CSS Flexible Box Layout

CSS3

What is CSS Flexible Box Layout?

CSS Flexible Box Layout affords us a powerful way to layout our web pages. In using “flex” we are able to decide which direction content should flow, and the sizing/spacing that should be applied. Note in that this method of layout includes only manipulation of content in a single direction, it is considered a one-dimensional layout system.

A core difference between CSS Flexible Box Layout and CSS Grid Layout is that fundamentally:

1. CSS Flexible Box Layout is used to organize and manipulate elements into rows ***or*** columns.
2. CSS Grid Layout is used to organize and manipulate elements within rows ***and*** columns.

If you’re breaking up content horizontally or vertically, the Flexible Box Layout options act as a fast and effective way to accomplish such a layout.

A common use-case might be spacing out links in a navigation—this would be applicable whether the navigation goes across the top of the web page horizontally, or even if these links were to placed in a vertical column down a sidebar. Let’s explore this example in both cases.

Thinking in the CSS Flexible Box Layout Way

When planning out a layout for the web, ensure you stick to common conventions that your users will be familiar with. This ensures a better user experience and makes everything more intuitive.

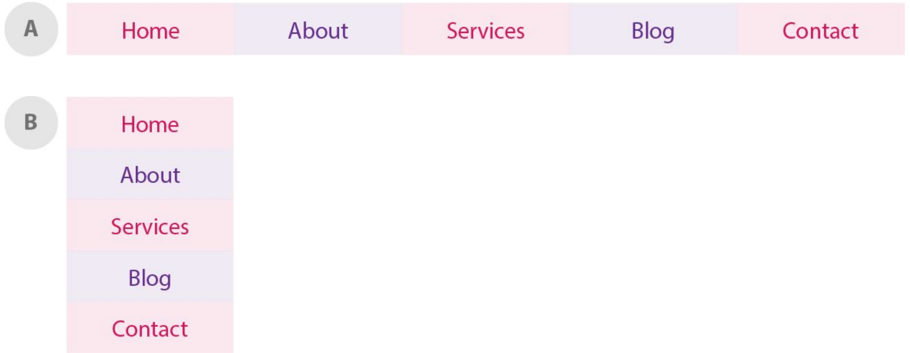
Suppose we have the following pages we want to link to via a navigation on our web site:

- Home
- About
- Services
- Blog
- Contact

Two common ways to lay this information out would be either…

A) Across the top of the web page.
B) In a sidebar or “pull-down” menu.

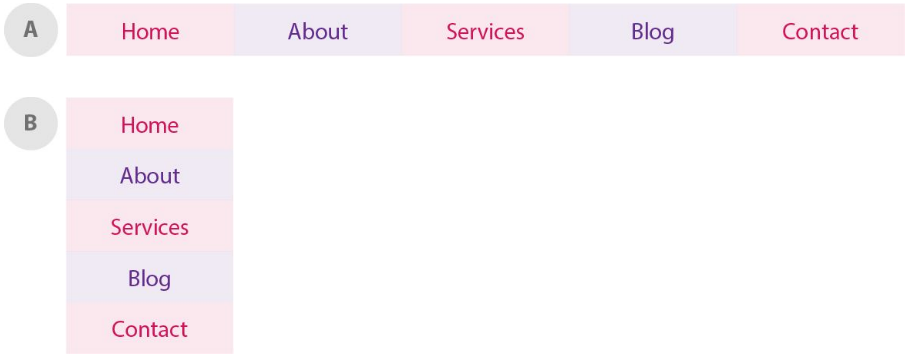
The menu in either approach may look something like the following...



Ensure the Layout is One-Dimensional

Because either of these approaches flow content in a single direction, they are one-dimensional layouts. It is in these layouts that Flexible Box Layout shines!

With simple layouts like this note that CSS Grid Layout—while it can accomplish this task—is not necessary.



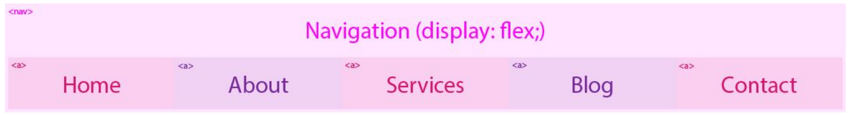
Horizontal Elements

Use of the CSS Flexible Box starts with a container element. This element you must set the following CSS rule:

display: flex;

From there, you can also define the direction, toggle support for additional rows, the horizontal justification, and vertical alignment:

flex-direction: row; /* “row-reverse” exists as well; it reverses the order. */
flex-wrap: nowrap; /* Locks it to one row; else use “wrap”/“wrap-reverse” */
justify-content: space-between; /* Elements evenly spaced horizontally. */
align-items: baseline; /* Elements centred vertically. */



In the case of a navigation, the styles may end up looking like so…

HTML in the <body> of the web page:

```
<nav>
   <a href=”#”>Home</a>
   <a href=”#”>About</a>
   <a href=”#”>Services</a>
   <a href=”#”>Blog</a>
   <a href=”#”>Contact</a>
</nav>
```

