CSS Grids.

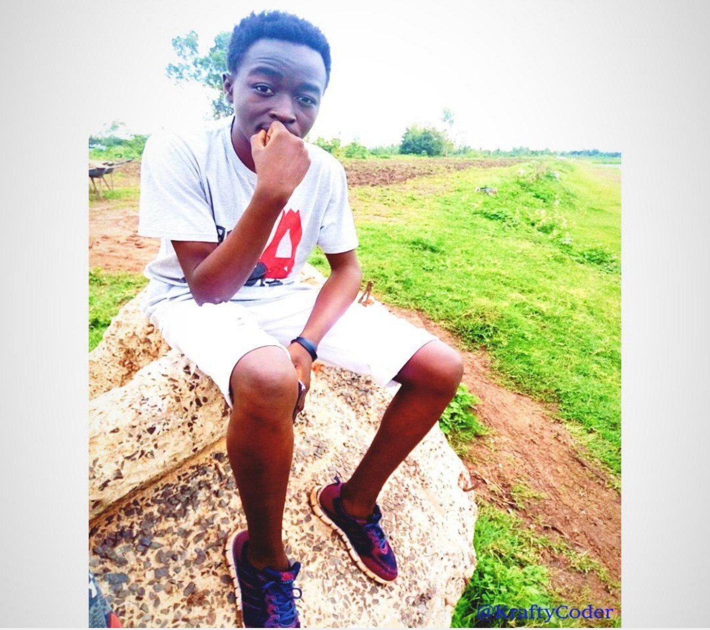
Understanding grids and what they do.

Podii Academy

ABOUT ME

Web developer, Open Source Enthusiast, designer and everything in between.

Community lead and mentor
Visit my site >> here

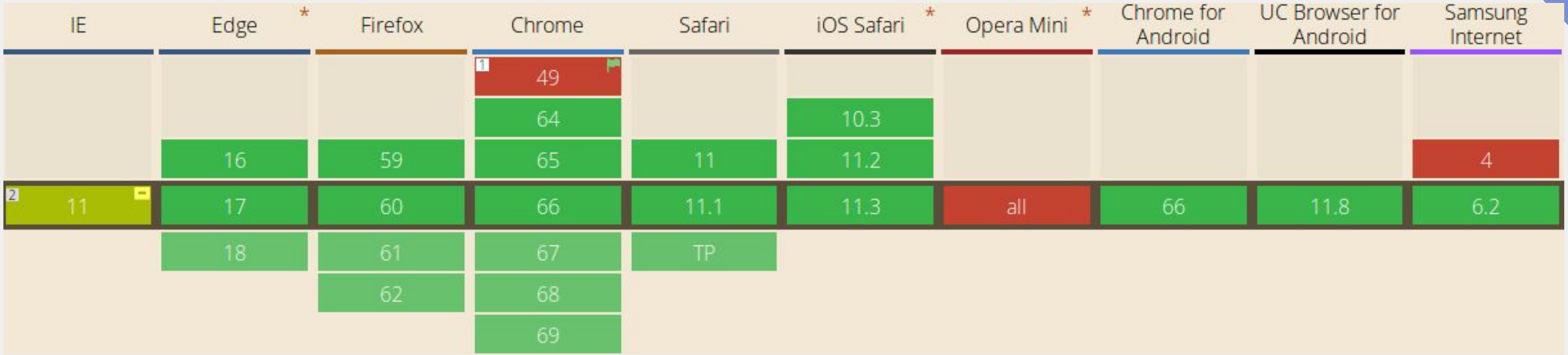


GRID-layout Concept

The core idea behind the Grid Layout is to divide a web page into columns and rows, along with the ability to position, size and resize the building block elements based on the rows and columns we have created in terms of size, position, and layer.

The grid also gives us a flexible way to change the position of elements with only CSS without any change to the HTML. This can be used with media queries to alter the layout at different breakpoints.

