DESARROLLO DE INTERFACES WEB

DISEÑO RESPONSIVE

La técnica que permite hacer webs adaptables a las condiciones del ordenador o dispositivo que las está accediendo

Pablo Sanz Raso

2º CFGS Desarrollo de Aplicaciones Web

ÍNDICE:

Conceptos básicos del diseño web responsive

El diseño web responsive, responde a las necesidades de los usuarios y los dispositivos que utilizan. Según sea el tamaño y las capacidades del dispositivo, los usuarios verán el contenido que se muestra de una forma diferentes. Por ejemplo, en un móvil la vista será en una sola columna; una tableta el mismo contenido se verá en dos columnas.

Los tamaños de los dispositivos cambian constantemente, por lo que es importante que el sitio pueda adaptarse a cualquier tamaño de pantalla. Además, los dispositivos pueden tener distintas características con las que interactuamos con ellos.

Consejos para que el diseño responsive sea más fácil de producir:

- **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”.
- **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.
- **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.
- **Usar el Javascript para lo que es**: 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.
- **Unidades relativas**: 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.

Las técnicas responsive se focalizan en diseñar para:

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.

o 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.

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.

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.

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.

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.

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.

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.

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.

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.

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 Media Queries

Orden de las Media Queries es importante y por tanto debes tener cuidado con ello. Así pues, el orden que se usa normalmente para las MediaQueries (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.
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.

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.

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.

El viewport 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.

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.

En dispositivos móviles 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.

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.

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.

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.

`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.

`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.

`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.

Detectar Media Queries, usando programación Javascript

Son de sobra conocidas las CSS Media queries, que nos permiten detectar distintas características de los dispositivos para aplicar unos estilos u otros en función de ellas.

Seguramente las usamos intensivamente en nuestros desarrollos, pero desde CSS. En este artículo veremos cómo podemos hacerlo desde Javascript. Se trata de una práctica muy fácil, que usa un API de Javascript que tiene total soporte en los navegadores, como podemos ver en la siguiente imagen:

¿Por qué detectar las media queries desde Javascript?

Es menos habitual que desde Javascript necesitemos detectar si ciertas media queries se están evaluando positivamente, pero también nos podemos ver en necesidad de ello. Hay veces que ciertos mecanismos es solo necesario realizarlos cuando estamos viendo la página en un ordenador con pantalla grande, pero no en un móvil o un tablet

Método matchMedia para testar media queries

El método que vamos a usar para poder comprobar si una media query se cumple se llama matchMedia(). Está dentro del objeto "window".

```
var mediaqueryList = window.matchMedia("(min-width: 500px)");
```

El método **matchMedia()** te devuelve un objeto "Media Query list". Básicamente te informa sobre la mediaquery que se ha comprobado y su valor. Si queremos ver si esta media query se cumple en un momento dado, simplemente tenemos que evaluar la propiedad "matches" de este objeto.

```
if(mediaqueryList.matches) {
  alert('La media query se cumple');
}
```

Ese objeto "mediaqueryList" es dinámico, en el sentido que, si las condiciones del navegador cambian a lo largo del tiempo, también cambiará la propiedad "matches". Por tanto, podría ocurrir que esa evaluación tenga resultados distintos en momentos distintos, si por ejemplo el usuario redimensiona la ventana del navegador.

Definir un manejador de evento para atender cambios en la evaluación de las media queries

Lo más seguro es que en tu aplicación quieras estar pendiente de los cambios en las media queries que estés evaluando con Javascript, para realizar acciones cuando esto ocurra.

Claro que podrías preguntar constantemente por el valor de la propiedad "matches", pero lo más eficiente es asociar un manejador de eventos para realizar acciones justamente cuando esa media query cambie de valor.

Esto lo conseguimos con el método "addListener" del objeto mediaqueryList, tal como se puede ver a continuación.

```
var mediaqueryList = window.matchMedia("(max-width: 920px)");
mediaqueryList.addListener( function(EventoMediaQueryList) {
      console.log('Ejecutado el listener', EventoMediaQueryList);
});
```

En este caso hemos asociado una función anónima como manejador de eventos. En esa función recibimos un objeto de tipo "MediaQueryListEvent", que nos ofrece información detallada sobre el evento que se ha producido.

Comprobar si después de ese evento se cumple o no la media query es tan sencillo como volver a evaluar su propiedad "matches", usando en este caso el propio objeto evento que estamos recibiendo por parámetro en el navegador.

```
var mediaqueryList = window.matchMedia("(orientation: portrait)");
mediaqueryList.addListener( function(EventoMediaQueryList) {
  if(EventoMediaQueryList.matches) {
    // Realizamos las acciones cuando cambia el estado de la mediaquery y ahora cumple su valor
    alert('Ahora has pasado a modo portrait, con tu pantalla orientada en vertical');
  } else {
    alert('Ahora has pasado a modo landscape, con tu pantalla orientada en horizontal');
  }
});
```

Puedes eliminar el manejador de eventos en cualquier momento, con el método removeListener(). Solo que para usarlo no puedes haber asociado el manejador con una función anónima como hemos hecho antes.

El código nos quedaría más o menos como el que sigue:

```
var mediaqueryList = window.matchMedia("(orientation: portrait)");

function manejador(EventoMediaQueryList) {
  if(EventoMediaQueryList.matches) {
    alert('La media query ahora se cumple');
  } else {
    alert('La media query ya no se cumple');
  }
}
// Asociamos el manejador de evento
mediaqueryList.addListener(manejador);
// Quitamos el manejador de evento para dejar de recibir notificación de cambios
mediaqueryList.removeListener(manejador);
```

Breakpoints

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.

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.

Medidas de teléfonos móviles

Maximo 600 px.

Medidas de tablets

Entre 600px a 992px.

Medidas ordenadores de escritorio

Desde 992px en adelante.

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.

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.

Las siguientes unidades son unidades relativas:

- %: 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)

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.

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)

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)

Nuevas unidades de CSS3: viewportwidth y viewportheight.

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.

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.

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
}
```

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.