

Responsive Web Design Patterns



By Pete LePage

Pete is a Developer Advocate

Responsive web design patterns are quickly evolving, but there are a handful of established patterns that work well across the desktop and mobile devices.

Most layouts used by responsive web pages can be categorized into one of five patterns: mostly fluid, column drop, layout shifter, tiny tweaks, and off canvas. In some cases, a page may use a combination of patterns, for example column drop and off canvas. These patterns, originally identified by Luke Wroblewski, provide a solid starting point for any responsive page.

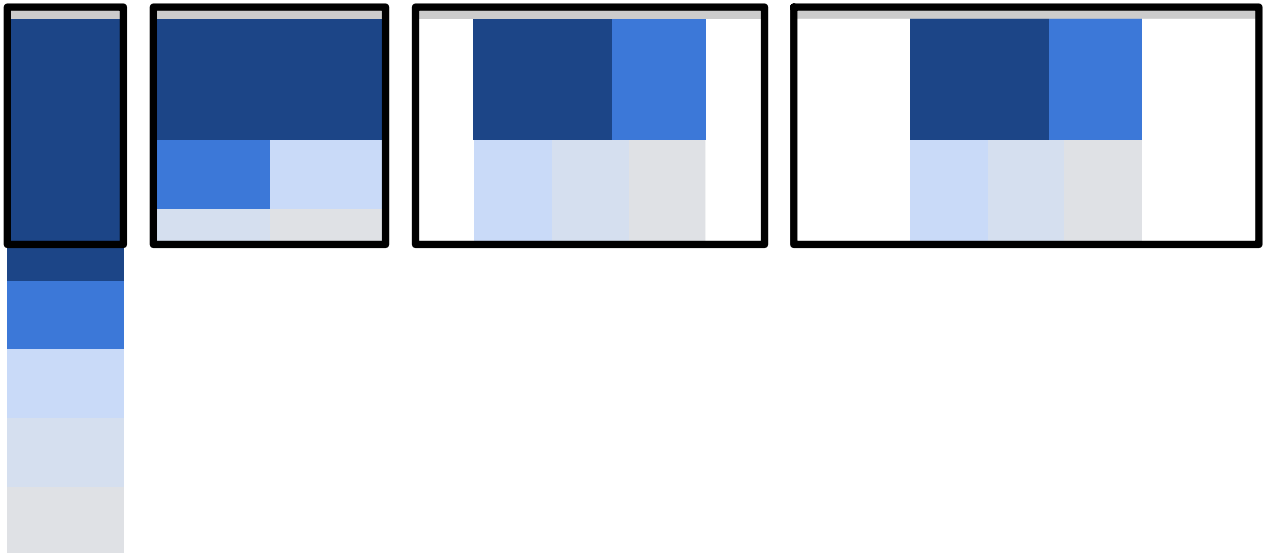
The patterns

For simplicity and ease of understanding, each the samples below were created with real markup using flexbox, typically with three content `div`'s contained within a primary container `div`. Each sample was written starting with the smallest view first, and breakpoints were added when necessary. The flexbox layout mode is well supported for modern browsers, though may still require vendor prefixing for optimal support.

Mostly Fluid

The mostly fluid pattern consists primarily of a fluid grid. On large or medium screens, it usually remains the same size, simply adjusting the margins on wider screens.

On smaller screens, the fluid grid causes the main content to reflow, while columns are stacked vertically. One major advantage of this pattern is that it usually only requires one breakpoint between small screens and large screens.



TRY IT

In the smallest view, each content div is stacked vertically. When the screen width hits 600px, the primary content div remains at `width: 100%`, while the secondary div's are shown as two columns below the primary div. Beyond 800px, the container div becomes fixed width and is centered on the screen.

