CSS Positioning Activity

Activity Instructions

Get the HTML and CSS file for the activity

To get the files you need, download the files and save them into your local repository, `positioning.html` and `positionstyles.css`, as part of your project. Save them at the root level (or in other words, not inside any folders.)

Solve the challenges

Modify the CSS to make the boxes in the HTML match the patterns in the images for each activity.

Pay attention to the drop-down 'hints' for each one

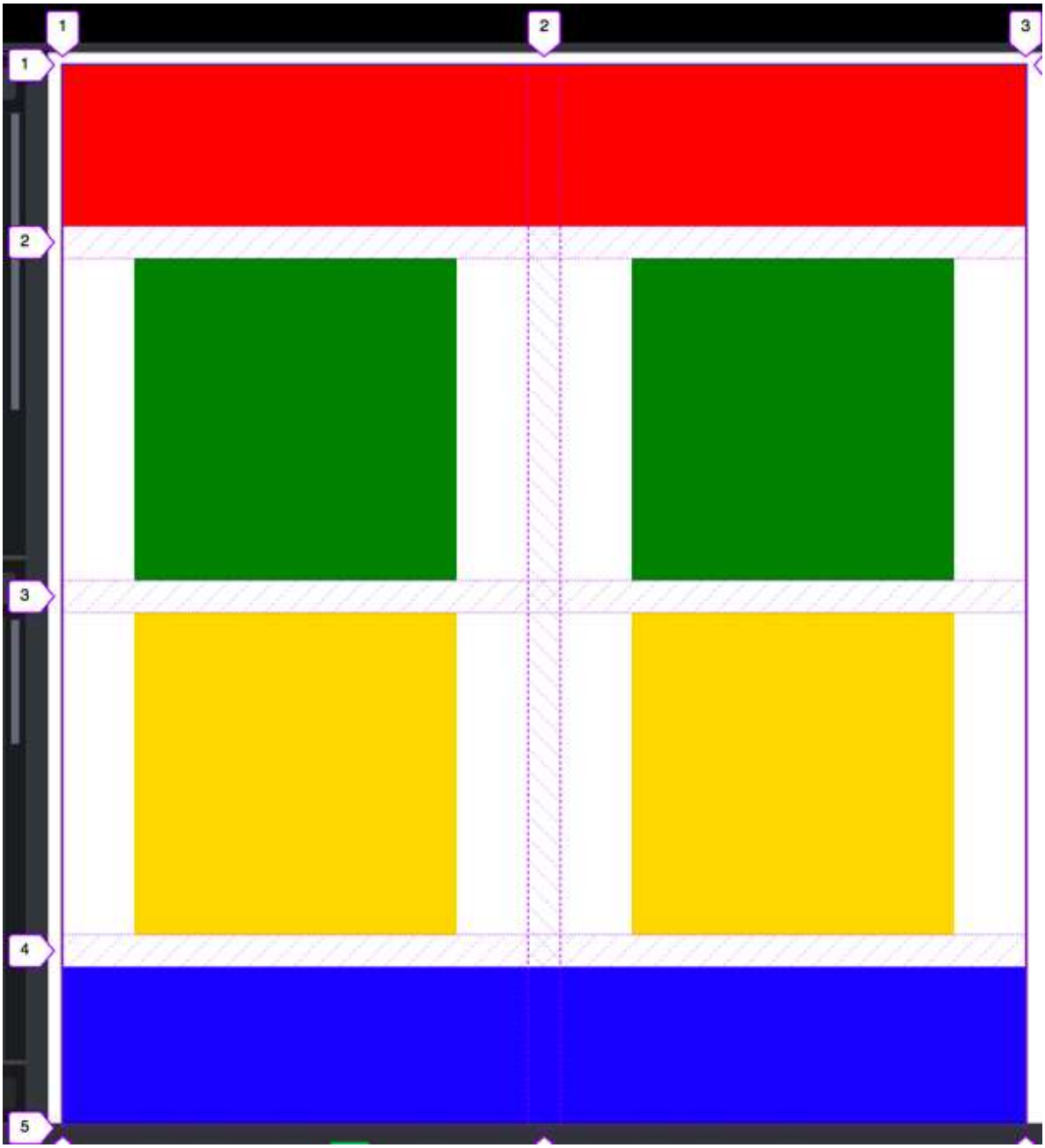
Hint 1

Check out the activity instructions in llearn!

