# CSS Grid: Architecture Blog

In this project, you’ll follow step-by-step instructions to build an avant-garde website using the CSS Grid Layout.

You can view a live version of this project [here](https://s3.amazonaws.com/codecademy-content/courses/learn-css-grid/project-iii/index.html).

Look over **index.html** to review the different HTML elements you have to work with, then navigate to **style.css**.

Notice that each section of the website has 3 elements — an image, a title, and a subtitle, and they go in alphabetical order, so image a goes with title b and subtitle c, and so forth.

If you want your final product to match ours exactly, follow the steps below very carefully. However, this is your project! You should feel free to experiment with placing the overlapping elements wherever you wish so that you can get more practice with CSS Grid Layout.

If you get stuck during this project or would like to see an experienced developer work through it, click “**Get Help**“ to see a **project walkthrough video**.

Start

**Tasks**

**14/14Complete**

Mark the tasks as complete by checking them off

## Setup

**1.**

As you can see in your **index.html** file, we are using an element with the class grid to hold all the elements we want to place on our CSS grid. In the .grid rule set in **style.css**, set the value of the display property to grid.

Hint

.grid {

display: grid;

}

**2.**

Use CSS to set the width of your grid to 900px and the height to 3000px.

**3.**

Set the margin property of your grid to auto.

Hint

margin: auto;

## Create Grid Layout

**4.**

Use the CSS grid-template property to make your grid 30 equal rows and 12 equal columns.

Remember, when using grid-template, the values before the slash will determine the size of each row. The values after the slash determine the size of each column.

Hint

Repeat is particularly useful with fr. For example, repeat(5, 1fr) would split your table into five equal rows or columns. Set the grid-template property to:

repeat(30, 1fr) / repeat(12, 1fr)

**5.**

It’’s hard to see what is going on with your grid because the images are too big. To make the images conform to the grid we can set the max-width property of all images by creating a rule set for img elements in **style.css** and adding a max-width declaration to it.

Set the max-width of all images in your project to be 100%.

It’s okay if the images disappear; they’ll come back in a later step.

Hint

img {

max-width: 100%;

}

**6.**

Add the following line of code to your style.css:

.d, .e, .f, .g, .h, .i, .j, .k, .l, .m, .n, .o {

display: none;

}

This will hide the elements that compose sections 2-5 so that you can focus on the positioning of the first section of the website.

## Laying The Foundation

**7.**

Identify the three elements that make up the first section of the website which is titled “Repurposed Living.”

The elements that make up this section are created in the **index.html** file shown here:

<div class="box a">

<img src="https://s3.amazonaws.com/codecademy-content/courses/learn-css-grid/project-iii/media/pexels-big.jpg">

</div>

<div class="box b title">

Repurposed Living

</div>

<div class="box c">

Bassicarella Architectes, Hamburg, Germany Repurposed Housing

</div>

**8.**

In **style.css**, find the rule for the element that is holding the first image, which has a class of a.

Use the short-hand the property grid-area and set its value so the element starts at row line 1 and column line 3 and ends at row line 5 and column line 13.

Hint

.a {

grid-area: 1 / 3 / 5 / 13;

}

**9.**

Now, finish the first section of the website by positioning elements with the class b and c so that they overlap the element with the class a, which you positioned in the previous step.

Hint

One possible way to position these elements is below:

.b {

grid-area: 2 / 2 / 3 / 4;

}

.c {

grid-area: 4 / 1 / 5 / 4;

}

## Repositioning Overlapping Elements

**10.**

Delete the following line of code from **style.css**:

.d, .e, .f, .g, .h, .i, .j, .k, .l, .m, .n, .o {

display: none;

}

**11.**

Add the line of code below which will make all the elements except the images invisible:

.e, .f, .h, .i, .k, .l, .n, .o {

display: none;

}

**12.**

Position each of the image elements so that they are slightly overlapping the previous one. The classes for the image elements are: a, d, g, j, m.

For reference you can view a live version of this project [here](https://s3.amazonaws.com/codecademy-content/courses/learn-css-grid/project-iii/index.html).

Hint

.a {

grid-area: 1 / 3 / 5 / 13;

}

.d {

grid-area: 5 / 1 / 12 / 7;

}

.g {

grid-area: 11 / 3 / 16 / 13;

}

.j {

grid-area: 15 / 1/ 22 / 7;

}

.m {

grid-area: 21 / 3 / 27 / 13;

}

**13.**

Remove the following line of code from your **style.css**:

.e, .f, .h, .i, .k, .l, .n, .o {

display: none;

}

**14.**

Position the rest of the title and subtitle elements relative to the image element for each section like we did with elements a, b, and c.

If you would like additional support you can use the hint code as a template, and then make changes to style to your liking.

Hint

.d {

grid-area: 5 / 1 / 12/ 7;

}

.e {

grid-area: 7 / 6 / 9 / 10;

}

.f {

grid-area: 9 / 7 / 10 / 10;

}

.g {

grid-area: 11 / 3 / 16 / 13;

}

.h {

grid-area: 12 / 2 / 14 / 5;

}

.i {

grid-area: 14 / 1 / 15 / 4;

}

.j {

grid-area: 15 / 1 / 22 / 7;

}

.k {

grid-area: 17 / 6 / 19 / 10;

}

.l {

grid-area: 19 / 7 / 20 / 9;

}

.m {

grid-area: 21 / 3 / 27 / 13;

}

.n {

grid-area: 22 / 2 / 24 / 5;

}

.o {

grid-area: 24 / 1 / 26 / 4;

}