Sites using this pattern include:

- [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%;
    }
    .c3, .c4, .c5 {
        width: 33.33%;
    }
}

}

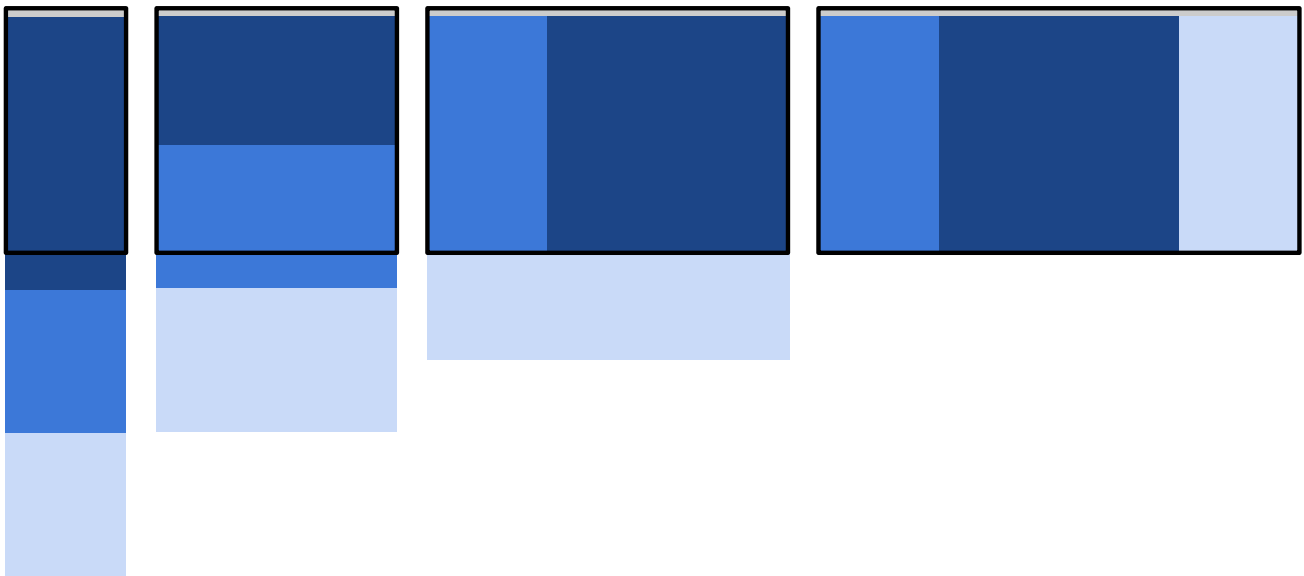
@media (min-width: 800px) {
    .container {
        width: 800px;
        margin-left: auto;
        margin-right: auto;
    }
}

```

Column drop

For full-width multi-column layouts, column drop simply stacks the columns vertically as the window width becomes too narrow for the content.



Eventually this results in all of the columns being stacked vertically. Choosing breakpoints for this layout pattern is dependent on the content and changes for each design.



TRY IT

Like the mostly fluid sample, content is stacked vertically in the smallest view, but as the screen expands beyond 600px, the primary and secondary content div's take the full width of the screen. The order of the div's is set using the order CSS property. At 800px all three content div's are shown, using the full screen width.

Sites using this pattern include:

- [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%;
    }

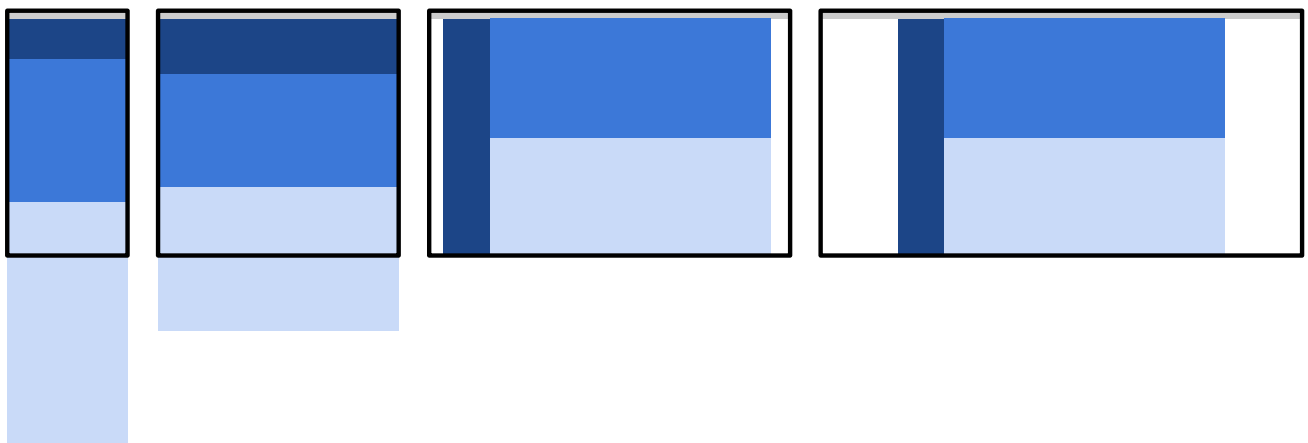
    .c3 {
        width: 20%;
    }
}

