Responsive Web (Ethan Marcotte)

1. A cuadrícula flexible

Se creamos o seguinte elemento:

```
#page {
	width: 960px;
	margin: 0 auto;
}
```

Estamos creando un elemento cun tamaño fixo e tamén o estamos centrando na nosa páxina. Pero se queremos aportarlle flexibilidade, cómo comezamos?

1.1. Tipos flexibles

Imos partir do seguinte código:

```
<h1>Achieve sentience with Skynet! <a href="#">Read More&raquo;</a></h1>
```

Unha cabeceira cun link dentro dela. Imos aplicarlle algo de estilo:

```
body {
	background-color: #DCDBD9;
	color: #2C2C2C;
	font: normal 100% Cambria, Georgia, serif;
}
```

Como se pode apreciar, estableceuse o tamaño da fonte en 100%. Isto o que fai é coller o tamaño de fonte por defecto nun navegador que, na maioría dos casos, é 16 pixels.

Probádeo utilizando a fonte base de 16px e contrastando as diferencias.

Poderíamos completar o noso código do seguinte xeito:

```
h1 {
	font-size: 24px;
	font-style: italic;
	font-weight: normal;
}
h1 a {
	color: #747474;
	font: bold 11px Calibri, Optima, Arial, sans-serif;
	letter-spacing: 0.15em;
	text-transform: uppercase;
	text-decoration: none;
}
```

Vemos que estamos a mezclar unidades: px con em. Aínda que as dúas sexan relativas, unha é máis relativa ca outra: em depende do contexto mentres que px depende da resolución que lle teñamos á pantalla (número de píxeles que pode representar). O ideal sería que fosen relativas ó contexto. Para facer isto podemos basearnos na seguinte fórmula:

obxectivo ÷ contexto = resultado

Por exemplo, imos modificar o tamaño do tipo de letra do `<h1>` expresado en píxeles por outro expresado en ems. Partindo da premisa de que a nosa fonte base é o 100%, é dicir, 16 px o noso contexto será ese mesmo. Entón:

24 ÷ 16 = 1.5

Faremos entón o seguinte:

```
h1 {
	font-size: 1.5em;
	font-style: italic;
	font-weight: normal;
}
```

Probádeo e comprobar si é certo.

Pasaremos agora a modificar o tamaño da letra do enlace. Neste caso o noso contexto cambiou, xa que non estamos a falar do contexto xeral (<body>) senón que o enlace se atopa dentro doutro contexto (<h1>), polo tanto as referencias cambiaron:

11 ÷ 24 = 0.4583333333333333

Hai que fixarse que non se fixo redondeo ningún. Usamos 0,4583333333333333 e non 0.4 ou 0.45 ou... Hai que darlle ó navegador a maior información posible. Xa se encargará él de levala a cabo do xeito máis eficiente que poida. Teremos pois:

```
h1 a {
	color: #747474;
	font: bold 0,4583333333333333em Calibri, Optima, Arial, sans-serif;
	letter-spacing: 0.15em;
	text-transform: uppercase;
	text-decoration: none;
}
```

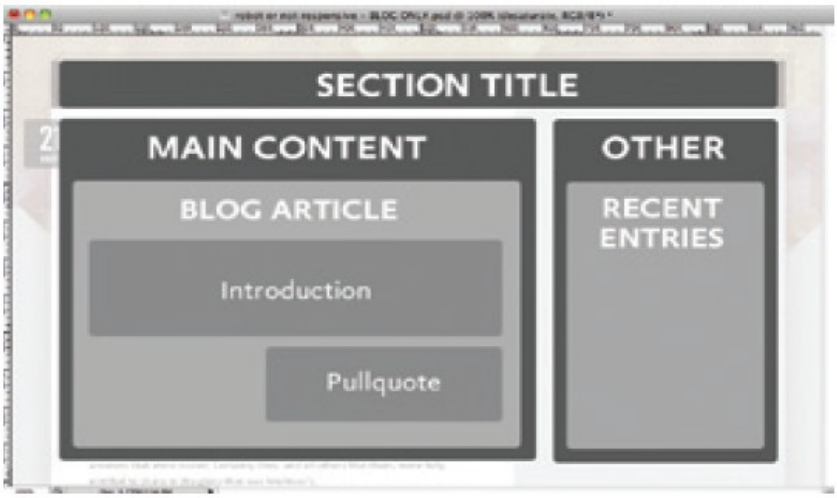
Probádeo e comprobar si é certo.

Probádeo tamén como se vos equivocárades de contexto e tomases o body de referencia (16px).

