UNIVERSITY of GREENWICH

CONTENT MANAGEMENT

CSS GRID LAYOUT

BASIC CONCEPTS OF GRID LAYOUT

MA Web Design and Content Planning
Vanessa A Costa
Feb 2018

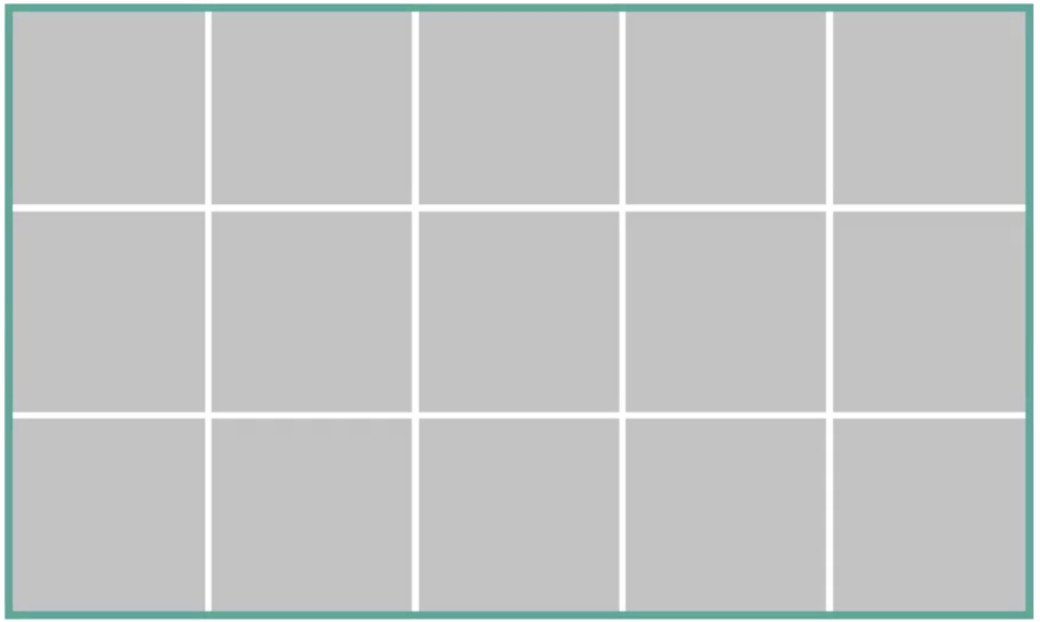
CSS GRID LAYOUT

- Introduces a two-dimensional grid system to CSS;
- The grid is the intersecting set of horizontal and vertical lines, where elements are placed respecting these column and row lines;
- It can be used to layout major page areas or smaller user interface elements;
- After defining a grid on the parent element, all direct children become grid items;
- Grid is a powerful specification and when combined with other parts of CSS such as flexbox, can help you to create layouts that were previously impossible to build in CSS.

*It all starts by creating a grid in your **grid container**.*

TERMINOLOGY

The Grid



Grid Container

To create a *grid container just declare*

display: grid or display: inline-grid on an element.

All *direct children* of that element will become *grid items*.

```html
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

