejemplo de etiqueta viewport sería el siguiente:

```
<meta name="viewport" content="user-scalable=no, width=device-width, initial-scale=1">
```

La anchura y la altura del viewport deben quedar más o menos claros, pues se refieren simplemente a las dimensiones fijas del viewport inicial. Sin embargo, esas dos medidas no se suelen indicar, siendo más habitual definir únicamente la anchura con el valor "device-width", que es una medida que hace referencia a la anchura de la pantalla del dispositivo.

Así que con width=device-width conseguimos que el viewport sea igual a la anchura real de la pantalla del dispositivo, de modo que no se tratará de emular una pantalla mayor de lo que realmente es y veremos los píxeles reales.

Con initial-scale=1 conseguimos que no se haga zoom sobre el documento. Es bien simple, el contenido de la web no se transformará, ni se agrandará, ni se hará menor.

Con user-scalable=no conseguimos que el usuario no pueda hacer zoom en la página, con lo que siempre se mantendrán las medidas que nosotros hemos definido al construir la web.

Nota: desactivar el zoom con user-scalable=no nos facilita tener siempre la escala a como nosotros hayamos definido en la etiqueta META viewport, lo que puede simplificarnos la vida a la hora de definir cómo se debe ver una web. Sin embargo, estaremos limitando la posibilidad de que el usuario haga zoom puntualmente para agrandar o empequeñecer alguna cosa. Por todo ello, cabe sopesar bien qué es lo que queremos permitir y si realmente definiendo un maximum-scale y minimum-scale, estamos acotando bien el uso de nuestra web.

Conclusión

El viewport requiere un estudio pormenorizado y algo de práctica para entender bien sus posibilidades y la manera de configurarlo, de modo que se optimice la forma con el que nuestra web se debe de ver en los dispositivos móviles. Como se ha podido comprobar, hay muchas y diversas posibilidades para explorar.

Esperamos que en este artículo te hayamos sacado de dudas, sin embargo posiblemente podríamos estar hablando del viewport durante un largo rato para resolver todas las necesidades de los desarrolladores y las dudas de aquellas personas que están diseñando webs optimizadas para móviles o adaptables a todos los dispositivos (responsive design).

Para mayores referencias, se puede consultar la página web "Configuring Viewport" de la documentación de Apple iOS para desarrolladores.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *19/07/2012*
Disponible online en *http://desarrolloweb.com/articulos/etiqueta-meta-viewport.html*

Imágenes responsive con la etiqueta Picture

Cómo usar la etiqueta Picture para crear imágenes responsive, un nuevo elemento de HTML5 que permite que las imágenes y su resolución se adapten a todo tipo de pantallas.

Uno de los principales componentes con los que tenemos que lidiar en toda web son las imágenes y su tratamiento para producir sitios adaptables es clave. Para adaptar una imagen muchas veces es suficiente con asignarle con CSS una anchura relativa, de modo que si su contenedor aumenta de tamaño, la imagen también aumentará.

La problemática sobre todo la encontramos con imágenes pesadas, como fotografías, donde la elección de su tamaño y calidad dependerá de la pantalla donde se visualicen. Si pensamos que esa imagen debe mostrarse en pantallas grandes, tendremos que generar un archivo de gran tamaño. Pero ese archivo de proporciones grandes representa un desperdicio de transferencia si pensamos que esa imagen se va a ver en un móvil con una pantalla realmente pequeña.

Lo ideal es poder distribuir imágenes acorde con el tamaño de la pantalla y no solamente sus pulgadas, sino a veces también la densidad de píxeles importa. En el pasado aparecieron algunos plugin Javascript especiales para hacer imágenes responsive, también CSS hacks o incluso servicios web, que nos permitían discriminar por el tamaño de la pantalla, o velocidad, entregando al navegador el archivo con el fichero de imagen más adecuado dependiendo de los diferentes escenarios. Soluciones propietarias que acabarán de ser sustituidas por la nueva etiqueta PICTURE.

El objetivo, en resumidas cuentas es el siguiente:

- Poder indicar una imagen y varios archivos de alternativa para distintos escenarios: pantallas pequeñas, medianas o grandes, velocidades de conexión, etc.
- Que el navegador sólo descargue una de las imágenes posibles y no todas. O sea, si una imagen tiene tres alternativas de tamaños, se le entregue al navegador solamente una de ellas, ahorrando la descarga de los otros tamaños alternativos que no necesita.