And even better, change in layout depending on the platform or device. (Responsiveness)



IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			[1] 49						
			64		10.3				
	16	59	65	11	11.2				4
[2] 11	17	60	66	11.1	11.3	all	66	11.8	6.2
	18	61	67	TP					
		62	68						
			69						

Initially we used to nest divs inside other divs and get to use flex-box for our stylings. Flex is only available for the row and column views.
This way we can’t have them arranged in any other way and wrapping wouldn't be of much help to us in this case.

Flex is really an amazing tool for styling but we can’t use it for the grid system. It will cause too much html code and too much styling involved too.

The flex CSS property specifies how a flex item will grow or shrink so as to fit the space available in its flex container.

More on this >> here

Flex can be set to the following values when in use: auto, initial , none or a positive number.

- ➔ Auto: sets the item to fill the container’s width and height and if there is free space, it goes on to occupy the space and shrinks to its minimum size to fit the container. This is equivalent to setting "flex: 1 1 auto"
- ➔ Initial: The default value; sized according to the height and width properties and also shrinks to the minimum size of the container, but doesn’t grow to absorb extra space. Equivalent to flex: 0 1 auto;
- ➔ Sized according to its width and height, completely inflexible. Neither shrinks or grows. Equivalent to flex: 0 0 auto;
- ➔ The item is given the specified proportion of the free space in the container. This is equivalent to setting "flex: <positive-number> 1 0".

We create a grid container by declaring
display: grid
or
display: inline-grid (on an element.)
As soon as we do this all direct children of that element will become grid items. In our case we use this in the wrapper div

A CSS grid flow can go in either direction. But by default it’s set to row.

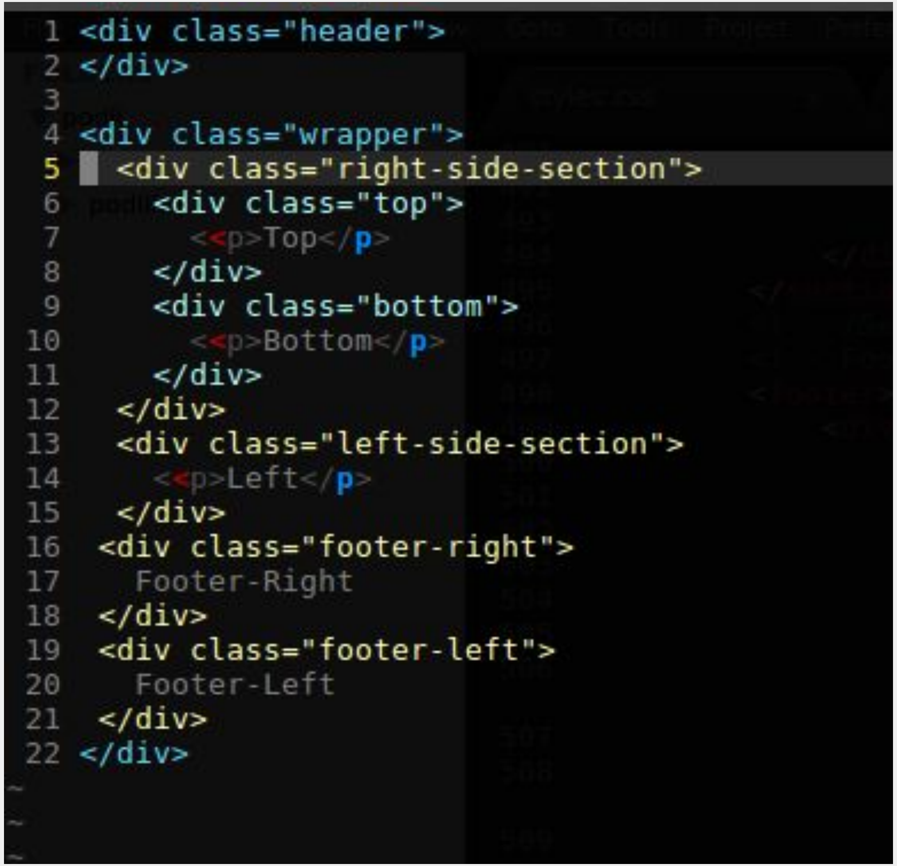
READ MORE >> here

if you need to change direction of the auto flow to columns you can do so by setting grid-auto-flow property to a value of column.