```

Layout shifter

The layout shifter pattern is the most responsive pattern, with multiple breakpoints across several screen widths.



Key to this layout is the way content moves about, instead of reflowing and dropping below other columns. Due to the significant differences between each major breakpoint, it is more complex to maintain and likely involves changes within elements, not just overall content layout.



TRY IT

This simplified example shows the layout shifter pattern, on smaller screens content is stacked vertically, but changes significantly as the screen becomes larger, with a left div and two stacked div's on the right.

Sites using this pattern include:

- [Food Sense](#) 
- [Seminal Responsive Design Example](#)
- [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

Tiny tweaks simply makes small changes to the layout, such as adjusting font size, resizing images, or moving content around in very minor ways.

It works well on single column layouts such as one page linear websites and text-heavy articles.



TRY IT

As its name implies, not much changes with this sample as the screen size changes. As the screen width gets larger, so do the font size and padding.

Sites using this pattern include:

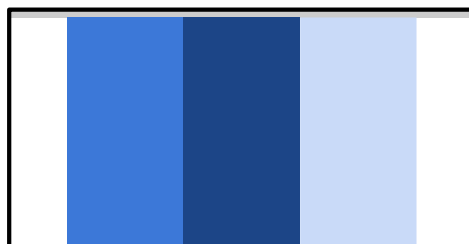
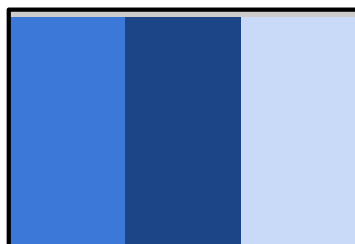
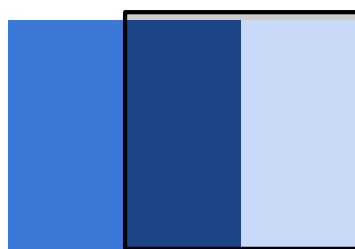
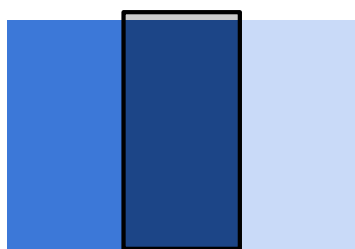
- [Ginger Whale](#) 
- [Future Friendly](#) 

```
.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

Rather than stacking content vertically, the off canvas pattern places less frequently used content—perhaps navigation or app menus—off screen, only showing it when the screen size is large enough, and on smaller screens, content is only a click away.





TRY IT

Rather than stacking content vertically, this sample uses a `transform: translate(-250px, 0)` declaration to hide two of the content divs off screen. JavaScript is used to show the divs by adding an open class to the element to make visible. As the screen gets wider, the off-screen positioning is removed from the elements and they're shown within the visible viewport.

Note in this sample, Safari for iOS 6 and Android Browser do not support the `flex-flow: row nowrap` feature of `flexbox`, so we've had to fall back to absolute positioning.

Sites using this pattern include:

- [HTML5Rocks Articles](#)
- [Google Nexus](#) 
- [Facebook's Mobile Site](#) 

```
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);
}
}
```

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see our [Site Policies](#). Java is a registered trademark of Oracle and/or its affiliates.

Last updated July 2, 2018.