CSS in the stylesheet for the web page:

```
nav {
   flex-direction: row;
   flex-wrap: nowrap;
   justify-content: space-between;
   align-items: baseline;
}
```

Vertical Elements

To change to a vertical layout, the same rules can be applied with a simple change to the flex-direction property used…

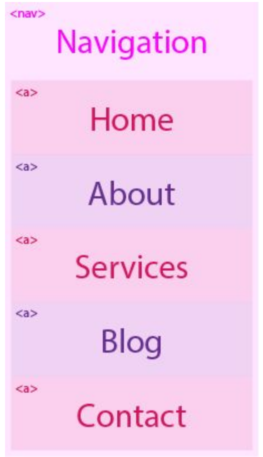
HTML in the <body> of the web page:

```
<nav>
  <a href="#">Home</a>
  <a href="#">About</a>
  <a href="#">Services</a>
  <a href="#">Blog</a>
  <a href="#">Contact</a>
</nav>
```

CSS in the stylesheet for the web page:

```
nav {
  flex-direction: column;
  flex-wrap: nowrap;
  justify-content: space-between;
  align-items: baseline;
}
```

Very quickly you can see that the layout has been altered.



Order

With the CSS Flexible Box Layout we can even have elements display in a different order than they appear in the HTML code.

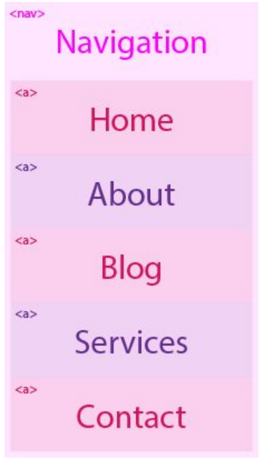
This can be tricky to get your head around, but let’s try it out to see how it works in practice. Recall the HTML we had utilized in the previous example(s):

```
<nav>
   <a href=”#”>Home</a>
   <a href=”#”>About</a>
   <a href=”#”>Services</a>
   <a href=”#”>Blog</a>
   <a href=”#”>Contact</a>
</nav>
```

Say we wanted to display the “Blog” link *before* the “Services” link… we can approach this, in CSS Flexible Box Layout situations, by utilizing the order property.

Try the following…

```
a:nth-child(1) { order: 1; }
a:nth-child(2) { order: 2; }
a:nth-child(3) { order: 4; } /* Move to next slot. */
a:nth-child(4) { order: 3; } /* Move to previous slot. */
a:nth-child(5) { order: 5; }
```



Consider where you might use this. Keep in mind that this can be combined with media queries!

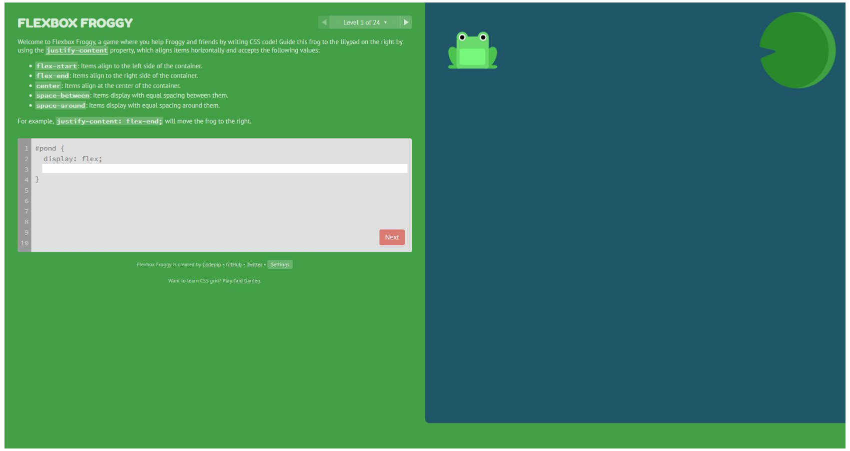
Do Stretches to Stay Flexible

Some of the best resources that exist on CSS Flexible Box Layout to-date include:

- MDN’s CSS Flexible Box Layout
- CSS Tricks’ A Complete Guide to Flexbox
- W3Schools’ CSS Flexbox

There even exists an exceptional interactive practice resource:

- Codepip’s Flexbox Froggy



Welcome to Flexbox Froggy, a game where you help Froggy and friends by writing CSS code! Guide this frog to the lilypad on the right by using the `justify-content` property, which aligns items horizontally and accepts the following values:

- `flex-start`: Items align to the left side of the container.
- `flex-end`: Items align to the right side of the container.
- `center`: Items align at the center of the container.
- `space-between`: Items display with equal spacing between them.
- `space-around`: Items display with equal spacing around them.

For example, `justify-content: flex-end;` will move the frog to the right.

```
#pond {
  display: flex;

}
```

Flexbox Froggy is created by Codepip • GitHub • Twitter • Settings

Want to learn CSS grid? Play Grid Garden.

Recommended Reading

If you'd like to explore CSS' flexible box layout:

- Meyer, E. A; Weyl, E. (October 2017). CSS: The Definitive Guide, 4th Edition. O'Reilly Media, Inc.
    - Chapter 12. Flexible Box Layout