```css
.wrapper {
  display: grid;
}
```

One

Two

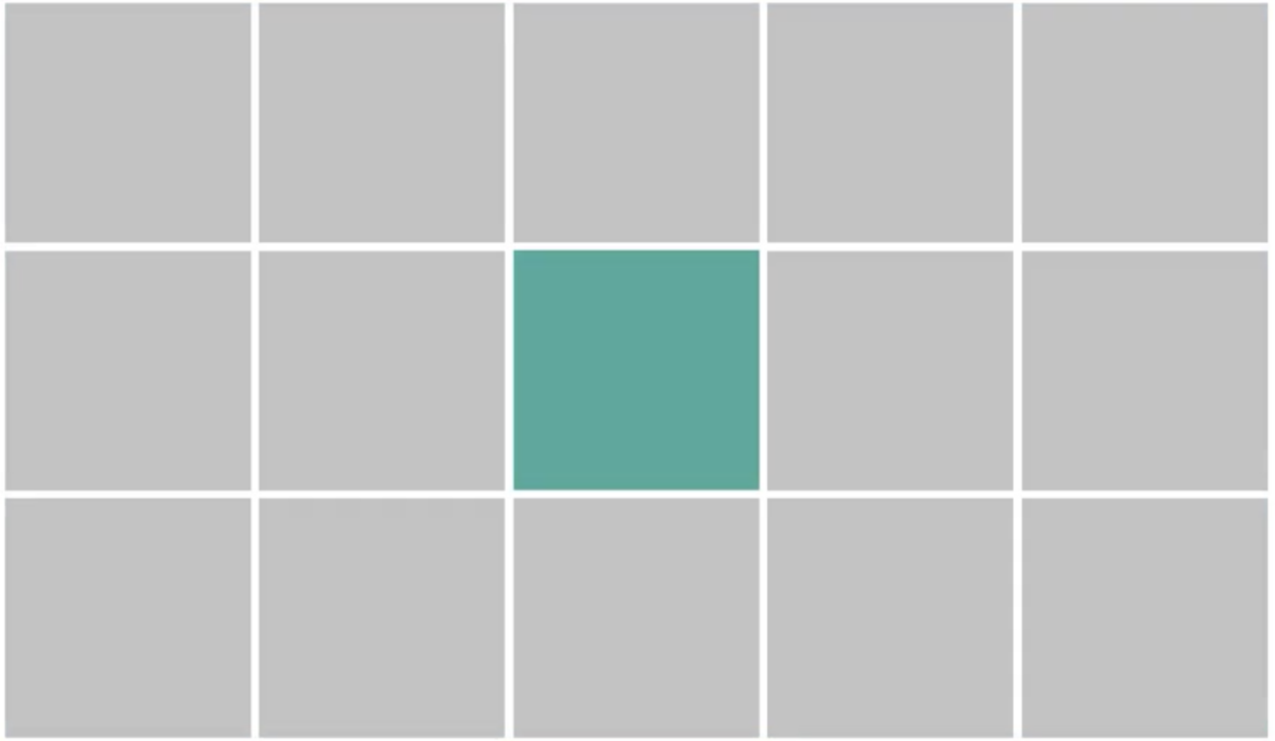
Three

Four

Five

Grid Cells

Is the smallest unit on a grid. Conceptually it is like a table cell. When the grid is defined as a parent the child items will lay themselves out in one cell each of the defined grid.



Grid Tracks

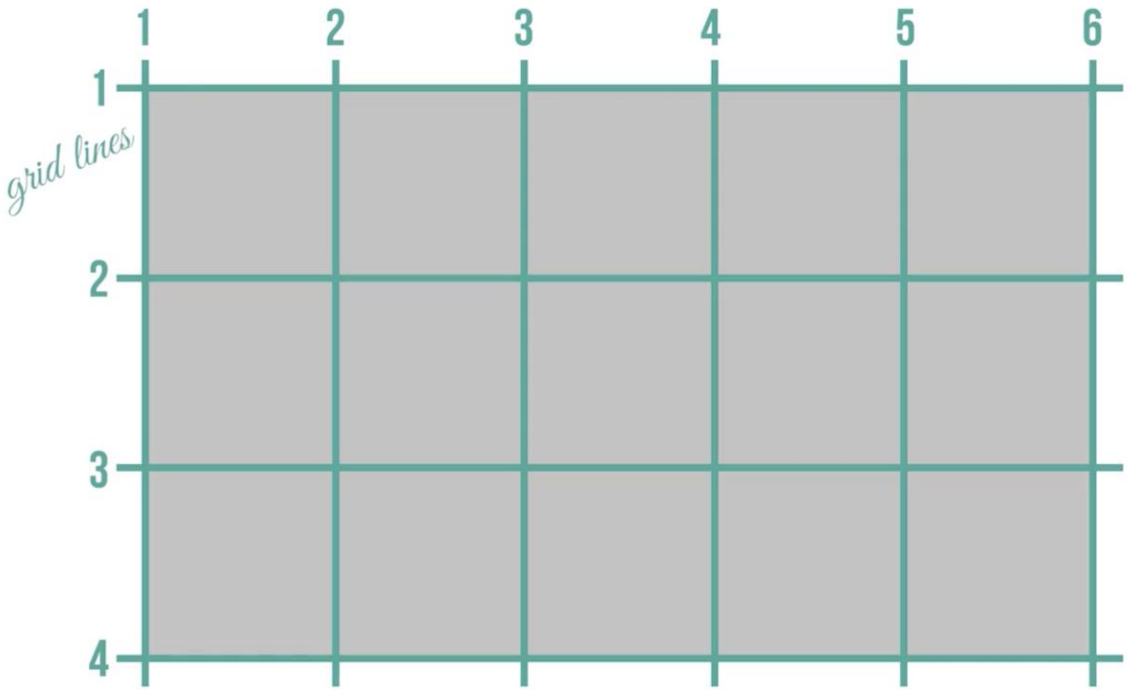
To define rows and columns on the grid, it is used the grid-template-columns and the grid-template-rows properties. These define grid tracks. A *grid track* is the space between any two lines on the grid. There are column tracks and row tracks.

Grid Lines

When we define a grid we define the grid tracks.

Grid then gives us numbered lines to use when positioning items.

In our five column, three row grid we have six column lines and 4 row lines (number of columns or rows +1).

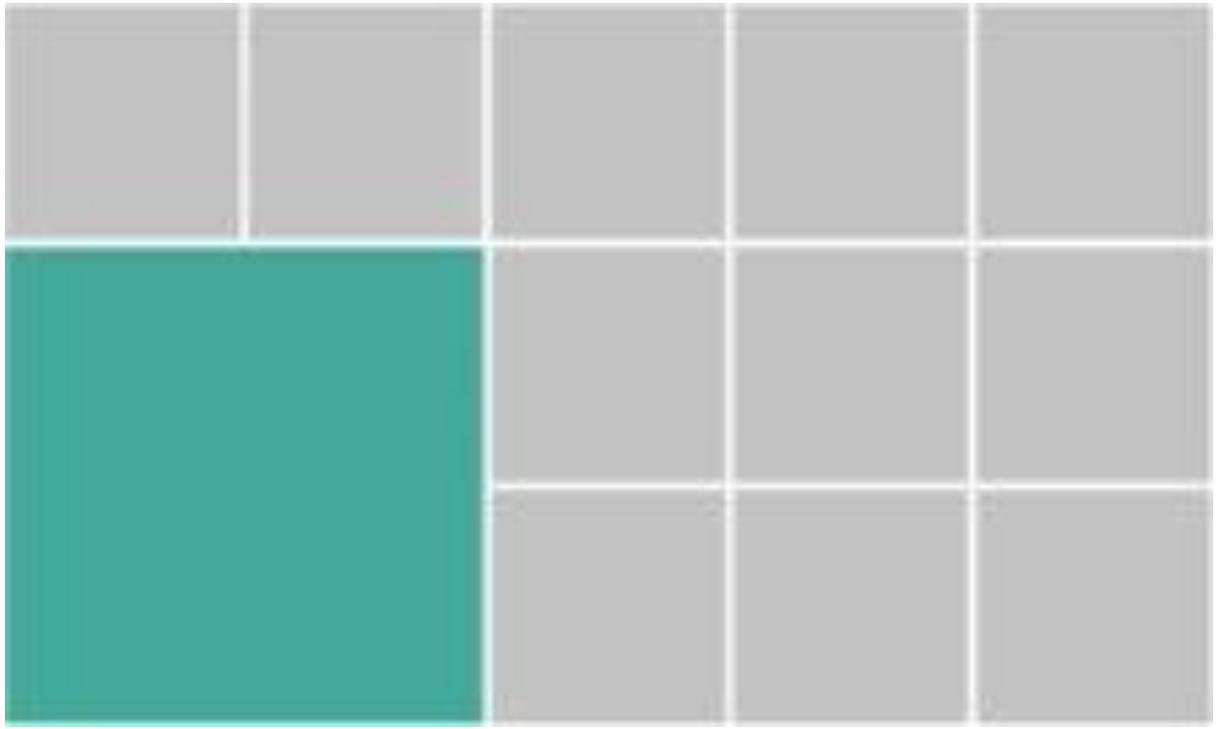


Grid Areas

Items can span one or more cells both by row or by column, and this creates a grid area.

Grid areas must be rectangular.

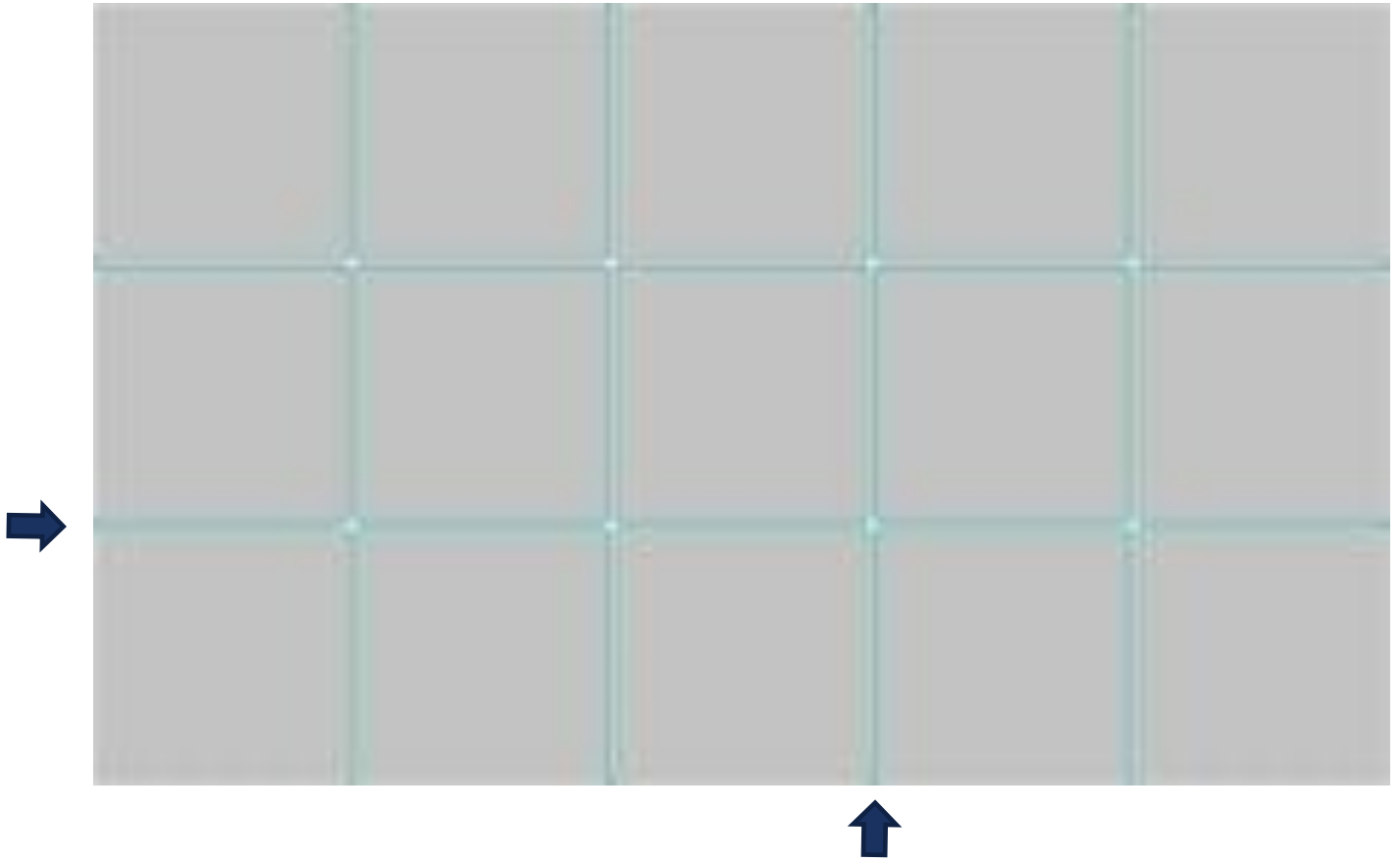
Example: grid area spans two row and two column tracks.



Gutters (grid-gap)

Gutters or *alleys* between grid cells can be created using the shorthand grid-gap, or grid-column-gap and grid-row-gap properties.

grid-column-gap: 10px;
grid-row-gap: 1em;
grid-gap:10px (shortcut for both)



The fr Unit

Fixed and Flexible Track Sizes

- A grid can be created with fixed track sizes, using pixels for example. This sets the grid to the specified pixel which fits to the desired layout;
- But is also possible to create a grid using flexible sizes with percentages or with the new fr unit designed for this purpose, the creation of flexible grid tracks;
- The unit fr – fraction of the available space in the grid container;
- Using the fr unit allows to create flexible grids, which is great for a responsive web;
- The fr unit grows and shrinks in proportion of the available space;
- It can be combined with fixed sizes;

