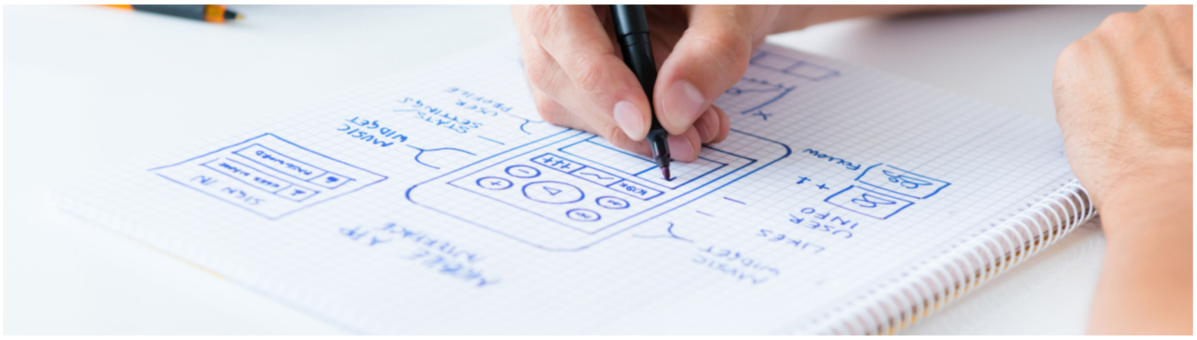
What is Mobile-First Layout?



When it comes to mobile devices, web designers are challenged to consider a number of constraints when it comes to designing webpages. For one, mobile devices generally have less bandwidth, processing power, and memory than their desktop counterparts. Additionally, mobile devices have smaller viewports. Websites that are designed with larger screens or processing power don't offer an optimum user experience when viewed on a mobile device as they often load slowly and lag in terms of performance. With the consumption of mobile media at an all-time high — and growing — **mobile-first** design strategy has become a principle in website design and development.

Recall, the opposite approach — desktop-first — designs for a desktop or large screen layouts first, and then uses media queries to override any settings for smaller devices.

As a design strategy, mobile-first dictates that a website should be designed for the smallest screen first. In other words, the mobile layout is considered the default layout upon which to build layouts for other devices and/or screen dimensions. The reasons for this are twofold:

1. Website Design Planning

The principles of user experience (the process of planning a website and creating designs) guide designers to consider mobile devices first when planning the website's messaging. The key focus is that the most important information should be clearly identified. It challenges designers to consider their designs in light of the following question:

> If there is no room on the mobile screen for certain information, why include it on the website at all?

2. Website Construction

When constructing a website using a mobile-first design, the CSS located outside of the media queries is considered global in nature. Therefore, this CSS is applied to any type of device, as it is loaded *first*.

Any media queries that are placed in the CSS stylesheet, are used to override these global settings. In a mobile-first design environment, typically these media queries are there to adjust layouts for larger dimensions — such as tablets, desktops, and larger screens. If the webpage is being viewed on a device that meets any of the media query parameters in the stylesheet, the CSS defined in that query will be applied. This mobile-first structure prevents elements from loading that might clog up a mobile display or load slowly due to a mobile device's limited processing power.

Best Practices: Design Mobile, First

When it comes to mobile-first, less is more! Be mindful of your choices during the initial planning stages of webpage design. Consider what is possible on mobile-first, then add from there.

When to Use Mobile-First vs. Desktop-First

Generally speaking, mobile-first layout is the preferred way to create a website for the following reasons.

Why is Mobile-First Preferred?	
Focused Messaging	By framing the website message for small screen sizes, the message must be focused and easy to digest. There's no room for distractions. This keeps users focused on their goals for visiting the website, or promote the website's **call-to-action**.
Improved Load Speed	Frequently, media queries are more efficient with a mobile-first approach. Rather than removing and overriding properties, as is typical with a desktop-first approach, media queries are often additive with mobile-first. By definition, mobile-first adds more CSS for larger screens while using less CSS for smaller screens. This approach is faster and leaner, thereby decreasing the time it takes a page to load.

However, desktop-first approaches are useful under the right conditions.

- If the primary device used to access a website has a large screen, with a few exceptions using smaller screens, desktop-first may be a better approach.
- If you are retrofitting an old, non-responsive website with new CSS to make a responsive layout, it may be easier to take a desktop-first approach.
- Some techniques, like creating responsive tables with CSS, work better from a desktop-first perspective.

When it comes to Mobile-First, Speed and Size Matter!

Average speed index is a measure of how quickly a page displays its content to a viewer. A best practice benchmark for this metric is three (3) seconds or better.

Example: Applying Mobile-First to a Grid Layout

Let's return to the responsive design we created using a desktop-first approach in the previous exercise. How might we change the CSS to reflect a mobile-first approach instead?