Elemento PICTURE

El elemento PICTURE en sí no representa contenido alguno, sirve sólo como un contenedor para escribir varias imágenes y que sea el propio navegador el que elija la más apropiada en cada caso.

Para indicar las imágenes dentro de PICTURE se indicarán varias etiquetas SOURCE. Se parece algo a la etiqueta VIDEO o AUDIO, en el que se pueden indicar varios archivos diferentes, de modo que el navegador localice aquel que consiga reproducir con su conjunto de codecs. Pero en este caso en vez de vídeos se indicarán archivos gráficos.

Además, a modo de fallback, dentro de la etiqueta PICTURE se encontrará una imagen corriente, con la etiqueta IMG, que será representada por todo navegador que todavía no implemente el tag HTML5 PICTURE.

Para que cada navegador sepa cuál es el archivo más interesante según sus condiciones a las etiquetas SOURCE se les debe escribir un atributo "media", indicando una mediaquery CSS que debería cumplirse para usar ese archivo fuente.

Es muy sencillo. Lo veremos mejor a la vista del siguiente ejemplo:

```
<picture>
    <source media="(min-width: 900px)" srcset="grande.png">
    <source media="(min-width: 550px)" srcset="media.png">
    <img src="archivo-pequeno.jpg">
```

```
</picture>
```

Este código funcionará en todos los navegadores. Los que estén familiarizados con el tag PICTURE mostrarán la imagen adecuada en cada caso. Los que no estén familiarizados simplemente mostrarán la imagen que se haya indicado en IMG.

SOURCE element

Ahora veamos el elemento SOURCE, usado para indicar cada una de las varias imágenes.

Básicamente nos permite indicar varios archivos gráficos. El navegador irá leyéndolos uno a uno, desde el primero hasta el último, en orden de colocación, hasta que encuentre uno que sea posible usar. Usará el primero que sea posible, descartando los demás que pueda haber por debajo.

Incluye los siguientes atributos.

srcset: Contiene el archivo de imagen que debe visualizarse. Es un campo requerido.

```
<source media="(max-width: 300px)" srcset="archivo-hasta-300px.jpg">
```

Pero además, podríamos indicar varias alternativas de imágenes, separadas por comas, para indicar versiones de la misma para distintas densidades de píxeles. Esto es muy interesante para los móviles modernos, que tienen pantallas con viewport pequeños pero que realmente tienen muchos más píxeles de los que aparentan. También para los monitores "retina" u otras marcas que tienen densidades de píxeles distintas. Para marcar la densidad de píxeles indicamos algo como "2x" o "3x" dependiendo si duplican o triplican la densidad estándar. (Si no se indica nada se entiende que ese archivo es para la densidad normal de píxeles.

```
<source media="(min-width: 900px)" srcset="foto.jpg, foto-hd.jpg x2"
```

En el caso anterior se distribuirá la imagen foto.jpg en pantallas con densidad de píxeles común y foto-hd.jpg para las densidades de píxeles mayores.

Media: Este atributo soporta cualquier media query que podrías usar en un selector del tipo @media. Se puede indicar solo en caso que se necesite y si no se indica nada, afectará a todos los tipos de pantallas.

> **Nota:** Como son mediaqueries CSS podríamos indicar más conceptos, como que el modo de disposición de la pantalla, portrait o landscape. Así mismo, podríamos usar cualquier tipo de medida en las reglas de Media querires, además de los píxeles (px). En un artículo anterior podríamos acceder a explicaciones detalladas de Mediaqueries.

Dada la forma con la que se selecciona el archivo entre los distintos SOURCE se debe tener en cuenta que podría haber varias imágenes que el navegador podría representar, pero solo se indicará la primera que se adapte a las reglas en la mediaquery.