Exercise 1

To help get you started, we will solve exercise 1 together.

Begin by looking at the image indicating what we are trying to do. It should look like the figure below:



Exercise 1

For this first exercise, I've given you a little extra information with the image. It includes the Grid lines so you can easily see how it is structured. We can quickly see that this is a Grid with two columns and four rows. There is also a gap between the rows and columns.

Take a look also at the HTML we have to work with.

```
<div class="content1" >
    <div class="red1" ></div>
    <div class="green1" ></div>
    <div class="green1" ></div>
```

```
    <div class="yellow1"></div>
    <div class="yellow1"></div>
    <div class="blue1"></div>
</div>
```

The first step with using CSS Grid is to identify which elements we are trying to move (these will be the Grid Items or Children) and then find their parent (this will be the Grid container or parent). CSS Grid is turned on at the parent. Once we do that, *all* of the direct children of that element become Grid Items. Because of this, it is helpful if all of the elements we need to position share a common parent.

In this case, we want to move the colored boxes (`red1, green1, yellow1, blue1`), and lucky for us, they all share a common parent already: `content1`

Adjust the box sizes

Before we start our Grid, let's take a moment and make sure the boxes are all the right size and shape. The image above shows that `red` and `blue` should be full width. An easy way to make something full width is to set it's `width` to `100%`. Change the `px` on the existing rules to a `%`.

`green` and `yellow` look to be about twice their current size. So let's change their `height` and `width` to be `200px`.

Turn on CSS Grid

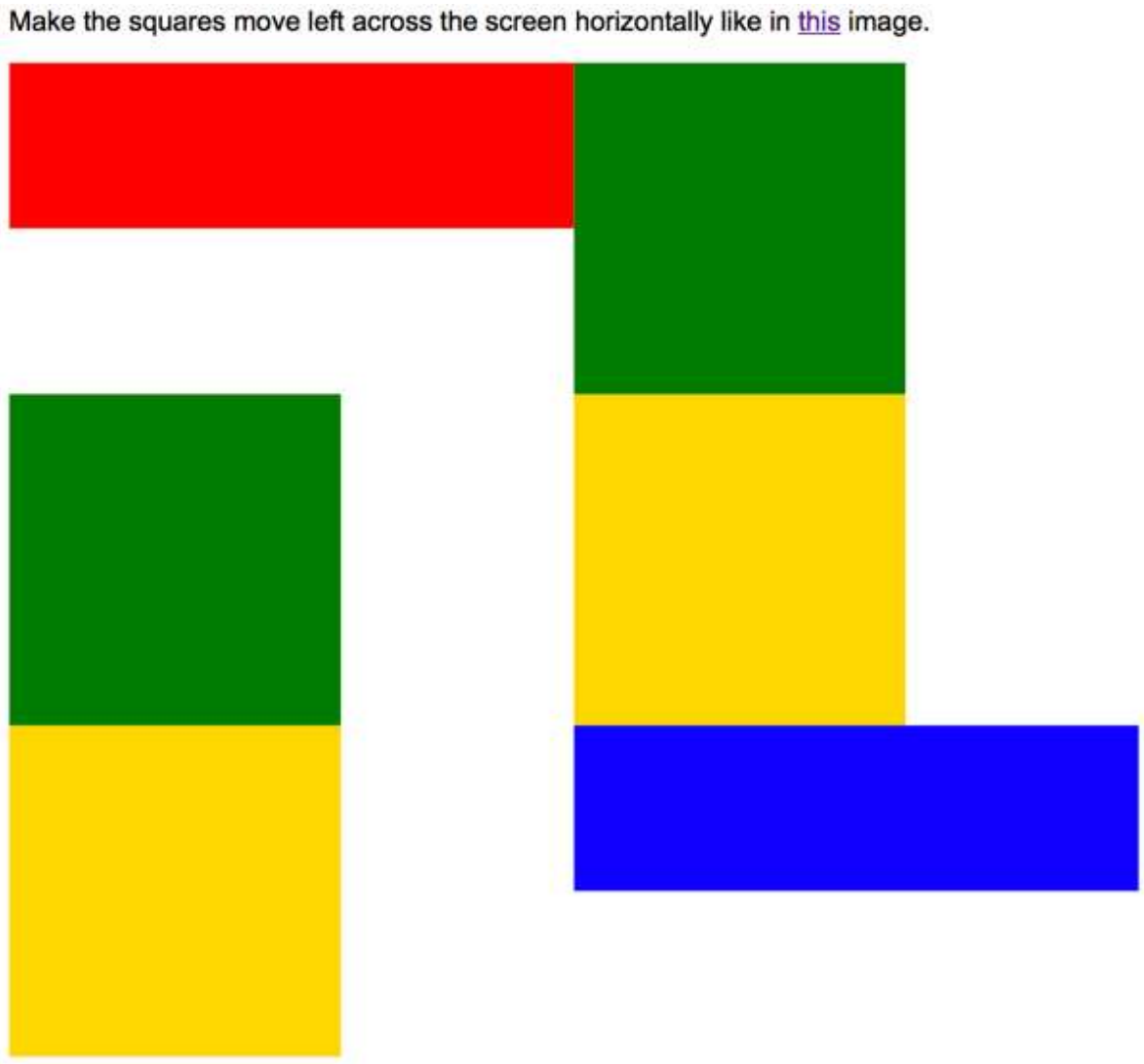
> For the rest of this activity, it will be helpful if you open this link: CSS-Tricks Guide to Grid and keep it handy.