GRID Items

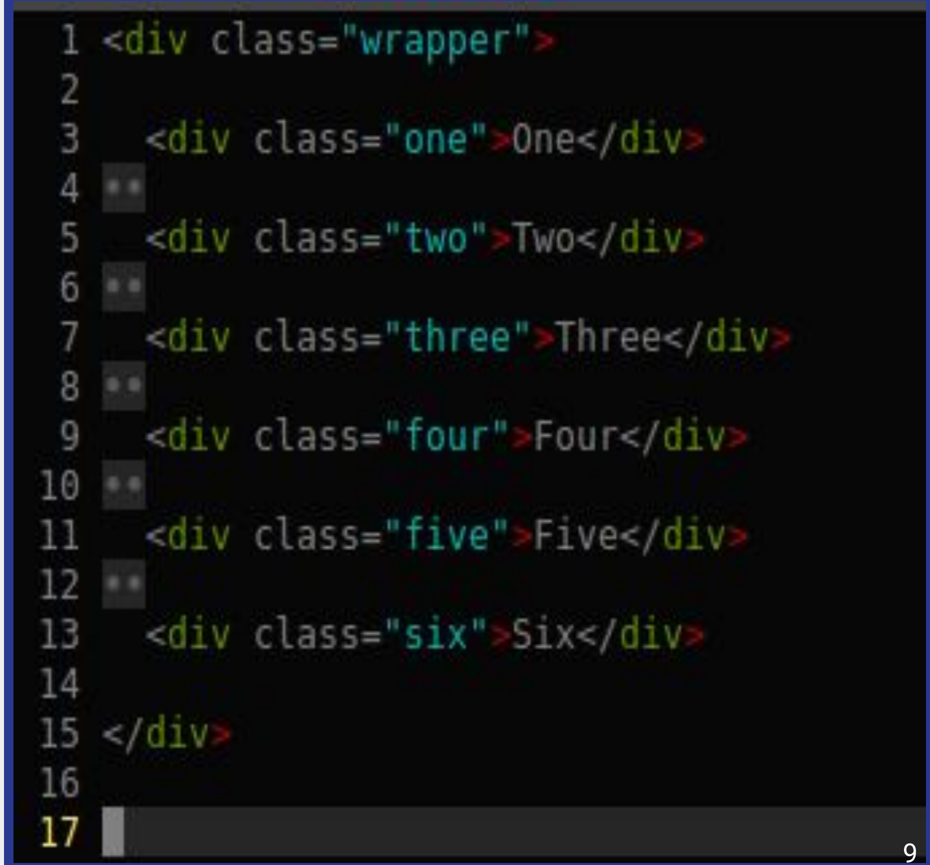
Your CSS grid items are <div> elements inside your container. Each item behaves similar to a <td> tag in a <table> but with a lot more flexibility.

