Patrones de diseño web adaptables

Los patrones de diseño web adaptables evolucionan rápidamente, pero existen varios patrones establecidos que funcionan bien en los diferentes equipos de escritorio y dispositivos móviles.

La mayoría de los diseños que se usan en las páginas web receptivas se pueden categorizar dentro de cinco clases de patrones: mostly fluid, column drop, layout shifter, tiny tweaks y off canvas. En algunos casos, en una página se puede usar una combinación de patrones; por ejemplo, column drop y off canvas. Estos patrones, originalmente identificados por Luke Wroblewski, son un punto de partida sólido para cualquier página receptiva.

Los patrones

Con el propósito de lograr una comprensión simple y fácil, cada uno de los que se presentan a continuación se crearon con lenguaje de marcado real a través de **flexbox**, generalmente con tres **div**de contenido dentro de un contenedor principal de **div**. Cada muestra se escribió primero a partir de la vista más pequeña y se agregaron puntos de interrupción cuando fue necesario. El **modo de diseño Flexbox es muy compatible** con los navegadores modernos; sin embargo, es posible que el proveedor deba realizar ajustes previos para lograr una compatibilidad óptima.

Mostly Fluid

El patrón Mostly fluid consiste, principalmente, en una cuadrícula fluida. Por lo general, en las pantallas grandes o medianas se mantiene el mismo tamaño y simplemente se ajustan los márgenes en las más anchas.

En las pantallas más pequeñas, la cuadrícula fluida genera el reprocesamiento del contenido principal, mientras que las columnas se apilan verticalmente. Una de las mayores ventajas de estepatrón es que, en general, solo se necesita un punto de interrupción entre las pantallas grandes y las pequeñas.

En la vista más pequeña, cada **div** de contenido se apila verticalmente. Una vez que el ancho de la pantalla alcanza los 600 píxeles, el **div** de contenido principal permanece en **width: 100%**, mientras que el **div** de contenido secundario se muestra como dos columnas debajo del **div** principal. Al superarse los 800 píxeles, el **div** del contenedor adopta ancho fijo y se centra en la pantalla.

Entre los sitios en los que se usa este patrón se incluyen los siguientes:

•A List Apart

•Media Queries

•SimpleBits

```
.container {
   display: -webkit-flex;
   display: flex;
   -webkit-flex-flow: row wrap;
   flex-flow: row wrap;
}
.c1, .c2, .c3, .c4, .c5 {
   width: 100%;
}
@media (min-width: 600px) {
   .c2, .c3, .c4, .c5 {
      width: 50%;
   }
}
@media (min-width: 800px) {
   .c1 {
      width: 60%;
   }
   .c2 {
      width: 40%;
   }
   /* Using 33.33%, doesn't always work right due to rounding */
   .c3, .c4 {
      width: 33%;
   }
   .c5 {
      width: 34%;
   }
}
@media (min-width: 800px) {
   .container {
      width: 800px;
```

```
      margin-left: auto;
      margin-right: auto;
   }
}
```

Colocación de columnas

En el caso de los diseños con varias columnas de ancho completo, durante el proceso de colocación de columnas éstas únicamente se colocan de forma vertical debido a que el ancho de la ventana es demasiado reducido para el contenido.

En un momento dado, todas las columnas se apilan verticalmente. La selección de puntos de interrupción para este patrón de diseño depende del contenido y cambia para cada diseño.

Como sucede con los ejemplos que son principalmente fluidos, el contenido se coloca verticalmente en la vista más pequeña, pero a medida que se expande la pantalla a más de 600 píxeles, los **div**de contenido principal y secundario ocupan todo el ancho de la pantalla. El orden de los **div**se configura con la propiedad CSS de orden. Con 800 píxeles, se muestran los tres **div**de contenido en todo el ancho de la pantalla.

Entre los sitios en los que se usa este patrón se incluyen los siguientes:

•Modernizr

•Wee Nudge

```
.container {
   display: -webkit-flex;
   display: flex;
   -webkit-flex-flow: row wrap;
   flex-flow: row wrap;
}
.c1, .c2, .c3 {
   width: 100%;
}
@media (min-width: 600px) {
   .c1 {
      width: 60%;
      -webkit-order: 2;
      order: 2;
```

```
  }
  .c2 {
    width: 40%;
    -webkit-order: 1;
    order: 1;
  }
  .c3 {
    width: 100%;
    -webkit-order: 3;
    order: 3;
  }
}
@media (min-width: 800px) {
  .c2 {
    width: 20%;
  }
  .c3 {
    width: 20%;
  }
}
```

Layout shifter

El patrón Layout shifter es el más adaptable, ya que posee varios puntos de interrupción en diferentes anchos de pantalla.

La clave para este diseño es el desplazamiento del contenido, en lugar de su reprocesamiento y colocación debajo de otras columnas. Debido a las diferencias significativas entre cada punto de interrupción principal, es más complejo de mantener, y es posible que se deban realizar cambios dentro de los elementos, no solo en el diseño de contenido general.