```html
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

```css
.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

Track Listings with Repeat () Notation

In case of a grid with many tracks use the repeat() notation, to repeat all or a section of the track listing.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

3 columns each with 1 fraction of the available space

```
.wrapper {
  display: grid;
  grid-template-columns: 20px repeat(6, 1fr) 20px;
}
```

8 columns: 1st with 20 px, 6 x 1fr , 8th with 20px

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(5, 1fr 2fr);
}
```

10 columns: 5 x (1 column 1 fr and 1 column 2fr)

The Implicit and Explicit Grid

Explicit Grid

Is the grid which you define number of columns and rows with grid-template-columns and grid-template-rows.

Implicit Grid

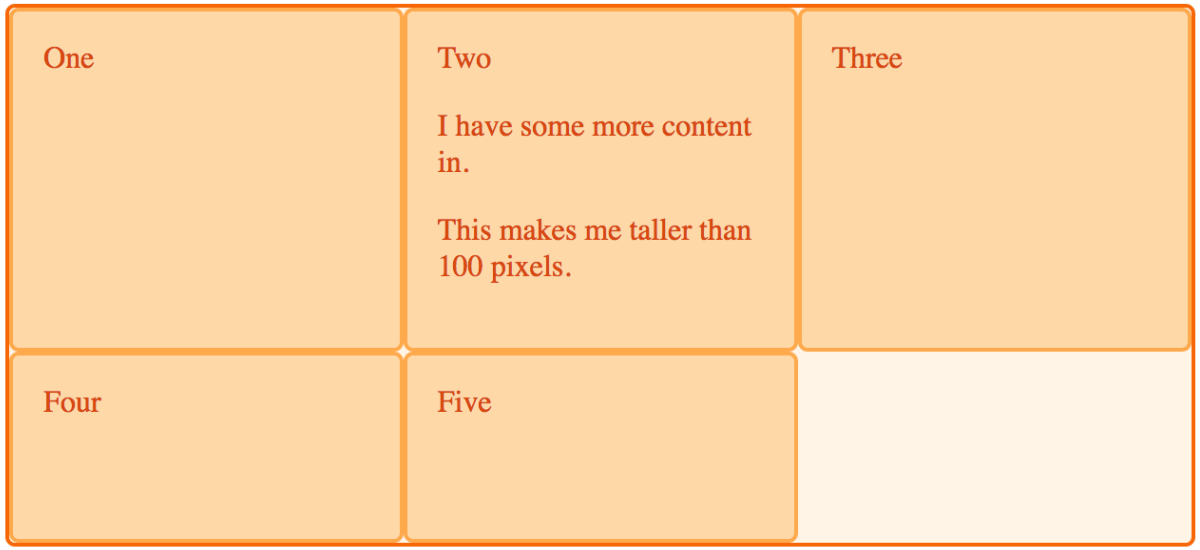
- When you create a grid defining just the column tracks with the grid-template-columns property, the grid will create automatically rows for content.
- If you place something outside of that defined grid, or due to the amount of content more grid tracks are needed, then the grid creates rows and columns. These tracks will be auto-sized by default, resulting in their size being based on the content that is inside them.
- But you can also define a set size for tracks created in the implicit grid with the grid-auto-rows and grid-auto-columns properties.