Acabamos de conseguir unha sinxela páxina co texto totalmente escalable e redimensionable.

1.2. Creando unha plantilla flexible

Imos a construír un esqueleto dunha web. Por exemplo así:




```
<div id="page">
   <div class="blog section">
      <h1>Achieve sentience with Skynet! <a href="#">Read More&raquo;</a></h1>
      <div class="main">
         ...
      </div><!-- /end .main -->
      <div class="other">
         ...
      </div><!-- /end .other -->
   </div><!-- /end .blog section -->
</div><!-- /end #page -->
```

Temos un contenedor xenérico para a páxina completa (`#page`), que conterá o módulo dun blog (`.blog`). Dentro do blog teremos outros dous contedores: un principal (`.main`) para os nosos artículos e outro (`.other`) para outras cousas.

Para aplicarlle un estilo faremos:

```
#page {
      margin: 36px auto;
      width: 960px;
}
.blog {
      margin: 0 auto 53px;
      width: 900px;
}
.blog .main {
      float: left;
      width: 566px;
}
.blog .other {
      float: right;
      width: 331px;
}
```

Fixamos o ancho do contenedor principal (`#page`) en 960 píxeles e o contido principal (`.blog`) estará centrado e terá 900 píxeles de ancho. Dentro do contido principal teremos o contido onde se ubicarán os artigos (.main) e unha especie de barra lateral (.other). Ambos cun ancho de 566 e 331 píxeles respectivamente.

Como se pode observar esté resulta un deseño inflexible e pouco adaptable. Fixando o ancho en 960 píxeles aparecerá unha barra de scroll no momento en que reduzcamos o tamaño da ventá que o contén. O que teremos que facer será conseguir pasar de píxeles a porcentaxes. Comezaremos por sustituír 960 por 90%, supoñendo que estamos tomando unha resolución de pantalla de 1024 e, calculando máis ou menos a ollo.

```
#page {
      margin: 36px auto;
      width: 90%;
}
```

Calcular o ancho de .blog farase coa fórmula xa utilizada e nos da 0.9375:

```
.blog {
      margin: 0 auto 53px;
      Width: 93.75%; /* 900/960 */
}
```

É recomendable poñer os operandos comentados ó lado porque, nun futuro, poden ser de moita utilidade. Agora pasamos a calcular o ancho do resto de elementos da nosa páxina:

```
.blog .main {
      float: left;
      Width: 62.88888888888889%; /* 566/900 */
}
.blog .other {
      float: right;
      Width: 36.77777777777778%; /* 331/900 */
}
```

Acabamos de conseguir unha plantilla totalmente flexible e adaptable ó contorno.

1.3. Margins e paddings flexibles

Supoñamos que queremos que a nosa páxina teña un título na cabeceira separado (margin/padding) do límite lateral do contenedor 48 píxeles.

```
.lede {
      padding: 0.8em; /* A ollo */
      padding-left: 48px;
}
```

O de sempre, calcularemos a distancia utilizando a fórmula pero, ollo:

- Cando establecemos margins flexibles sobre un elemento, o contexto é o ancho do elemento contenedor.
- Cando establecemos paddings flexibles sobre un elemento, o contexto é o ancho do propio elemento.

Tendo isto en conta aplicaremos un padding sobre o título que, para manter simetría co resto do contido, terá un ancho de 900 píxeles.

```
.lede {
      padding: 0.8em; /* A ollo */
      padding-left: 5.33333333333333%; /* 48/900 */
}
```

Tamén, para manter simetrías, deixaremos o mesmo marxe lateral nas entradas de texto do blog e similar na barra lateral (`.recent-entries`).

```
.recent-entries {
      margin: 0 auto; /* centrado horizontalmente */
      Width: 69.78851963746224%; /* 231/331 */
}
```

Agora o libro quere introducir unha data para cada artigo. Dita data terá 69 px de ancho. Para facela adaptable teremos que calcular o seu tamaño tendo en conta o seu contenedor. Deixamos 48 píxeles de marxe esquerdo e deixaremos tamén outros 48 de marxe dereito. Así que, a información real ocupará 566-48-48 (470 px).

```
.date {
      float: left;
      width: 14,68085106382979%; /* 69/470 */
}
```

Para darlle un efecto de maior dinamismo, o libro propón desprazar á esquerda do contido a caixa coa data. Por exemplo 81 px. Cal será o tamaño a tomar como referencia? Pois o tamaño do contendor onde estaba/está (470 px).

```
.date {
      float: left;
      Margin-left: -17,23404255319149%; /* -81/470 */
      width: 14,68085106382979%; /* 69/470 */
}
```

