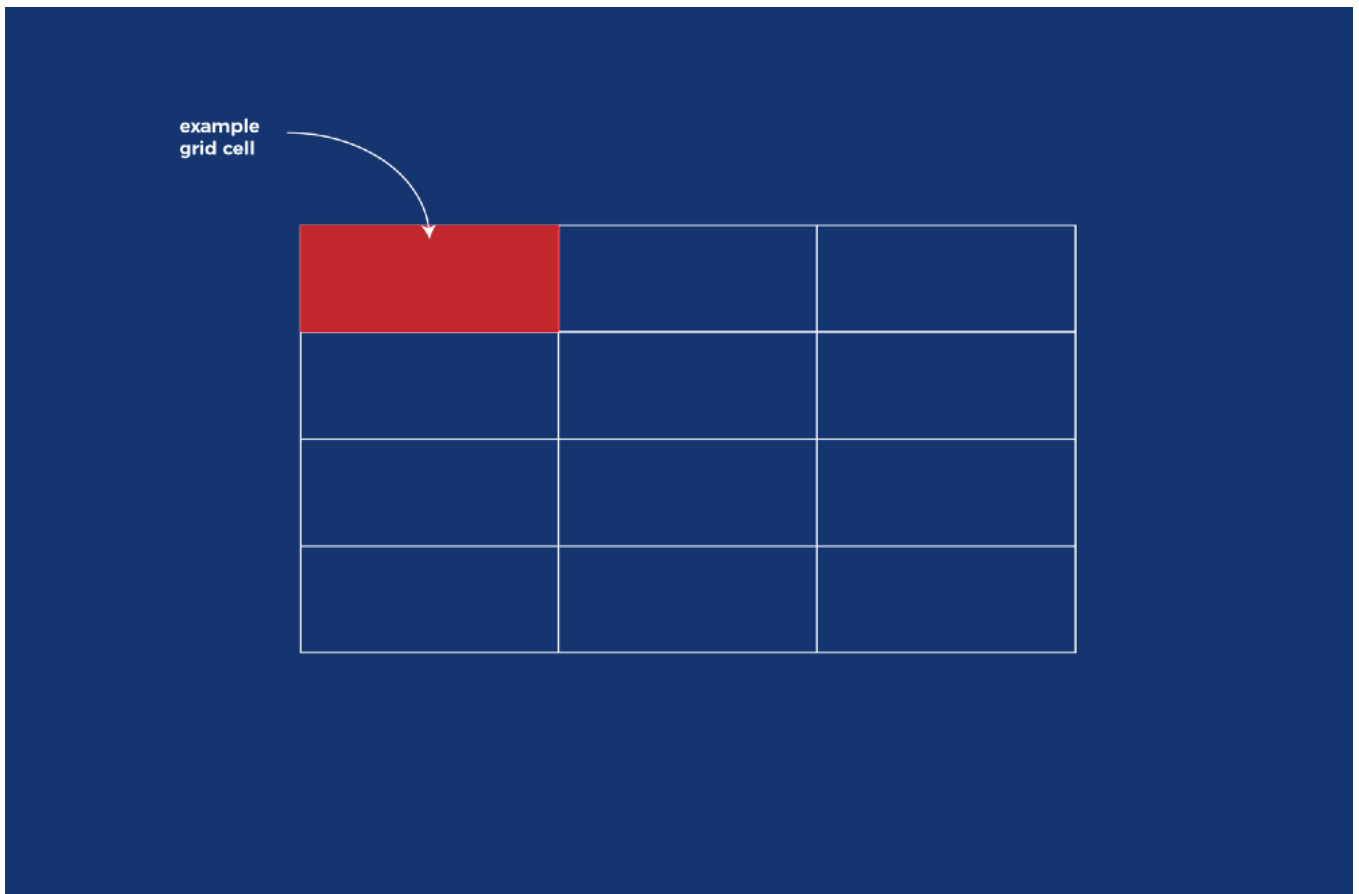# Naming and Positioning Items by Grid Areas

In the previous lesson, we explicitly created the grid system. Now, let's put the grid to use.

The goal of this section is to learn to position grid items using grid areas.

If you recall from an earlier lesson, a grid area is any area bounded by 4 grid lines.

A group of adjacent grid cells such as the entire row span below, may also account for a grid area.

How do you start using grid areas?

The logical place to begin is naming grid areas.

Let me explain that.

Consider the code block below:

```html
<div class="aside"></div>
<div class="main"></div>
<div class="footer"></div>
```

3 divs—simple. By the way, It is semantically better to use aside main and footer tags. I'll just keep things simple.

Now look at this:

```css
.main {
  grid-area: content;
}
.footer {
  grid-area: footer;
}
.aside {
  grid-area: sidebar;
}
```

What is happening there?

If you write a bit of Javascript, or any programming language, then the concepts of variables isn't new to you.

In Javascript, we could say:

```javascript
var gridArea = "content"
```

What is done above is like saying, 'save the string, `content` into the variable **gridArea**

What the CSS declarations above does is quite similar.

Every grid item can be assigned to an area within the grid container. Read that again 😆

> Every grid item can be assigned to an area within the grid container

However, before doing that, it is imperative to attach the grid items to an area name. Like you assign variables in Javascript—sort of.

```css
.main {
  grid-area: content;
}
.footer {
  grid-area: footer;
}
.aside {
  grid-area: sidebar;
}
```

The code block above says, let the `.main` element have an area name of `content`. Let the `.footer` element have an area name of `footer`. Finally, let the `.aside` class have a grid name of `sidebar`

Now the grid items have all been assigned area names.

In programming, variables are set to be used elsewhere. Let's make use of the grid area names in the next lesson.