Track Sizing and Minmax()

When you are using an explicit grid or defining the size for automatically creating rows or columns and want to give tracks a minimum size, but also ensure they expand to fit any content that is added, use the minmax() function.

```css
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: minmax(100px, auto);
}
```

```html
<div class="wrapper">
  <div>One</div>
  <div>Two
    <p>I have some more content in.</p>
    <p>This makes me taller than 100 pixels.</p>
  </div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```



Example:

Automatically created rows will be a minimum of 100 pixels tall, and a maximum of auto. Using auto means that the size will look at the content size and will stretch to give space for the tallest item in a cell, in this row.

Positioning Items Against Lines

Remember the grid lines...that is our target when we want to place items!

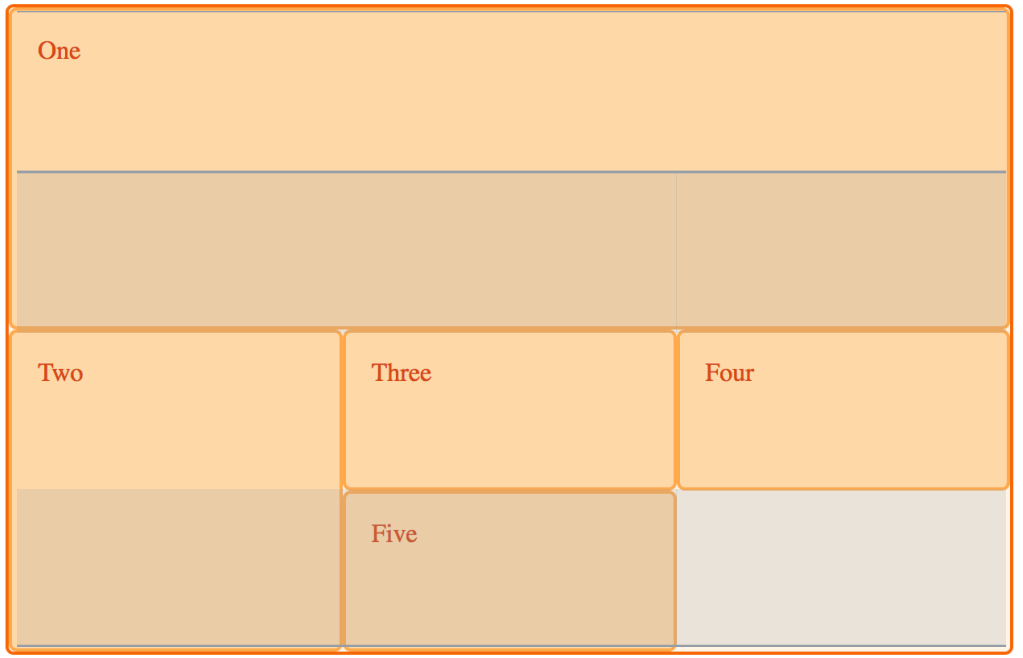
- The properties we need: grid-column-start, grid-column-end, grid-row-start, grid-row-end;
- Items can me placed into a precise location on the grid using line numbers, names or by targeting an area of the grid.

```html
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
  <div class="box4">Four</div>
  <div class="box5">Five</div>
</div>
```

```css
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}

.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}

.box2 {
  grid-column-start: 1;
  grid-row-start: 3;
  grid-row-end: 5;
}
```



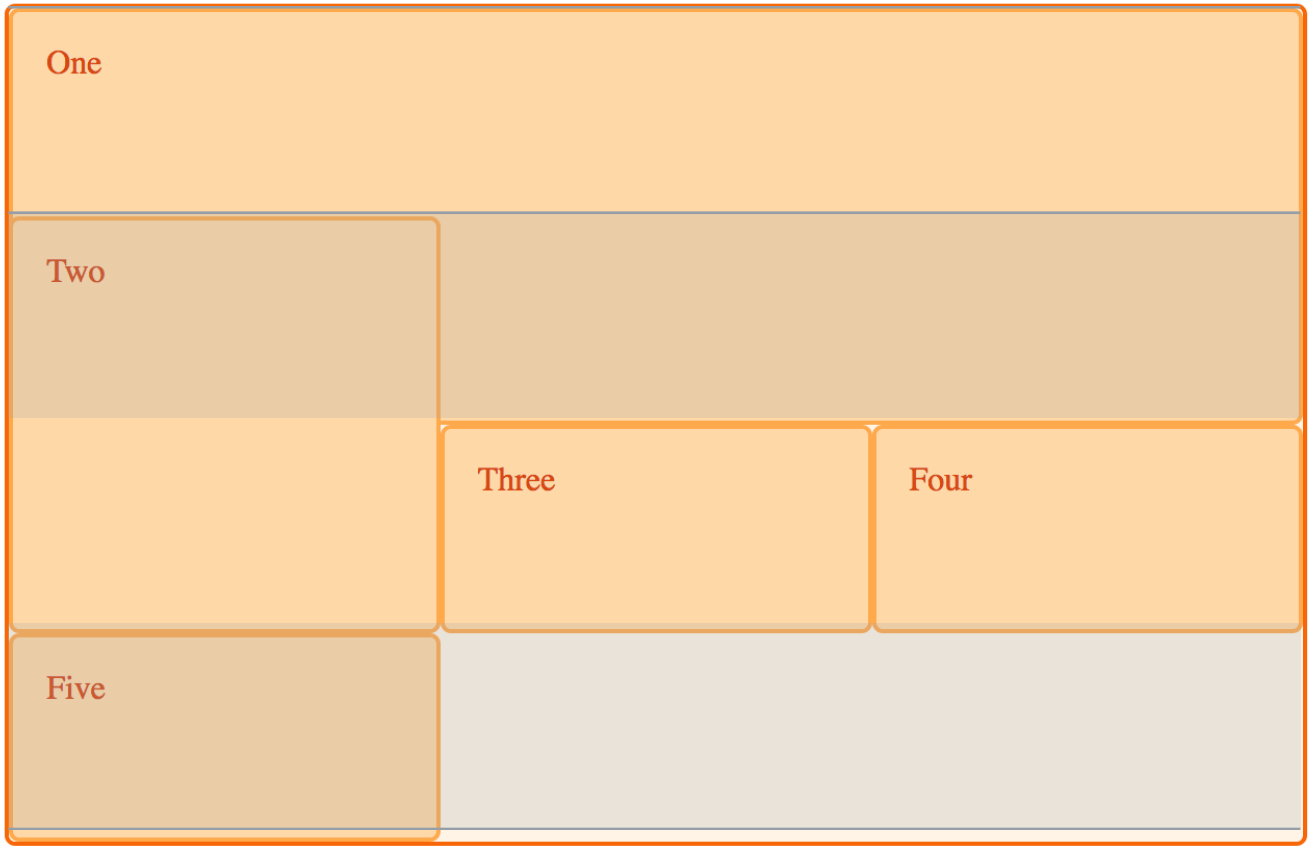
Layering Items with Z-Index

- Grid items can occupy the same cell.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}

.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}

.box2 {
  grid-column-start: 1;
  grid-row-start: 2;
  grid-row-end: 4;
}
```



box2 is now overlapping box1, it displays on top as it comes later in the source order.