Recall the CSS code we added to the stylesheet was:

```
/* presentational code below */
```

```
   .box {
     font-family: Arial, sans-serif;
     height: 150px;
     border: 4px dotted navy;
     background-color: lightblue;
     font-size: 3rem;
     text-align: center;
     color: navy;
   }

/* web page layout starts here */
    .row {
      display: flex;
      flex-flow: row wrap;
      justify-content: space-between;
   }
    .box {
      flex-basis: 19%;
   }

   @media (max-width: 500px) {
     .box {
       flex-basis: 46%;
       background-color: lightyellow;
     }
   }

   @media (max-width: 325px) {
     .box {
       flex-basis: 100%;
       background-color: lightgreen;
     }
   }
```

Step 1

First, we have to consider what styles in the CSS are global. Currently, there are presentational styles listed at the top of the document. These styles are considered global, and are overridden by the media queries as needed. What is helpful in this example is that the media queries change the background color, so it is clear to see if the media query construct is working as expected.

Now, look at what changes with each media query breakpoint. In this example, the flex-basis value changes across breakpoints, as well as the background color. However, at the very end, in the alternative answer, we suggest that maybe our mobile styles are using the default block display. If we took this approach with mobile-first, where would our Flexbox declaration occur in the code?

Step 2

Next, to convert the CSS to a mobile-first structure, we need to reconstruct the media queries. With desktop-first, commonly web developers use max-width media queries. With this approach, max-width stopped certain styles from displaying once a screen got too large.

In contrast, with a mobile-first approach **min-width** is used with media queries. The min-width declaration means that the styles that are added with the media queries will apply to larger screens in a cumulative manner.

Exercise

The breakpoints in this desktop-first approach were set to a max-width of 500px, and a max-width of 325px. Change the CSS to reflect a mobile-first approach by rewriting the media queries as min-width. You'll need to rearrange the colors and adjust the flex-basis to preserve this code in a mobile-first approach.

Instructions:

- Click on the "styles.css" link tab at the top of the code tool.
- Adjust the CSS code to reflect a mobile-first approach using min-width to adjust the media queries.
- Click the 'run code' button to lock in your answer (**Note:** you will not notice a change in display).
- Check your CSS code against the correct code in the 'Check Your Answer' tab

Run Code Save Export Reset

index.html styles.css

```
<!doctype html>
<html>
 <head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="/styles.css" />
</head>
 <body>
<div class="row">
  <div class="box">Box 1</div>
  <div class="box">Box 2</div>
  <div class="box">Box 3</div>
  <div class="box">Box 4</div>
</div>
</body>
</html>
```

Theme Select:

rubyblue

Click on the following tab to check your answer.

Check Your Answer

In the original desk-top approach structure, the smallest screen size had a background color of lightgreen, with lightyellow in the middle, and lightblue at the largest size. Remember, you will need to rearrange the colors to preserve this code in a mobile-first approach. You will also need to rearrange the Flexbox code to maintain the original layout structures.

Below is one possible answer to this problem:

First, we need to consider the global styles in the presentational code. With a mobile-first approach, we want to place the CSS styling for the smallest screen size here, as these will be the global styles that will be overwritten by the media queries. In our code, the smallest screen size has a background color of 'lightgreen,' so, we'll need to adjust the presentational code.

Next, in this solution, Flexbox is declared outside of the media query. Remember, by default, the divs will have a flex-basis of 100%, so there's no reason to declare this in our code. Therefore, we can remove the .box styling from our global styles.

Now, we define the media queries. Remember that styles in mobile-first are applied cummulatively, so styles will apply to larger screens in a cumulative manner.

So the first media query will be for the next largest breakpoint, corresponding to screens that have a minimum width that is greater than 325px. In other words, we want the background to be 'lightgreen' for screen widths *up to* 325px, so we will add a media query with min-width: 326px to reflect a breakpoint for screens that are widths that are larger than 325px. In order to ensure the layout at this screen size reflects boxes stacked in two rows of two boxes each, we need to add the flex-basis (flex-basis: 46%;) to this media query. Also, we need to add the background color for this breakpoint to be lightyellow.

Lastly, we need to create the media query for the largest screen sizes. Recall in the desktop-first structure this corresponded to screen sizes with a width of greater than 500px. Therefore, we want to set up this media query for screens a minimum width of 501px or greater (min-width: 501px). At this screen size, the desired layout is a horizontal row of four boxes side-by-side. Therefore, we need to add the flex-basis (flex-basis: 19%) to the media query as well. The background color for this media query should be lightblue.

To summarize the mobile-first CSS adjustments:

- Set global style to **background-color: lightgreen;**
- Remove **.box styling** from global styles
- Create media query for screen widths with a **min-width: 326px** and **flex-basis: 46%;**
- Create media query for screen widths with a **min-width: 501px** and **flex-basis: 19%;**

Below is the complete CSS code that reflects a mobile-first approach.

```
/* presentational code below */
   .box {
     font-family: Arial, sans-serif;
     height: 150px;
     border: 4px dotted navy;
     background-color: lightgreen;
     font-size: 3rem;
     text-align: center;
     color: navy;
   }

/* web page layout starts here */

   @media (min-width: 326px) {
     .box {
       flex-basis: 46%;
       background-color: lightyellow;
     }

     .row {
       display: flex;
       flex-flow: row wrap;
       justify-content: space-between;
     }
   }
```

```
@media (min-width: 501px) {
  .box {
    flex-basis: 19%;
    background-color: lightblue;
  }

  .row {
     display: flex;
     flex-flow: row wrap;
     justify-content: space-between;
  }
}
```