En este ejemplo simplificado, se muestra el patrón Layout shifter. En las pantallas más pequeñas, el contenido se apila verticalmente, pero cambia considerablemente a medida que se agranda la pantalla, con un **div**a la izquierda y dos **div** apilados a la derecha.

Entre los sitios en los que se usa este patrón se incluyen los siguientes:

•Food Sense

•Ejemplo de Seminal Responsive Design

•Andersson-Wise Architects

```
.container {
   display: -webkit-flex;
   display: flex;
   -webkit-flex-flow: row wrap;
   flex-flow: row wrap;
}
.c1, .c2, .c3, .c4 {
   width: 100%;
}
@media (min-width: 600px) {
   .c1 {
      width: 25%;
   }
   .c4 {
      width: 75%;
   }
}
@media (min-width: 800px) {
   .container {
      width: 800px;
      margin-left: auto;
      margin-right: auto;
   }
}
```

Tiny tweaks

El patrón Tiny tweaks permite realizar pequeños cambios en el diseño, como ajustar el tamaño de la fuente, cambiar el tamaño de las imágenes o desplazar el contenido de maneras muy poco significativas.

Funciona correctamente en diseños con una sola columna, como los sitios web lineales de una sola página y los artículos con mucho texto.

Como lo indica el nombre, en este ejemplo se producen pocos cambios pequeños cuando cambia el tamaño de la pantalla. A medida que aumenta el ancho de la pantalla, también cambian el tamaño de la fuente y el relleno.

Entre los sitios en los que se usa este patrón se incluyen los siguientes:

•Ginger Whale

•Future Friendly

```
.c1 {
   padding: 10px;
   width: 100%;
}
@media (min-width: 500px) {
   .c1 {
      padding: 20px;
      font-size: 1.5em;
   }
}
@media (min-width: 800px) {
   .c1 {
      padding: 40px;
      font-size: 2em;
   }
}
```

Off canvas

En lugar de apilar contenido verticalmente, el patrón Off canvas coloca contenido menos usado (tal vez menús de navegación o de apps) fuera de la pantalla y solo lo muestra cuando el tamaño de la pantalla es suficientemente grande. En las pantallas más pequeñas, el acceso al contenido es posible con solo a un clic.

En esta muestra, en lugar de apilarse el contenido verticalmente, se usa una declaración **transform: translate(-250px, 0)** para ocultar dos de los **div** de contenido fuera de la pantalla mediante la propiedad. Se usa JavaScript para mostrar los divs agregando una clase abierta al elemento para hacerlo visible. A medida que se ensancha la pantalla, el posicionamiento fuera de esta se elimina de los elementos y estos se muestran dentro de la ventana de visualización visible.

Ten en cuenta que en este ejemplo Safari para iOS 6 y el navegador de Android no son compatibles con la función **flex-flow: row nowrap** de **flexbox**, por lo cual se debió recurrir nuevamente al posicionamiento absoluto.

Entre los sitios en los que se usa este patrón se incluyen los siguientes:

•HTML5Rocks Articles

•Google Nexus

•Sitios para celulares de Facebook

```
body {
   overflow-x: hidden;
}
.container {
   display: block;
}
.c1, .c3 {
   position: absolute;
   width: 250px;
   height: 100%;
   /*
      This is a trick to improve performance on newer versions of Chrome
      #perfmatters
   */
   -webkit-backface-visibility: hidden;
   backface-visibility: hidden;
   -webkit-transition: -webkit-transform 0.4s ease-out;
   transition: transform 0.4s ease-out;
   z-index: 1;
}
.c1 {
   /*
   Using translate3d as a trick to improve performance on older versions of Chrome
   See: http://aerotwist.com/blog/on-translate3d-and-layer-creation-hacks/
   #perfmatters
   */
   -webkit-transform: translate(-250px,0);
   transform: translate(-250px,0);
}
.c2 {
   width: 100%;
   position: absolute;
}
.c3 {
   left: 100%;
}
.c1.open {
   -webkit-transform: translate(0,0);
```

```
    transform: translate(0,0);
}
.c3.open {
    -webkit-transform: translate(-250px,0);
    transform: translate(-250px,0);
}
@media (min-width: 500px) {
    /* If the screen is wider then 500px, use Flexbox */
    .container {
        display: -webkit-flex;
        display: flex;
        -webkit-flex-flow: row nowrap;
        flex-flow: row nowrap;
    }
    .c1 {
        position: relative;
        -webkit-transition: none 0s ease-out;
        transition: none 0s ease-out;
        -webkit-transform: translate(0,0);
        transform: translate(0,0);
    }
    .c2 {
        position: static;
    }
}
@media (min-width: 800px) {
    body {
        overflow-x: auto;
    }
    .c3 {
        position: relative;
        left: auto;
        -webkit-transition: none 0s ease-out;
        transition: none 0s ease-out;
        -webkit-transform: translate(0,0);
        transform: translate(0,0);
    }
}
```