2. Imaxes flexibles

Imos a comezar a meter imaxes no noso proxecto. A primera estará dentro dun contenedor (`div`) do seguinte xeito:

```
<div class=”figure”>
	<p>
		<img src=”robot.jpg” alt=”Robot” /> <!-- http://bkaprt.com/rwd/10/ -->
		<b class=”figcaption”>Lo, the robot walks</b>
	</p>
</div>

.figure {
	float: right;
	margin-bottom: 0.5em;
	margin-left: 2,531645569620253%; /* 12/470 */
	width: 49,14893617021277%; /* 231/470 */
}
```

A imaxe escollida ten 409x640 píxeles de dimensión. Sobra dicir que excede o tamaño reservado para ela no contenedor. Isto arranxámolo coa propiedade `max-width` do seguinte xeito:

```
.img {
	max-width: 100%;
}
```

De feito, esta propiedade non sé se pode aplicar ás imaxes senon a outros elementos coma `embed`, `object`, `video`,...

De todos xeitos, esta propiedade tamén ten un problema: chámase IE7 e versións anteriores. É posible que IE7 a soporte pero seguro que IE6 non. Se se quere proporcionar soporte para ditas versións haberá que incluír as seguintes etiquetas:

```
/* Para versións posteriores a IE6 */
img, embed, object, video {
	max-width: 100%;
}

/* Para versións anteriores a IE6 */
img, embed, object, video {
	width: 100%;
}
```

Para facer isto podemos utilizar as etiquetas condicionais:

```
<!--[if IE 6]>
	Follas de estilo para Internet Explorer 6
<![endif]-->

<!--[if gte IE 7]>
	Follas de estilo para Internet Explorer 6 e superiores
<![endif]-->

<!--[if gt IE 6]>
	Follas de estilo para navegadores superiores a Internet Explorer 6
<![endif]-->

<!--[if lte IE 6]>
	Follas de estilo para Internet Explorer 6 e inferiores
<![endif]-->

<!--[if lt IE 7]>
```

```
      Follas de estilo para navegadores inferiores a Internet Explorer 7
<![endif]-->
```

Pero ollo que `width` e `max-width` non teñen demasiado que ver. Só estamos a falar de soporte, non de que fagan o mesmo.

Como acabamos de ver, na maioría dos casos podemos solucionar o problema con `max-width` pero, haberá outros casos (imaxes moi anchas, encabezados,...) nos que podemos sustituír dita opción por `overflow: hidden`.

Exercicio de Imaxes width.

Os posibles problemas que poidan xurdir con navegadores Iexplorer antigos e a resolución do rescalado das imaxes se poden solventar cun filtro CSS propietario de Microsoft (AlphaImageLoader).

2.1. Backgrounds flexibles

Podemos utilizar imaxes de background flexibles coa propiedade CSS3 `background-size`, á que tamén lle podemos indicar os porcentaxes desexados.

3. Media Queries

Xa vimos como conseguir que o noso proxecto se adapte ó tamaño de calquer tipo de pantalla sen ningún problema. Non obstante, isto que debería ser unha vantaxe, en moitos casos pode resultar un inconvinte, xa que pode darse o caso que a redimensión automática afecte a visibilidade do site. Se reducimos moito o tamaño de ventá as imaxes poden chegar a ser ilexibles, por exemplo. Para isto xurden as **media queries**. De todos xeitos, antes de que as media queries se convertisen nun estándar (2012) o W3C proporcionaba os **media types** para CSS2. Se vos lembrades, os media types eran algo polo estilo:

```
<LINK REL="stylesheet" TYPE="text/css" MEDIA="print, handheld" HREF="foo.css">
```

Deste xeito podíamos reflexar comportamentos diferentes ante entornos diferentes. Os media types recoñecidos son:

- `all`.
- `braille`.
- `embossed`.
- `handled` (teléfonos e tablets).
- `print`.
- `projection`.
- `screen`.
- `speech` (lectores de pantalla).
- `tty`.
- `tv`.