Switch to the CSS file now (`positionstyles.css`) and find the section of code for the first exercise. You will find that there is already a rule defined for `.content1`. Enable Css Grid for that element by adding the line `display: grid;` to that rule.

You will notice that nothing changed! We have to define at least the columns of the Grid we need before anything happens. Let's do that next. If you review the example image again, you will see that we need two equal columns for our grid. That is easy to do with the `fr` unit and the `grid-template-columns` property. If we add that, our rule should look like the following:

```
.content1 {
    /* This is the parent of the activity 1 boxes. */
    display: grid;
    grid-template-columns: 1fr 1fr;
}
```

As soon as you define those columns you should see a change. It should look similar to the figure below:



Exercise 1 halfway there!

It's actually a lot closer to finished than it might appear!

Finishing up

To get our layout to match the image, we really only have three steps left. The first is to tell `red` and `blue` that they should take up *two* columns instead of the default one. And then we need to adjust the horizontal alignment of the yellow and green boxes. Finally, we need to add some space between the columns and rows.

We can use the `grid-column` property to specify how many columns an element should span. Remember that when using this property, we specify the column by describing the grid lines it falls between. If you look at the example image again, you will notice that

there are numbers along the top edge. If we want something in column one, we would place it between line 1 and line 2. If we want something to take up all of columns 1 *and* 2 then we would place it *between* lines 1 and 3! Look at the CSS below to see what this would look like for the red box.

```
.red1 {
    width: 100%;
    height: 100px;
    background-color: red;
    grid-column: 1/3;
}
```

Do the same to the blue box.

Horizontal alignment is controlled with the `justify-` properties. In this case, the specific one we want is `justify-items`. We can add this to our grid container to center the yellow and green boxes. My rule now looks like this:

```
.content1 {
    /* This is the parent of the activity 1 boxes. */
    display: grid;
    grid-template-columns: 1fr 1fr;
    justify-items: center;
}
```

Almost there now! One last item. We are supposed to have some space (called a gap) between our rows and columns. We can use the `gap` property to add this. I think about `20px` should be good. My final CSS for my grid container looks like this:

```
.content1 {
        /* This is the parent of the activity 1 boxes. */
        display: grid;
        grid-template-columns: 1fr 1fr;
        justify-items: center;
```

```
        gap: 20px;
    }
```

Done!

Exercises 2-6

Now complete the rest of the exercises. Help each other out! Here are a few final tips as well.

Tips

Don't overuse `position`. At first, it might seem to be the answer to your positioning prayers, but it can quickly become a nightmare if overused. Try to solve layout problems with other methods first and fallback to position only if they don't work.

Remember that `margin` is a great way to shift things around short distances (and `margin-top: -50px;` would move an element **up** by `50px`).

When using `float` remember that the element that you are floating must come before the other elements in the HTML. Also remember to `clear` your floats when you want things to stop floating.