“Oold” technique



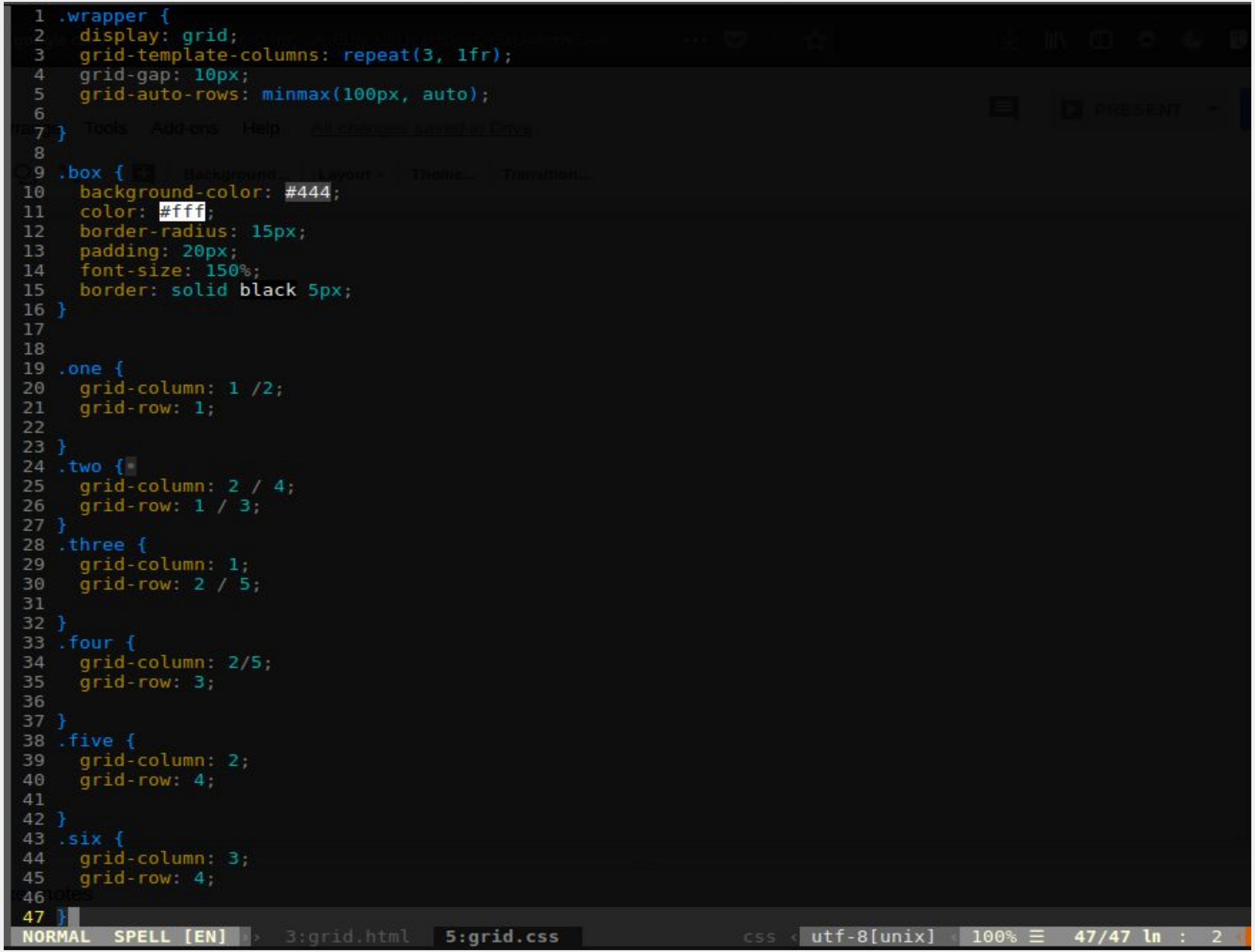
```
<div class="header">
</div>

<div class="wrapper">
  <div class="right-side-section">
    <div class="top">
      <p>Top</p>
    </div>
    <div class="bottom">
      <p>Bottom</p>
    </div>
  </div>
  <div class="left-side-section">
    <p>Left</p>
  </div>
 <div class="footer-right">
   Footer-Right
 </div>
 <div class="footer-left">
   Footer-Left
 </div>
</div>
```

NEW POWERED BY GRID



```
<div class="wrapper">

  <div class="one">One</div>

  <div class="two">Two</div>

  <div class="three">Three</div>

  <div class="four">Four</div>

  <div class="five">Five</div>

  <div class="six">Six</div>

</div>