Tamén podían ser incluídos no arquivo de CSS deste xeito:

```
@media screen {
	body {
		font-size: 100%;
	}
}

@media print {
	body {
		font-size: 15pt;
	}
}
```

Coa aparición (e posterior inundación) de terminais e dispositivos móbiles xurdiron os problemas. Ben é certo que existía un media type que se facía cargo desta categoría (handled) pero conforme ían aparecendo novas versións e ditas versións utilizaban navegadores cada vez máis avanzados optaban por utilizar screen en vez de handled. A solución final pasa por media query. Un exemplo de media query sería:

```
@media screen and (min-width: 1024px) {
	body {
		font-size: 100%;
	}
}
```

Cada unha delas estará composta por un media type (screen) e pola query en sí (min-width: 1024px). A query tamén estará dividida en dúas partes: o nome da propiedade (win-width) e o valor da mesma (1024px). Alternativamente tamén podemos facer:

```
<LINK rel="stylesheet" href="foo.css" media="screen and (min-width: 1024px)">
```

Ou tamén:

```
@import url (“foo.css”) screen and (min-width: 1024px);
```

Tamén é interesante suliñar que se poden compoñer propiedades. Por exemplo:

```
@media screen and (min-device-width: 480px) and (orientation:landscape) { ...
```

Lista de propiedades que podemos utilizar:

- width (ancho da ventá).
- height (alto da ventá).
- device-width (ancho da pantalla do dispositivo).
- device-height (alto da pantalla do dispositivo).
- orientation (normal/portrait ou apaisada/landscape).
- aspect-ratio (16:9,...).
- device-aspect-ratio.
- color (número de bits por color).
- color-index.
- monochrome.
- resolution (densidade dos píxeles, por exemplo, 300 dpi).
- scan.
- grid.

Imos a comezar a utiliza as media queries co código de exemplo. Empezaremos creando o seguinte estilo para un título:

```
.main-title {
	background-color: #000;
	color: #FFF;
	font-weight:normal;
	font-size:3.625em;
	font-family:"League Gothic", "Arial Narrow", Arial, sans-serif;
	text-transform:uppercase;
}
```

Se redimensionamos o tamaño da ventana do navegador vemos que se adapta perfectamente perom se a facemos moi pequena, o resultado non é o desexado. Entón imos engadir a continuación a seguinte media query:

```
@media screen and (max-width: 768px) {
	.main-title {
		font-size:1.5em;
		font-family:Calibri;
	}
}
```

O que estamos a facer e sobrescribir as propiedades do tamaño da fonte o do tipo de fonte para que, no caso en que estemos a utilizar dispositivos con menos de 768 px de ancho, o resultado de visualización sexa máis axeitado.

O contenedor principal tamén estaba a ocupar máis ou menos o 90% do ancho da páxina en resolucións grandes. Podemos facer que, no caso de atoparnos con dispositivos pequenos, adapte o contido do seguinte xeito:

```
#page {
	margin: 36px auto;
	width: 90%;
}

@media screen and (max-width: 768px) {
	#page {
		position:relative;
		margin:20px;
		width: auto;
	}
}
```

Pero, ollo, non sempre temos que mirar cara resolucións máis pequenas. Pode darse o caso de que, a grandes resolucións, nos atopempos con que a nosa páxina perde funcionalidade ou que simplemente as imaxes deixan de verse ou de situarse coma nos queremos. É por iso que temos que controlar a máxima resolución de pantalla pero tamén a mínima:

```
@media screen and (min-width: 1200px) {
    .welcome, .blog, .gallery {
        width: 49.375%;
    }
    .welcome, .gallery {
        float: right;
        margin: 0 0 40px;
    }
    .blog {
        float: left;
        margin: 0 0 20px;
    }
}
```

Deste xeito, a grandes resolución (a partires de 1200 px) a nosa páxina adaptarase para amosar máis imformación por pantalla.

3.1. Compatibilidade

As media queries están soportadas pola maioría de navegadores do mercado pero isto non quere dicir que TODOS os navegadores existentes as soporten. Internet Explorer 8 e inferiores, por exemplo, non proporcionan soporte nativo para ellas. Hai certas solucións pero todas pasan polo uso de Javascript. Por exemplo:

- css3-mediaquerises-js (https://code.google.com/p/css3-mediaqueries-js/)
- respond.js (https://github.com/scottjehl/Respond). Máis sinxela, únicamente da soporte a min-width e max-width en navegadores vellos.