CSS Grid is a great way to position elements. Most of these can be most easily solved by using it. Remember that to use CSS Grid, first find the elements you want to adjust, then find their parent. Use `display: grid;` on that parent. Also, here are a couple of good resources to help you with CSS Grid: CSS-Tricks Guide to Grid and Grid Garden.

You should be able to solve all of these positioning problems **without** modifying the HTML. If you find yourself tempted to tweak the HTML, you are probably working too hard.

If you are struggling with an activity, start by simply making the boxes the right size and shape. It will often help you to see where to go next.

...and did I mention that you **should not** modify the HTML? I was pretty serious about that.

Publish

Open your `positioning.html` website in your browser to make sure the web page displays correctly. Pay special attention to whether your styles are all displaying. Push all the new files and changes to your GitHub Repo. Once it shows up in the GitHub repo, use that URL to submit. Once verified, submit the URL to your page in I-Learn.

Your URL will be a little longer this time around. You will start with your normal URL, but this time we don't want the index.html file...we want `positioning.html`, so your URL should reflect that. For example, if I normally submitted `https://my-repo.github.io/` as my URL, for this assignment, I would submit `https://my-repo.github.io/`**positioning.html**

Code

Here is the code (both HTML and CSS) that this activity is based off of. If something has happened to the code you got with the setup at the beginning of the semester, you can restore it with this. Create the files as directed below and copy and paste the code.

```
<!DOCTYPE html>
<html lang="en">
    <head>
    <!-- HTML - Name: positioning.html -->
    <title>Positioning Activity</title>
    <link rel="stylesheet" type="text/css" href="positionstyles.css"
    </head>
```

```
<body>
<div class="activity">
    <h2>WDD 130 positioning exercises</h2>
    For each activity match make the boxes match the image by mo
<div class="activity">
    <h2>Activity 1</h2>
    <section class="hint"><input type="checkbox" ><i></i> <h3>Hi
    <p>Follow the <a href="https://byui-wdd.github.io/wdd130/act
    <div class="content1" >
    <div class="red1" ></div>
    <div class="green1" ></div>
    <div class="green1" ></div>
    <div class="yellow1"></div>
    <div class="yellow1"></div>
    <div class="blue1"></div>
    </div>
</div>
<div class="activity">
    <h2>Activity 2</h2>
    <section class="hint"><input type="checkbox" > <h3>Hint 2</h
    <p>Overlap and stagger the squares like in <a href="https://
    <div class="content2" >
    <div class="red2" ></div>
    <div class="green2" ></div>
    <div class="yellow2"></div>
    <div class="blue2"></div>
    </div>
</div>
<div class="activity">
    <h2>Activity 3</h2>
    <section class="hint"><input type="checkbox" > <h3>Hint 3</h
    <li>grid-column:</li>  </ul></div></section>
    Create the 2 column layout like in <a href="https://byui-wdd
    <div class="content3" >
    <div class="red3" ></div>
    <div class="green3" ></div>
    <div class="yellow3"></div>
```

```
    <div class="blue3"></div>
    </div>
</div>
<div class="activity">
    <h2>Activity 4</h2>
    <section class="hint"><input type="checkbox" > <h3>Hint 4</h
    Duplicate the positioning in <a href="https://byui-wdd.githu
    <div class="content4" >
    <div class="red4" ></div>
    <div class="green4" ></div>
    <div class="yellow4"></div>
    <div class="blue4"></div>
    </div>
</div>

<div class="activity">
    <h2>Activity 5</h2>
    <section class="hint"><input type="checkbox" > <h3>Hint 5</h
    Wrap the text around the square like in <a href="https://byu
    <div class="content5" >
    <div class="red5" ></div>
    <div class="green5" ></div>
    <div class="yellow5">Step 01: Continue creating the main pag
        If you have changed your mind on how you want the conten
    <div class="blue5"></div>
    </div>
</div>