```

THE Styling For Grid layout



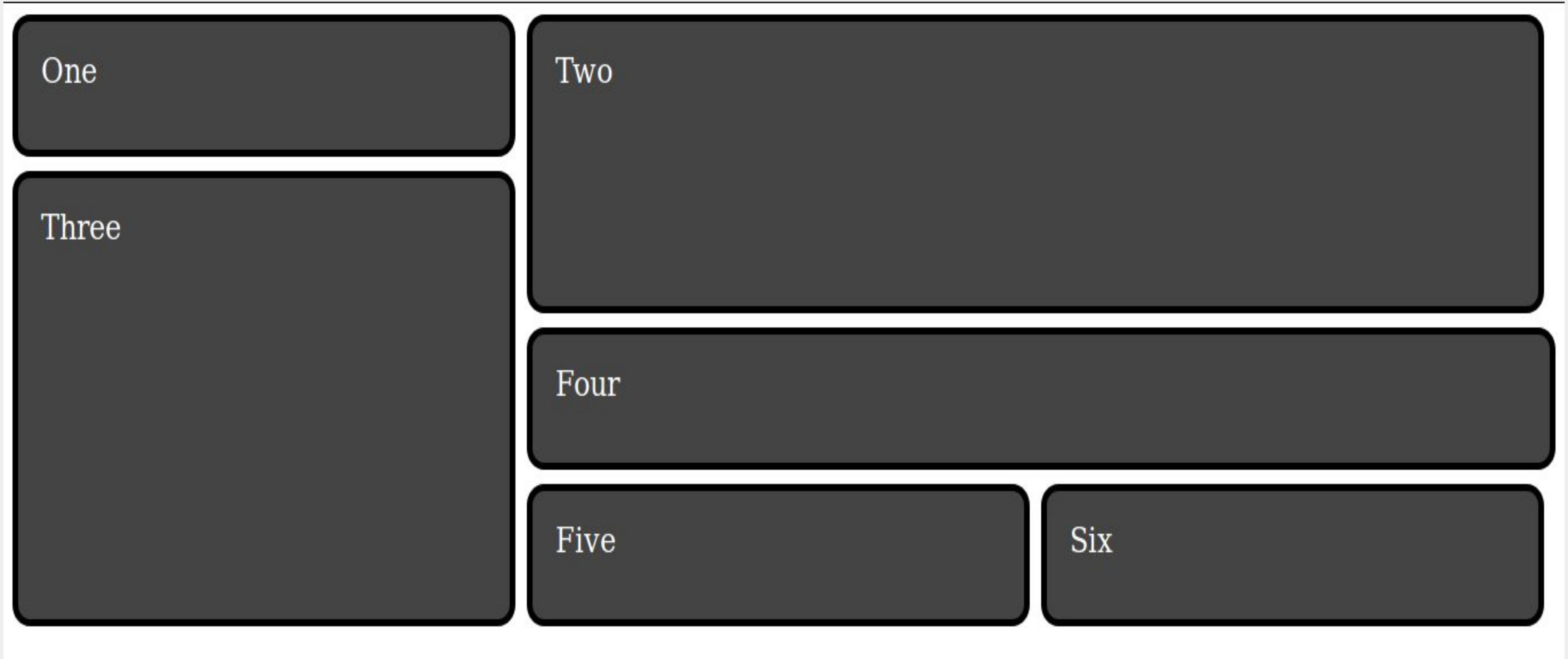
```css
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 10px;
  grid-auto-rows: minmax(100px, auto);

}

.box {
  background-color: #444;
  color: #fff;
  border-radius: 15px;
  padding: 20px;
  font-size: 150%;
  border: solid black 5px;
}


.one {
  grid-column: 1 /2;
  grid-row: 1;

}
.two {
  grid-column: 2 / 4;
  grid-row: 1 / 3;
}
.three {
  grid-column: 1;
  grid-row: 2 / 5;

}
.four {
  grid-column: 2/5;
  grid-row: 3;

}
.five {
  grid-column: 2;
  grid-row: 4;

}
.six {
  grid-column: 3;
  grid-row: 4;

}
```

Tracks can be defined using any length unit. Grid also introduces an additional length unit to help us create flexible grid tracks. The new fr unit represents a fraction of the available space in the grid container.

We also use direct fractions e.g. 1 / 3 as in the code we’ve just seen.

The next grid definition would create three equal width tracks that grow and shrink according to the available space.

Flex-Box RESULT of the code



Questions !?!

Thank you *.*