

# Web Presentation (CSS) for Layout, continued

# The CSS **display**: property

*Defaults for all HTML elements that display content...*

**display: block;**

- Stacks-up, top-over-bottom with other elements
- Uses the box model (padding, margin, borders)
- Is 100% wide (takes-up all the width on a line)

**display: inline;**

- Lines-up, side-by-side with other elements
- Ignores or does strange things to padding, margin, borders (no box model)
- Is only as wide as it needs to be; if wider than 100%, wraps to the next line

*Options for layout...*

**display: inline-block;**

- Uses the box model (like display: block) AND...
- Is only as wide as it needs to be (like display: inline)

**display: table-cell;**

- Neighboring block elements line-up side-by-side with equal height
- Ignores some aspects of the box model

**display: flex;**

- Neighboring block elements line-up side-by-side with equal height
- Uses the box model

**display: grid;**

- Everything within a GRID element can be placed anywhere you want, within the grid's box

# Step 1: Create Layout Blocks

```
<body>
```

```
  <div class="container">
```

```
    <headerheader
```

```
    <articlearticle
```

```
    <asideaside
```

```
    <footerfooter
```

```
  </div><!--.container-->
```

```
</body>
```

Other common structural elements:

- NAV
- MAIN
- SECTION

...and when all else fails:

- DIV

## Step 2: Choose a Layout Technique

At some point in your web page, you can break the normal document flow and position content side-by-side to create a columnar layout.

- ~~HTML Table~~ (not acceptable for layout)
  - ~~Float layout~~ (old fashioned but still used on older websites)
- 
- Inline-block ← Okay for small layouts (e.g. navigation elements)
  - CSS table layout ← Some use in the industry
- 
- Flex layout
  - Grid layout
- } Newest properties - complicated but powerful

# Choices

- **HTML tables** are for tabular data only! (Not for layout)
- The CSS **float** property is for moving small content to the side (left or right) and letting the rest of the content flow around it. (*Not for layout*)
- The CSS **inline-block** property is for lining up block elements side-by-side (simple - good for one-off occasional use)
- The CSS **table-cell** property is for creating a (simple) page layout with columns and rows (like a table, but not actually an HTML table)
- The CSS **flex** property is for lining up block elements side-by-side in a *one dimensional layout*
- The CSS **grid** property is for creating a any page layout with columns and rows - *a two dimensional layout*

# Flex vs. Grid: one dimension vs. two

## Flex

- Create a “container” – everything inside lines-up however you want
- Content is not constrained – grows vertically as needed
- Good for presenting a lot of dynamic blocks of content ("flexible")

## Grid

- Create a “container” – define what goes where, vertically (columns) and horizontally (rows)
- Content is made to fit the grid as much as possible
- Good for creating a page layout

# CSS Grid Layout

New Core Technology from the W3C

# Early 2017

- New CSS properties for Grid Layout
  - 18 new properties
  - Three new functions
- Full support
  - Chrome
  - Firefox
  - Safari
  - iOS Safari
  - Android Browser
  - Chrome for Android
  - Edge (recent)
  - IE (not fully)