<div class="activity">
    <h2>Activity 6</h2>
    <section class="hint"><input type="checkbox" > <h3>Hint 6</h
    Duplicate the positioning in <a href="https://byui-wdd.githu
    <div class="content6" >
    <div class="red6" ></div>
    <div class="green6" ></div>
    <div class="yellow6"></div>
    <div class="blue6"></div>
    </div>
```

```
    </div>
    </body>
</html>
```

```
/* CSS - Name: "positionstyles.css" */
/* Activity 1 styles */
    .content1 {
        /* This is the parent of the activity 1 boxes. */
    }
    .red1 {
    width: 100px;
    height: 100px;
    background-color: red;
    }
    .green1 {
    width: 100px;
    height: 100px;
    background-color: green;
    }
    .yellow1 {
    width: 100px;
    height: 100px;
    background-color: gold;
    }
    .blue1 {
    width: 100px;
    height: 100px;
    background-color: blue;
    }
     /* Activity 2 styles */
    .content2 {
        /* This is the parent of the activity 2 boxes. */
    }
```

```
.red2 {
width: 100px;
height: 100px;
background-color: red;
}
.green2 {
width: 100px;
height: 100px;
background-color: green;
 }
.yellow2 {
width: 100px;
height: 100px;
background-color: gold;
}
.blue2 {
width: 100px;
height: 100px;
background-color: blue;
}
 /* Activity 3 styles */
.content3 {
    /* This is the parent of the activity 3 boxes. */
}
.red3 {
width: 100px;
height: 100px;
background-color: red;
 }
.green3 {
width: 100px;
height: 100px;
background-color: green;
 }
.yellow3 {
width: 100px;
height: 100px;
```

```
background-color: gold;
 }
.blue3 {
width: 100px;
height: 100px;
background-color: blue;
 }
 /* Activity 4 styles */
.content4 {
    /* This is the parent of the activity 4 boxes. */
    height: 400px;
}
.red4 {
width: 100px;
height: 100px;
background-color: red;
 }
.green4 {
width: 100px;
height: 100px;
background-color: green;
}
.yellow4 {
width: 100px;
height: 100px;
background-color: gold;
 }
.blue4 {
width: 100px;
height: 100px;
background-color: blue;
 }
 /* Activity 5 styles */
 .content5 {
    /* This is the parent of the activity 5 boxes. */
     height: 400px;
}
```

```
.red5 {
width: 100px;
height: 100px;
background-color: red;
 }
.green5 {
width: 100px;
height: 100px;
background-color: green;

 }
.yellow5 {
width: 100px;
height: 100px;
background-color: gold;

}
.blue5 {
width: 100px;
height: 100px;
background-color: blue;
 }
 /* Activity 6 styles */
 .content6 {
    /* This is the parent of the activity 6 boxes. */
}
.red6 {
width: 100px;
height: 100px;
background-color: red;
}
.green6 {
width: 100px;
height: 100px;
background-color: green;
 }
.yellow6 {
```

```
    width: 100px;
    height: 100px;
    background-color: gold;
    }
    .blue6 {
    width: 100px;
    height: 100px;
    background-color: blue;
    }


/* Do not make any changes below here */
     .activity {
    width: 70%;
    margin: 20px auto;
    font-family: Arial, sans-serif;
    border: 1px solid black;
    padding: 10px;
    clear:both;
    overflow: auto;
    }

.hint {
    border: 1px solid grey;
    background: #e0e0e0;
    padding: .5em;
    position: relative;
    margin: 1em 0;
}
.hint h3 {
    margin: 0;
}
.hint:hover {
    background: #d0d0d0;
}
.hint > div {
    display: none;
```

```
}

.hint input[type=checkbox] {
    position: absolute;
    width: 100%;
    height: 100%;
    opacity: 0;
    z-index: 1;
    cursor: pointer;
}

.hint input[type=checkbox]:checked ~ div {
    display: block;
}

.hint i {
  position: absolute;
  transform: translate(-6px, 0);
  margin-top: 16px;
  right: 10px;
    top: -3px;
}
.hint i:before, .hint i:after {
  content: "";
  position: absolute;
  background-color: black;
  width: 3px;
  height: 9px;
}
.hint i:before {
  transform: translate(2px, 0) rotate(45deg);
}
.hint i:after {
  transform: translate(-2px, 0) rotate(-45deg);
}

.hint input[type=checkbox]:checked ~ i:before {
```

```
  transform: translate(-2px, 0) rotate(45deg);
}
.hint input[type=checkbox]:checked ~ i:after {
  transform: translate(2px, 0) rotate(-45deg);
}
.hint a {
  position: relative;
  z-index: 1;
}
```