THIS PROPERTY ALLOWS US TO MAKES DESIGNED LAYOUTS LIKE THIS:

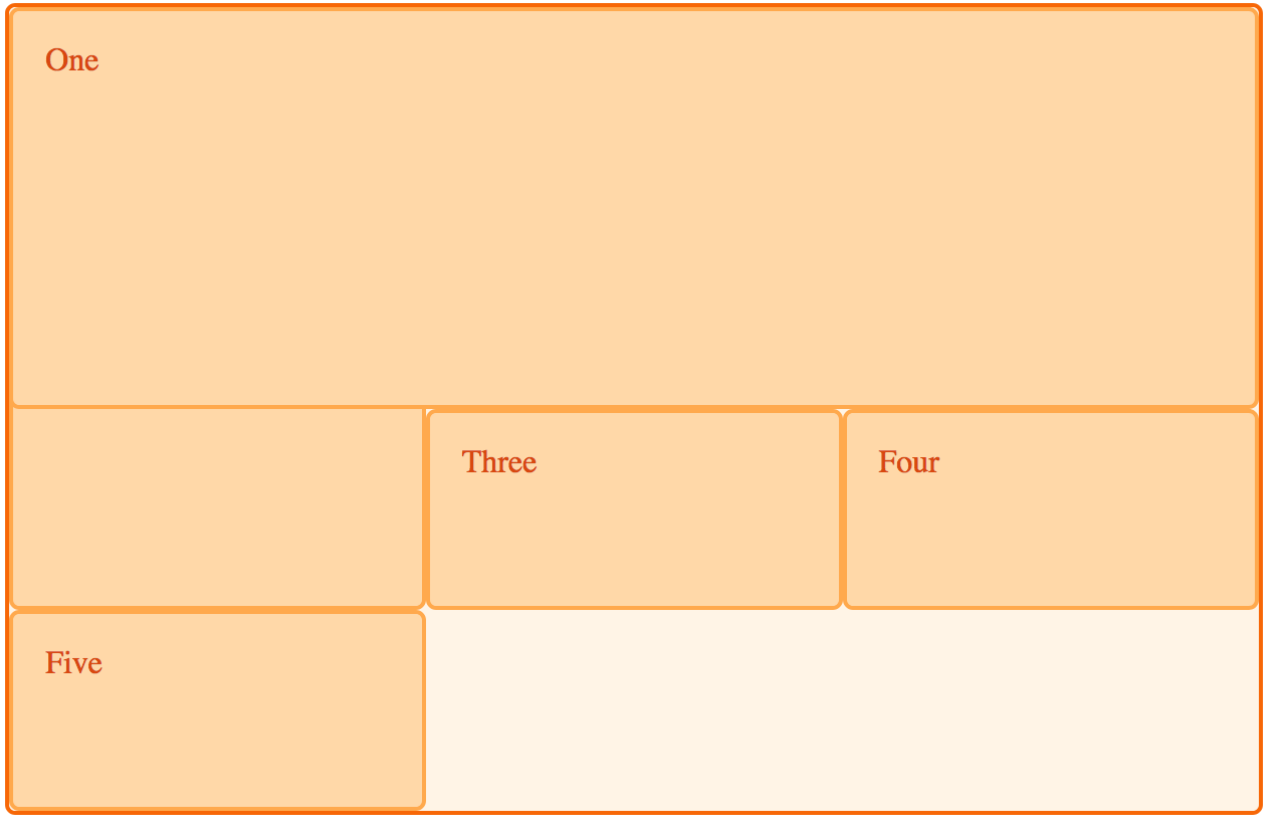
Controlling the Order

- We can control the order in which items stack up by using the z-index property - just like positioned items. If we give box2 a lower z-index than box1 it will display below box1 in the stack.

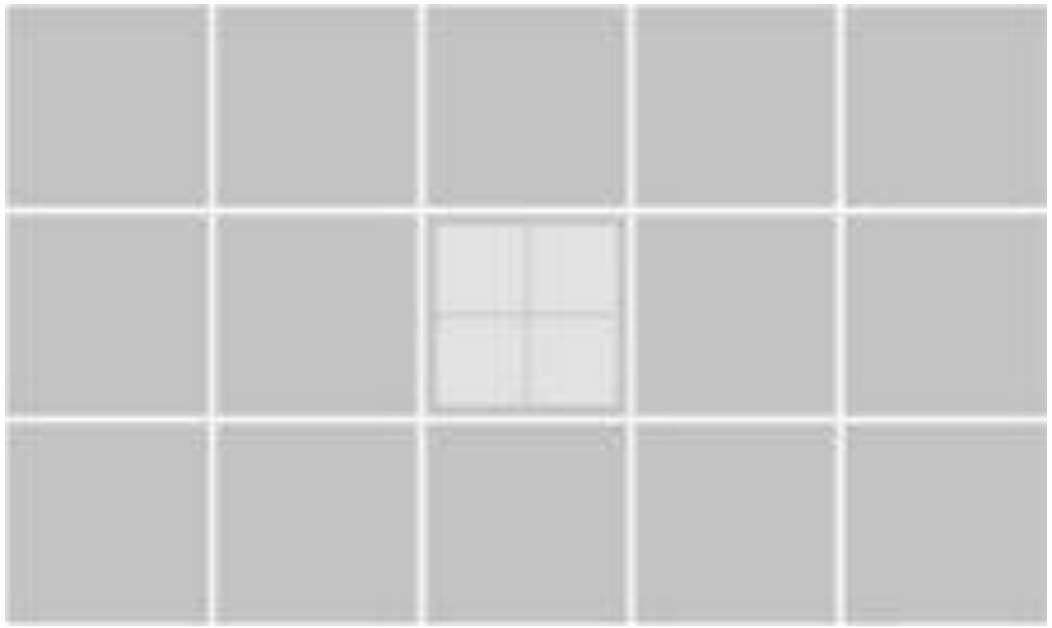
```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}

.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
  z-index: 2;
}

.box2 {
  grid-column-start: 1;
  grid-row-start: 2;
  grid-row-end: 4;
  z-index: 1;
}
```



NESTING GRIDS



- A grid item can become a grid container...
- The nested grid has no relationship to the parent.
- The grid-gap of the parent and the lines in the nested grid do not align

FLEXBOX OR CSS GRID

Learn both and use accordingly to your goal!