3.2. Viewport

Cando Apple lanzou o iPhone en 2007, crearon un novo valor para o atributo name da etiqueta meta: `viewport`. Por defecto, os navegadores dos dispositivos móviles (iPhone, iPad,...) supoñen que as webs teñen un ancho fixo (por exemplo: 980 píxeles) co que imos ver do mesmo xeito unha web no noso móbil e no noso PC, sen ter en conta que hai unha diferencia abrumadora de tamaño entre ambas pantallas. Hai que ter coidado tamén porque pode ser que as media queries tampouco se apliquen correctamente. Nos podemos cambiar este comportamento coa etiqueta **`viewport`**, podemos controlar o tamaño do lienzo (canvas). Por exemplo, podemos indicar ás nosas páxinas que terán un ancho fixo de 320px.

```
<meta name=”viewport” content=”width=320” />
```

Isto faise porque os móviles tenden a redimensionar o tamaño das webs facendo que se reduza a visibilidade e a funcionalidade. O content pode ter os seguintes valores:

- **width**: valor do ancho do viewport. Pode estar en tamaño absoluto (píxeles pero sen indicar a unidade) ou que detecte o tamaño do dispositivo (`device-width`). O valor que se soe utilizar case sempre é device-width, para traballar sempre co ancho real da pantalla do dispositivo.
- **height**: valor do alto do viewport. Pode estar en tamaño absoluto (píxeles pero sen indicar a unidade) ou que detecte o tamaño do dispositivo (`device-width`). No é demasiado utilizado.
- **initial-scale**: define a escala inicial do viewport. Os valores poden ser decimais, 1, 2, 3,... O mínimo que acepta é 0.1.
- **user-scalable**: podemos permitir (YES, por defecto) que o usuario poida facer zoom ou non (NO).
- **minimum-scale**: podemos indicar cal é o mínimo zoom que lle pode facer á páxina. Os valores poden ser decimais, 1, 2, 3,...
- **maximun-scale**: podemos indicar cal é o máximo zoom que lle pode facer á páxina. Os valores poden ser decimais, 1, 2, 3,...

Viewport de momento só funciona nos navegadores dos terminais móbiles. Tampouco ten moito sentido que o faga nos navegadores de PC.

Imos a ver uns exemplos de uso do viewport. Partimos da base dunha web moi sinxela cun h1 e unha imaxe de 320px de ancho:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8">
    <title>Sitio sin viewport</title>
</head>
<body>

<h1>Escalando el tamaño de un sitio</h1>
<img src="magoo.jpg" alt="magoo" width="320" height="427" />

</body>
</html>
```

As capturas fixéronse sobre un iPhone 5 en orientación portrait.

Sen viewport

Podemos ver que o navegador do movil emula unha pantalla de 980px co que, o resultado final e amosar a páxina “encollida”, é dicir, non se está amosando a tamaño real

Viewport 320

```
<meta name="viewport" content="width=320"/>
```

Neste caso estamos indicando que o canvas serán 320 px, por iso se amosa a imaxe ocupando todo o ancho do dispositivo e o h1 cortado porque non cabe sin salto de liña.

Viewport 640

```
<meta name="viewport" content="width=640"/>
```

Neste caso estamos indicando que o canvas serán 640 px, por iso se amosa a imaxe ocupando a metado do ancho do dispositivo e o h1 sobrepasando o ancho da imaxe porque ten sitio para elo. Se o poñemos en formato landscape o comportamento será o mesmo, é dicir, o ancho da pantalla seguirá sendo 640, a imaxe ocupará a metade do ancho e o texto sobrepasará o ancho da imaxe.

Viewport device-width

```
<meta name="viewport" content="width=device-width"/>
```

Neste caso estamos indicándolle que adapte ó contido ó 100% do ancho do dispositivo (tanto portrait como landscape), sen ter que utilizar medidas absolutas. Este será o valor recomendado para a maioría das veces.

4. Se responsive

Xa temos todos os coñecementos necesarios para facer a nosa web totalmente flexible e adaptable ós cambios de ventana e adaptable ós diferentes tipos de dispositivos que a visualizarán. A pregunta é: xa está? Con iso chega? A resposta sería NON.

Dun tempo a esta parte a evolución no uso dos dispositivos móbiles foi en aumento, chegando a desprazar claramente en número de adquisicións ós ordenadores convencionais. Isto implica un cambio de filosofía á hora de realizar as nosas webs. O usuario que utiliza un smartphone, por exemplo, non se comporta igual ou non recibir a mesma información (ou do mesmo xeito) que se se atopase nun pc. O usuario do smartphone prefire axilidade, concisión e claridade.

O feito de que se modifique o comportamento dos usuarios cando utilizan un determinado tipo de dispositivos (que, cada vez, vai máis en aumento) fai que este comportamento se extenda no uso dos dispositivos tradicionais. É dicir, os usuarios queren navegar no pc dun xeito parecido ó móbil.

Por iso xurden tendencias como **mobile first design**, que consisten simplemente en deseñar a nosa web inicialmente pensando nun dispositivo móbil e extender dito deseño (ou a veces nin sequera é necesario) ó resto dos dispositivos. Por exemplo, partir do deseño nun smartphone, pasar por unha tablet e rematar nun pc.