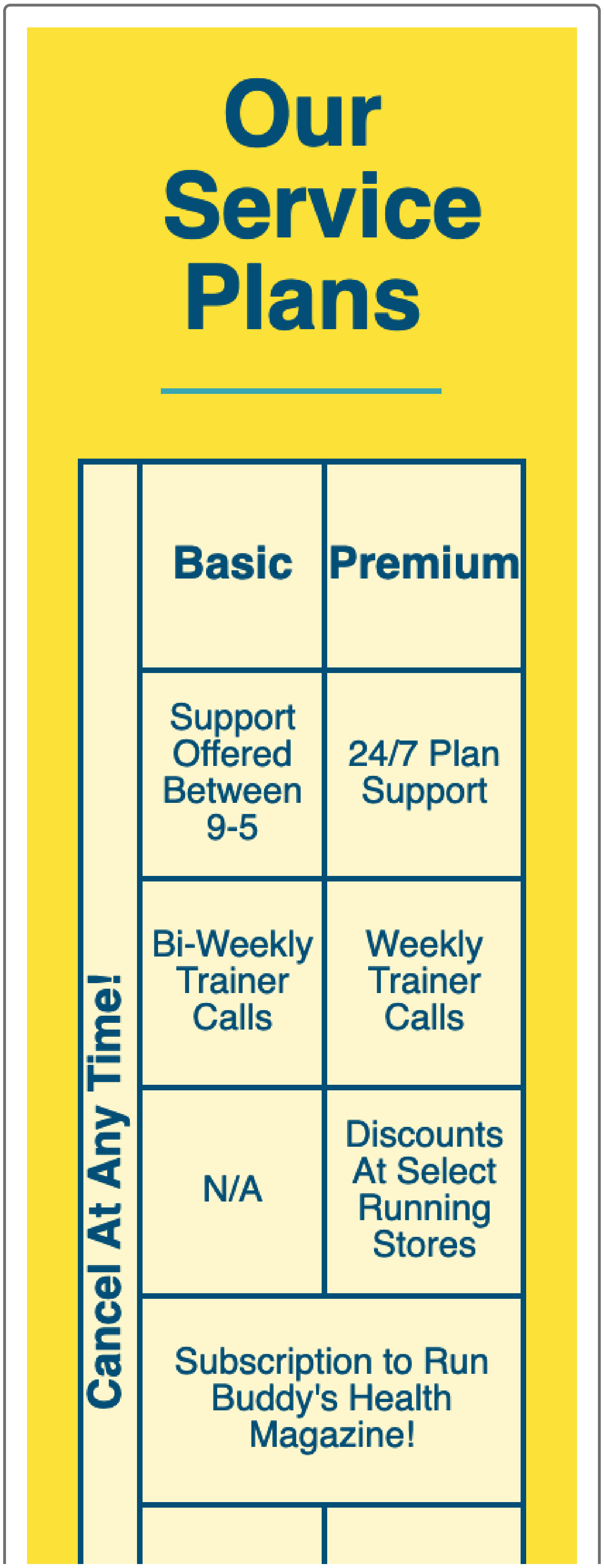
2.4.11 Make the Service Plan Grid Mobile-Responsive

Excellent work! The design team is thrilled with our progress. However, when they view the service plan on a mobile device, some of the layout feels a little compressed:

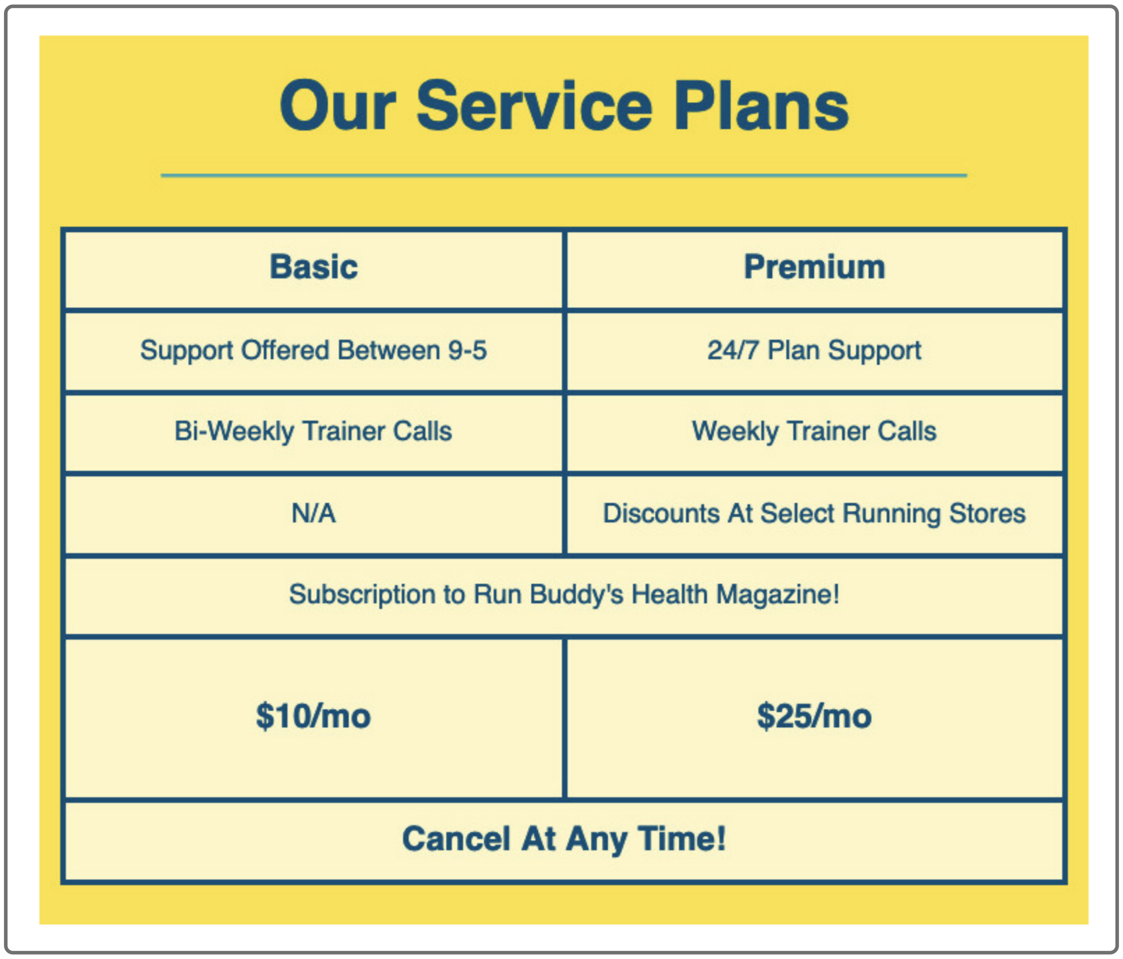


Our Service Plans

Cancel At Any Time!	Basic	Premium
	Support Offered Between 9-5	24/7 Plan Support
	Bi-Weekly Trainer Calls	Weekly Trainer Calls
	N/A	Discounts At Select Running Stores
	Subscription to Run Buddy's Health Magazine!	

To improve the look of the service plan grid on mobile devices, they've requested some changes, such as moving the "Cancel At Any Time" item and changing some font sizes.

Here's a mock-up of the new mobile layout:



Our Service Plans

Basic	Premium
Support Offered Between 9-5	24/7 Plan Support
Bi-Weekly Trainer Calls	Weekly Trainer Calls
N/A	Discounts At Select Running Stores
Subscription to Run Buddy's Health Magazine!	
$10/mo	$25/mo
Cancel At Any Time!	

With our knowledge of media queries and grid positioning, we should be able to make these changes pretty easily. Let's leverage the work that's already been done and use the media query rule for screens less than 768px.

Here are the steps we'll follow:

1. Change the table to two columns.
2. Move the basic column to the first column.
3. Span the Subscription grid item across columns one and two.
4. Remove the vertical font for the "Cancel At Any Time" grid item.
5. Position the "Cancel At Any Time" grid item to be in the last row.

Let's add these rules to the existing media query for screens sizes under 768px.

Try to use some of the things you've learned in this lesson to make these rules yourself:

```
@media screen and (max-width: 768px) {

  .service-grid-container {
    grid-template-columns: 1fr 1fr;
  }

  .service-grid-item.basic {
    grid-column: 1;
  }

  .service-grid-item.both {
    grid-column: 1 / -1;
  }

  .service-grid-item.cancel {
    transform: none;
    writing-mode: unset;
    grid-column: 1 / -1;
    grid-row: -1;
  }
}
```

Let's break down this media query, rule by rule, so we can understand each integral transformation.

Let's start with the parent element rule:

```
.service-grid-container {
  grid-template-columns: 1fr 1fr;
}
```

With CSS Grid and media queries, we can adjust how many rows or columns our grid needs on the fly. By setting the value of `grid-template-columns` to `1fr 1fr`, we created two equal columns for this grid.

Next we set the "basic" plan to the first column by using the following rule:

```
.service-grid-item.basic {
  grid-column: 1;
}
```

We set the basic column as the first column instead of the second. Because we moved all of the basic grid items to start at the first column, the premium grid items naturally flowed into the second column due to implicit grid. Because we transformed our grid, we need the following rule to reposition the spanning row to start at column one:

```
.service-grid-item.both {
  grid-column: 1 / -1;
}
```

The next rule not only turned off the font manipulation but also moved this grid item and transformed it into a row:

```
.service-grid-item.cancel {
  transform: none;
  writing-mode: unset;
  grid-column: 1 / -1;
  grid-row: -1;
}
```

This was necessary to maximize the horizontal screen space on a mobile device screen.

The `grid-column: 1 / -1;` declaration spanned both rows starting at the first column. The `grid-row: -1;` declaration positioned this grid item on the last row. In this rule, we also set the `transform` property to `none` and the `writing-mode` property to `unset` to negate the styles that vertically transform this font.

Congrats! The new feature is now complete. Next, we'll wrap up the branch and merge it.