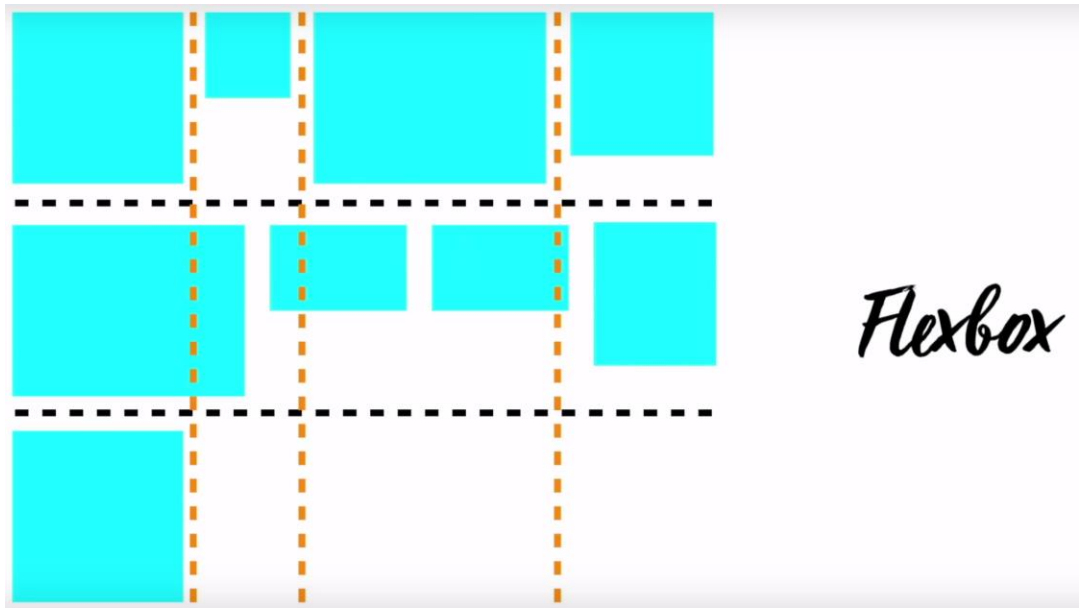
When to use what...depends in what you want to do.

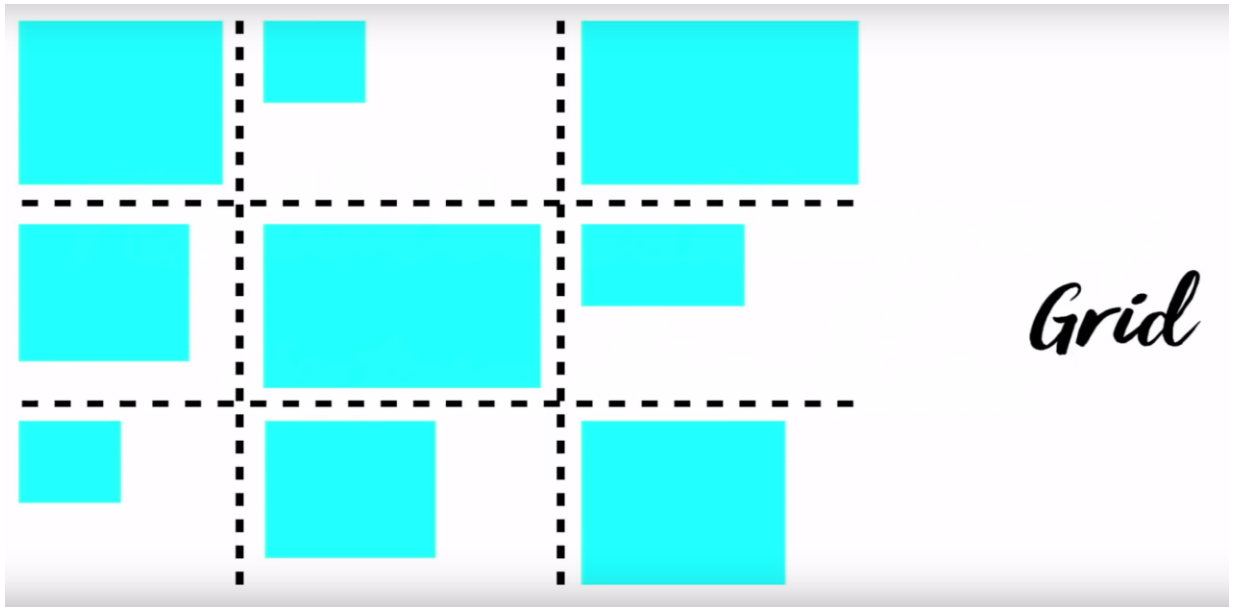
Flexbox

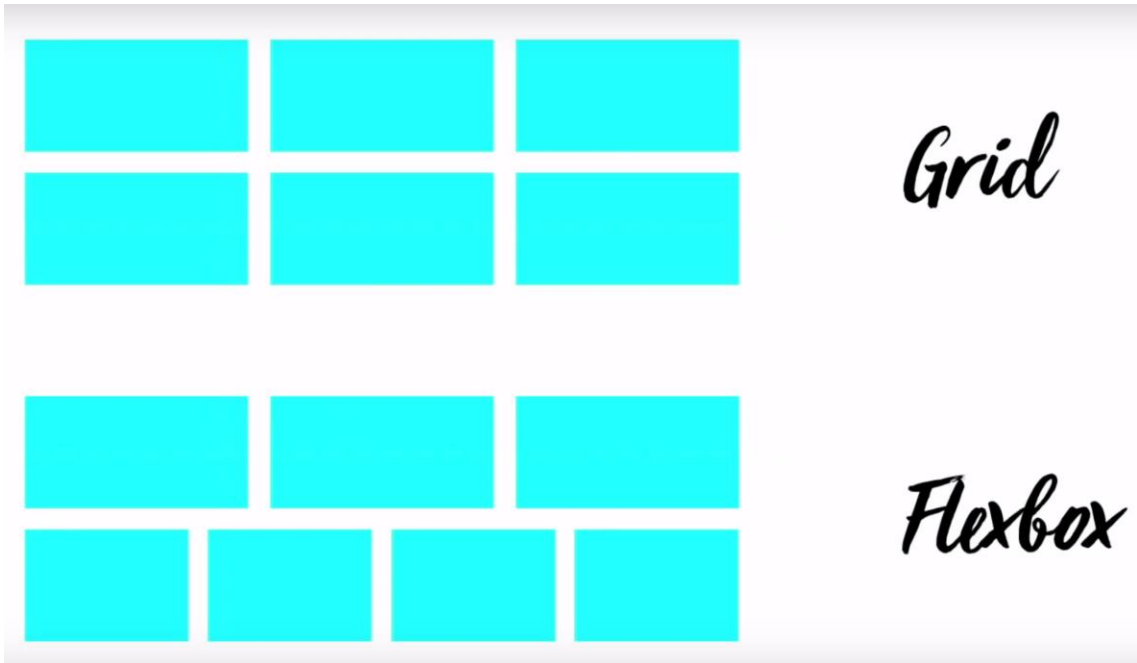
- Unidimensional (row or column)
- Starts from the content defining the space-content out;
- Calculations done in each row, one at the time, with no regard to the other rows;
- Things don't line up;