Sabiendo esto, podría ser interesante quitar el mediaquery de algún SOURCE, para que se use ese de manera predeterminada. En ese caso lo lógico sería colocar el SOURCE sin mediaquery al final del todo.

```
<picture>
    <source media="(min-width: 900px)" srcset="grande.png">
    <source media="(min-width: 550px)" srcset="media.png">
    <source srcset="peque.png">

    <img src="predeterminado.png">
</picture>
```

En el código anterior se usaría grande.png para pantallas de 900 píxeles en adelante. Se usaría media.png en pantallas de 550 píxeles y hasta 899 píxeles de anchura. Se usaría peque.png en todas las demás pantallas. Por último se usaría predeterminado.png en cualquier navegador que no sea capaz de interpretar el elemento PICTURE.

Sizes: Este atributo nos ayudará a realizar una selección de la imagen apropiada de una manera distinta. Es un poco más sofisticado y por ello puede parecer un poco más complejo, pero en la práctica estamos delegando más al navegador la elección de la imagen que irá a utilizar, por lo que no debe causar problemas.

Hasta el momento al navegador le estamos indicando las imágenes que debe usar para cada rango de píxeles. Además sobre qué imágenes deben aplicarse para distintas densidades de píxeles. Ahora con el atriburo "sizes" podemos indicar la anchura de la imagen y dejar al navegador que escoja la imagen en función de varios archivos proporcionados. El navegador será el responsable de, atendiendo al tamaño de la ventana y la densidad de píxeles, escoger la mejor alternativa.

Este atributo se combina con el atributo de antes, "srcset" para indicar las distintas alternativas de archivos gráficos.

```
<picture>
    <source media="(min-width: 600px)" sizes="65vw" srcset="peque.png 256w, media.png 456w, grande.png 856w, gigante.png 1280w">
    <img src="media.png" sizes="65vw" srcset="peque.png 256w, media.png 456w, grande.png 856w, gigante.png 1280w">
</picture>
```

> **Nota:** Como ves en el anterior código, los dos atributos "sizes" y "srcset" pueden aplicarse también a una imagen (etiqueta IMG) sin necesidad de incluir ésta en un elemento PICTURE. Son dos atributos nuevos de las imágenes que podemos usar para generar imágenes responsive, o para ser más detallistas a la hora de aplicar los diferentes archivos gráficos.

Para el código anterior. Anchuras de pantalla superiores a 600px aplica lo indicado en la etiqueta SOURCE y para pantallas menores se aplica lo indicado en la etiqueta IMG. Es un poco redundante, pero también podría ocurrir que un navegador no conozca PICTURE pero sí los atributos "sizes" y "srcset".

> **Nota:** Al probar este código para imágenes responsive lo normal es que hagamos grande y pequeña la

ventana del navegador. En la mayoría de casos apreciaremos que la imagen va cambiando, haciéndose grande a medida que aumentamos la ventana. No obstante, debido al sistema de caché, y la inteligencia del navegador, puede que en ocasiones parezca que no te muestra la imagen que debería. Esto es porque, si el navegador ha descargado ya la imagen de alta resolución, aunque reduzcas la ventana, te seguirá mostrando esa imagen grande que se supone que es la de más calidad (a no ser que indiques otra cosa con las mediaquery del atributo "media").

type: Este atributo permite indicar el tipo de una imagen, de modo que el navegador pueda seleccionarlo si es que conoce ese formato de archivo.

Compatibilidad con el elemento picture

Este nuevo elemento está disponible ya en la mayoría de los navegadores. Solo falta compatibilidad con Internet Explorer, Safari para iOS y Opera Mini. Sin embargo, debido al fallback que se indica con la etiqueta IMG, no hay motivo para dejar de usarlo.

Es perfectamente usable ya mismo porque los navegadores que lo interpreten gozarán de la posibilidad de tener imágenes adaptadas a resoluciones. Los que aun no lo implementen simplemente lo ignorarán, dejando toda la responsabilidad sobre la imagen indicada en la etiqueta IMG de toda la vida.

No obstante, si queremos dar soporte para que algún navegador antiguo también pueda implementar imágenes responsive con el elemento picture, podemos cargar el correspondiente polyfill. Este proyecto llamado "Picturefill" contiene un script para que mediante Javascript se pueda dar soporte a PICTURE, así como los atribtos "srcset" y "sizes".

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *12/04/2016*
Disponible online en *http://desarrolloweb.com/articulos/imagenes-responsive-etiqueta-picture.html*