CSS Grid

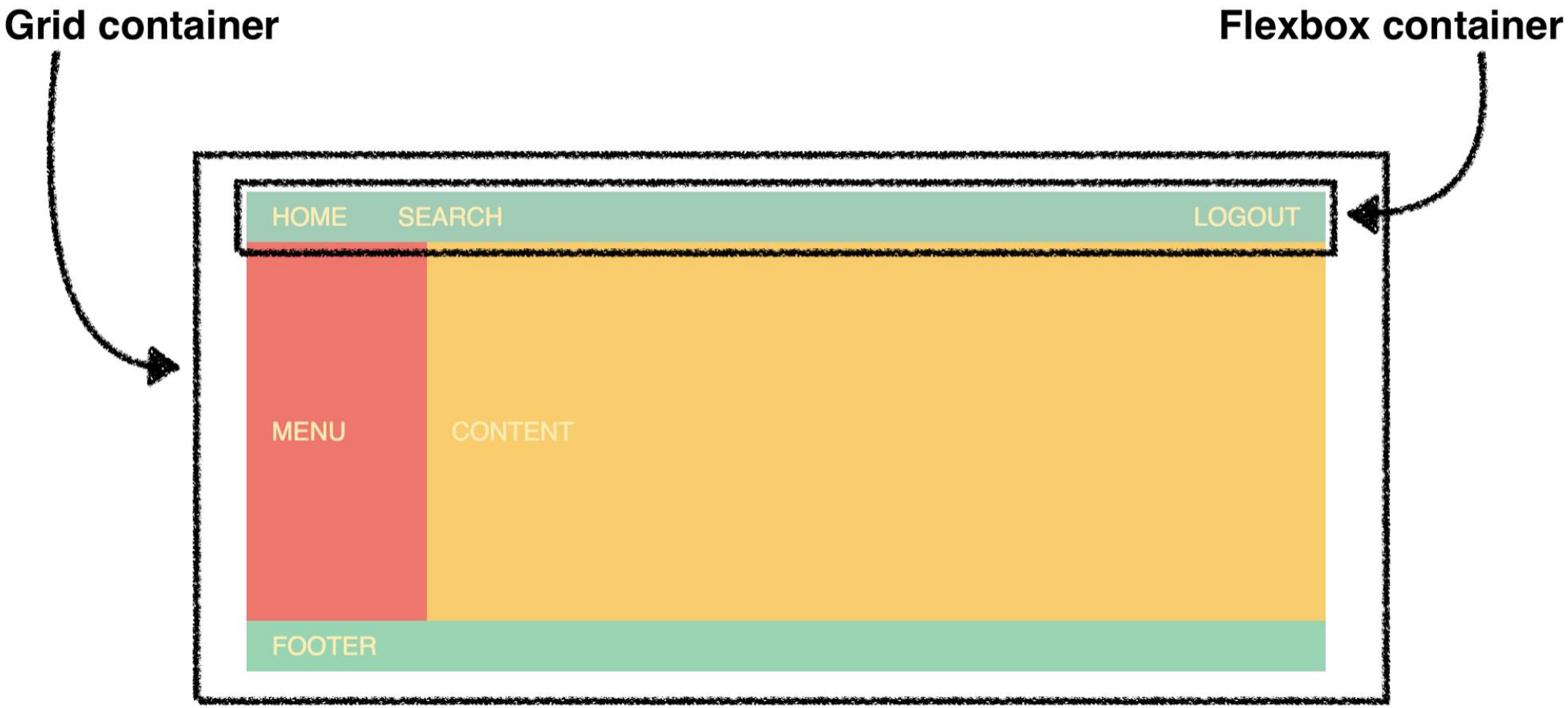
- Bi-dimensional;
- Starts from the space, when you define the grid;
- Allocating the content to the grid;
- Possibility of overlapping elements;
- Grid aligns everything into two directions: row and columns;


Flexbox


Grid


Grid
Flexbox

FLEX AND GRID



CAN I USE

CSS Grid Layout - CR

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors. Includes support for all `grid-*` properties and the `fr` unit.

Usage % of all users

Global 82.8% + 3.79% = 86.59%

unprefixed: 82.8%

Current aligned | Usage relative | Date relative | Show all

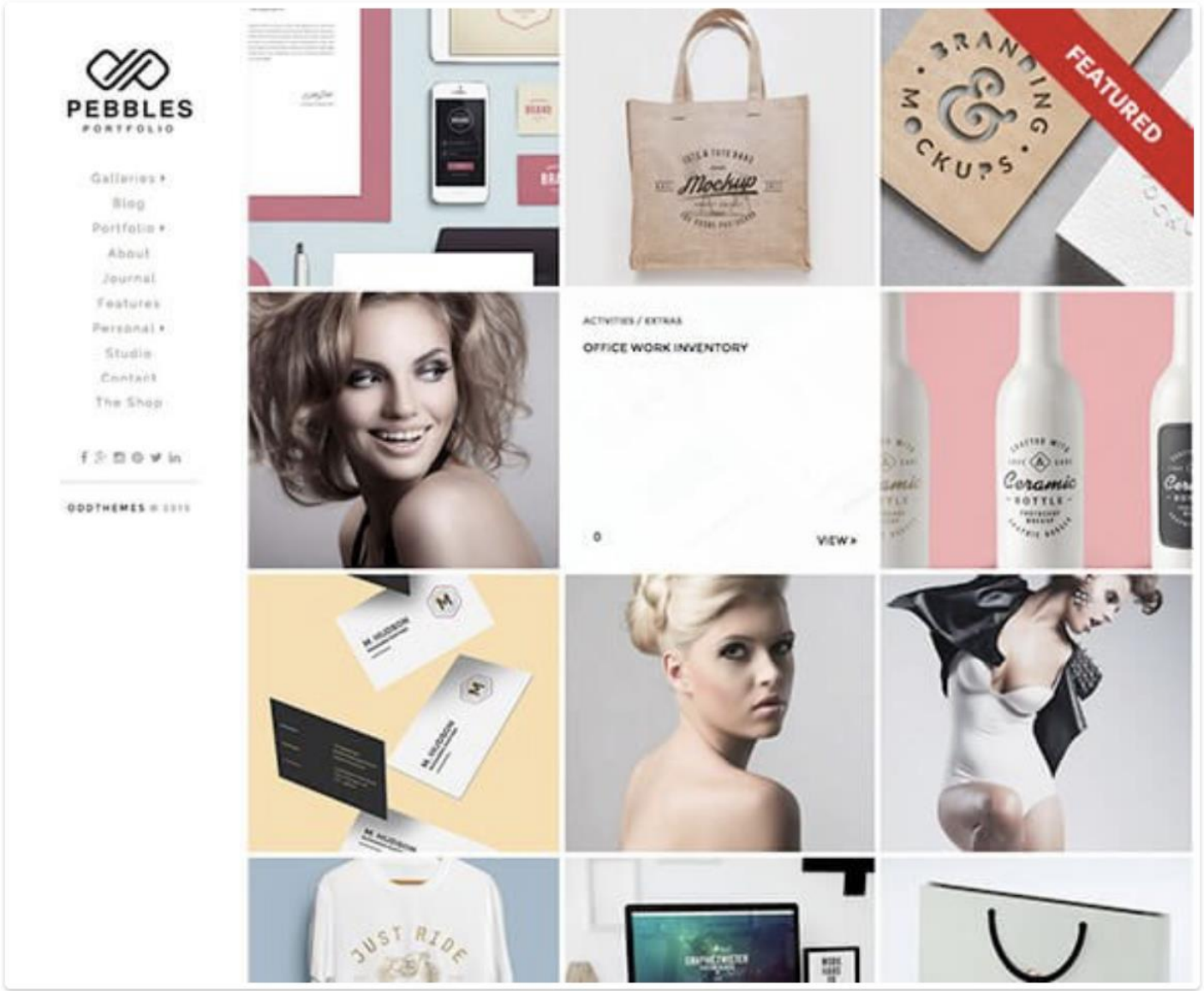
IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			1 49						
			62		10.2				
		57	63		10.3				4
2 11	16	58	64	11	11.2	all	64	11.8	6.2
	17	59	65	11.1	11.3				
		60	66	TP					
		61	67						

Notes | Known issues (2) | Resources (13) | Feedback

1 Enabled in Chrome through the "experimental Web Platform features" flag in chrome://flags

2 Partial support in IE refers to supporting an older version of the specification.

https://caniuse.com/#feat=css-grid

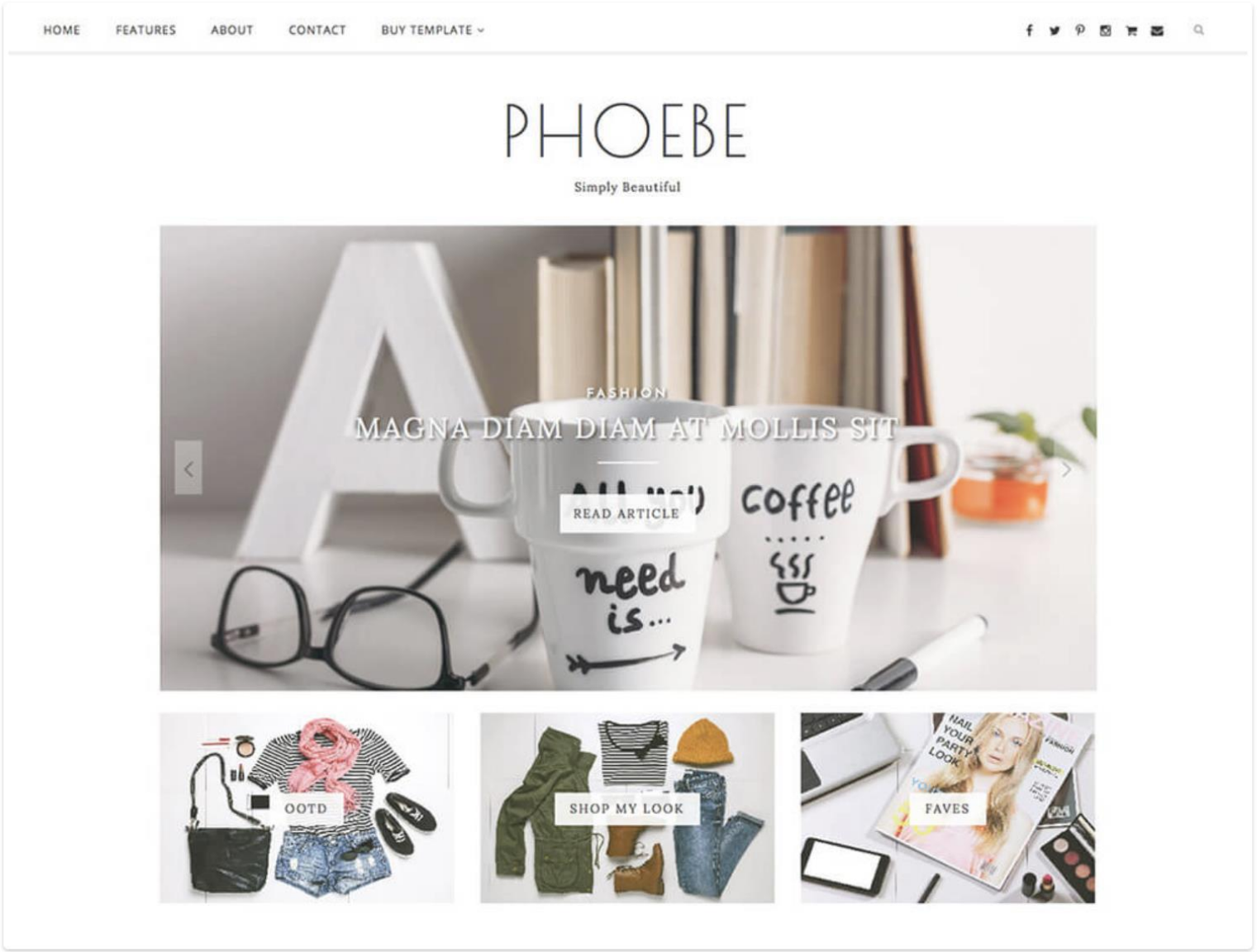


MAKE

YOUR

LIFE

EASY!



USE CSS GRID.

THANK YOU!

REFERENCES

https://gridbyexample.com/

https://gridbyexample.com/learn/

https://gridbyexample.com/patterns/

https://developer.mozilla.org/en-US/docs/Web/CSS/grid

https://www.w3schools.com/css/css_grid.asp

https://codepen.io/collection/XQKoYq/4/#

https://caniuse.com/#feat=css-grid

https://hackernoon.com/the-ultimate-css-battle-grid-vs-flexbox-d40da0449faf

https://css-tricks.com/snippets/css/complete-guide-grid/

https://learncssgrid.com/

http://griddy.io/

REFERENCES – YOU TUBE

https://www.youtube.com/watch?v=N5Lt1SLqBmQ (You Tube: Grid by Example, by Rachel Andrew)

https://www.youtube.com/watch?v=FEnRpy9Xfes (You Tube: Layout Land, by Jen Simmons)