## CSS properties

`grid-template-columns`

`grid-template-rows`

`grid-template-areas`

`grid-template`

`grid-auto-columns`

`grid-auto-rows`

`grid-auto-flow`

`grid`

`grid-row-start`

`grid-column-start`

`grid-row-end`

`grid-column-end`

`grid-row`

`grid-column`

`grid-area`

`grid-row-gap`

`grid-column-gap`

`grid-gap`

## CSS function

`repeat()`

`minmax()`

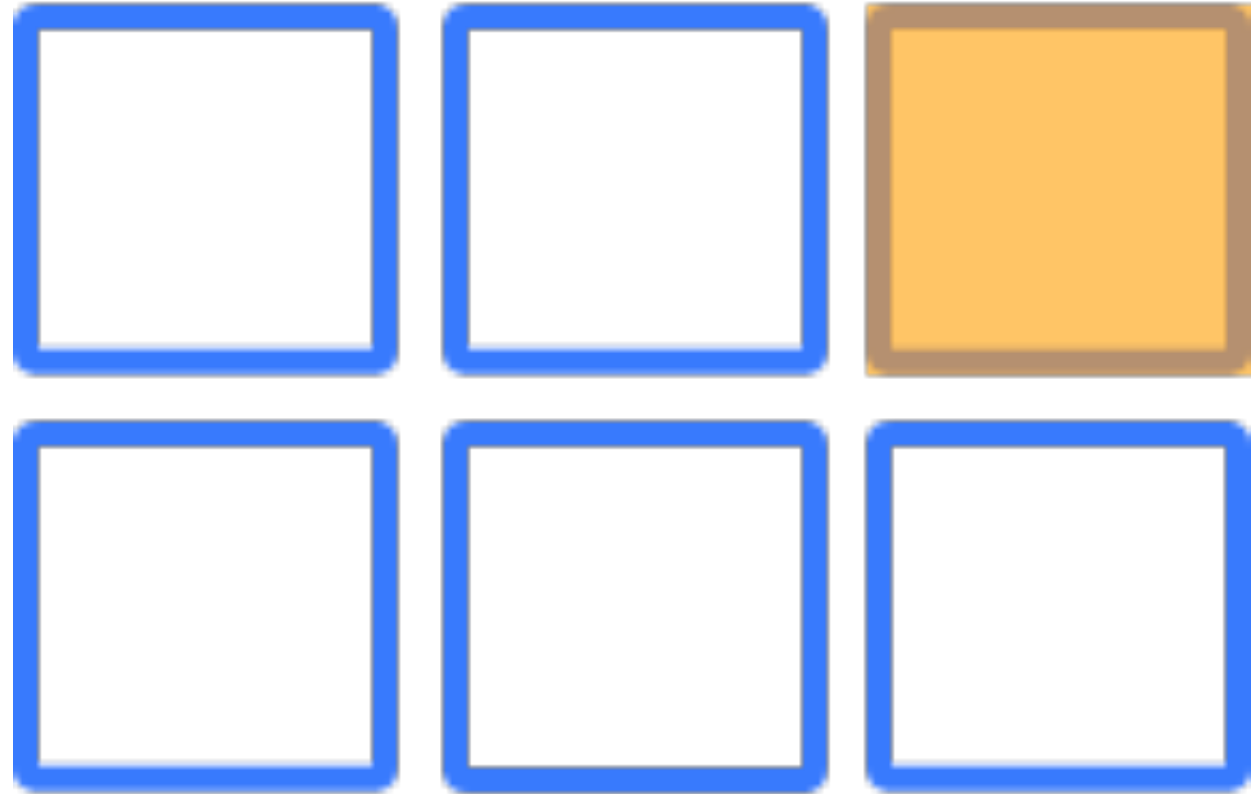
`fit-content()`



# Grid Terminology and Concepts

# Grid Cell

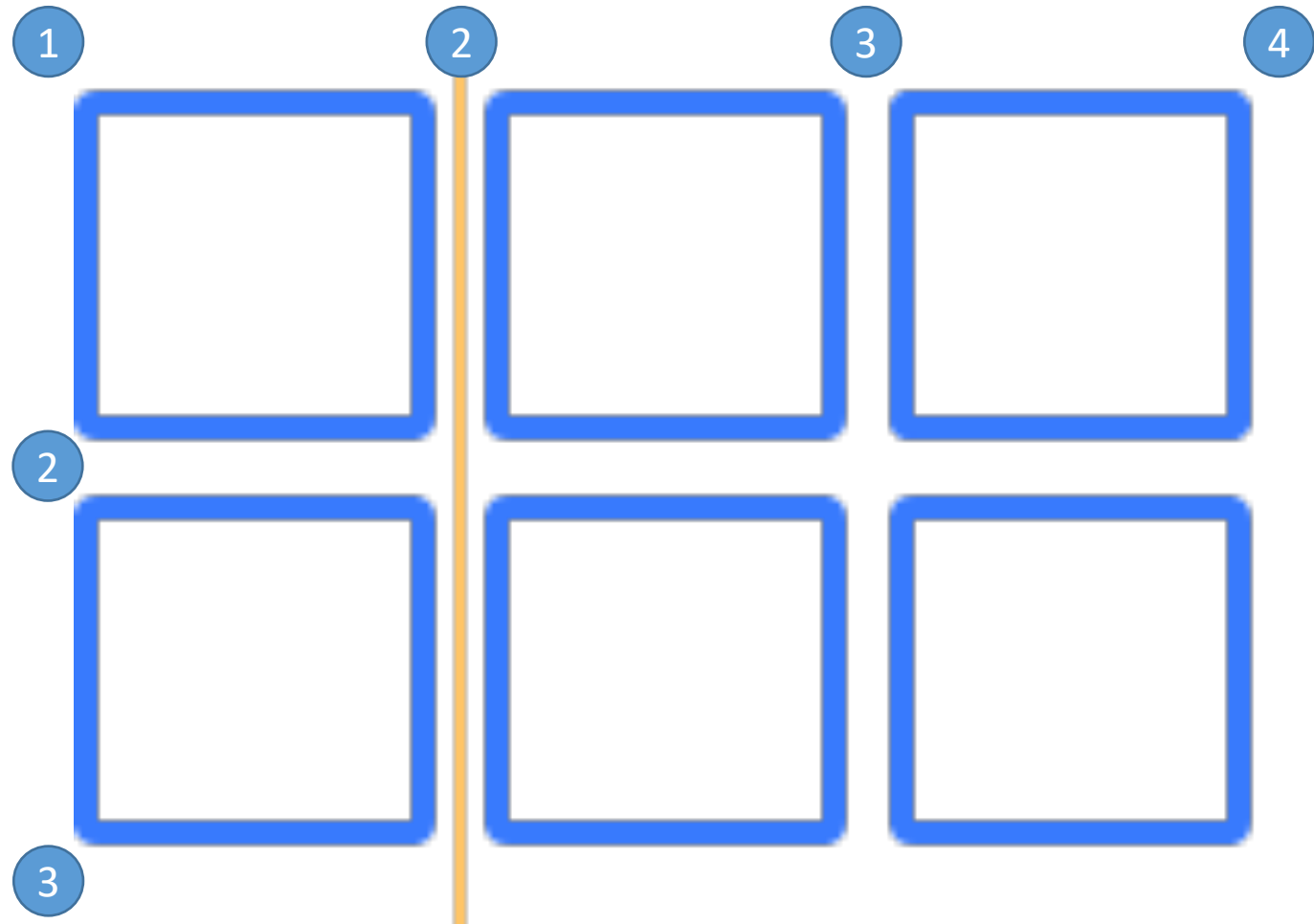
A single unit of a CSS grid



# Grid Lines

The vertical and horizontal lines that divide the grid and separate the columns and rows

*Programmers beware!  
Counting lines starts with  
"one" (not zero)*

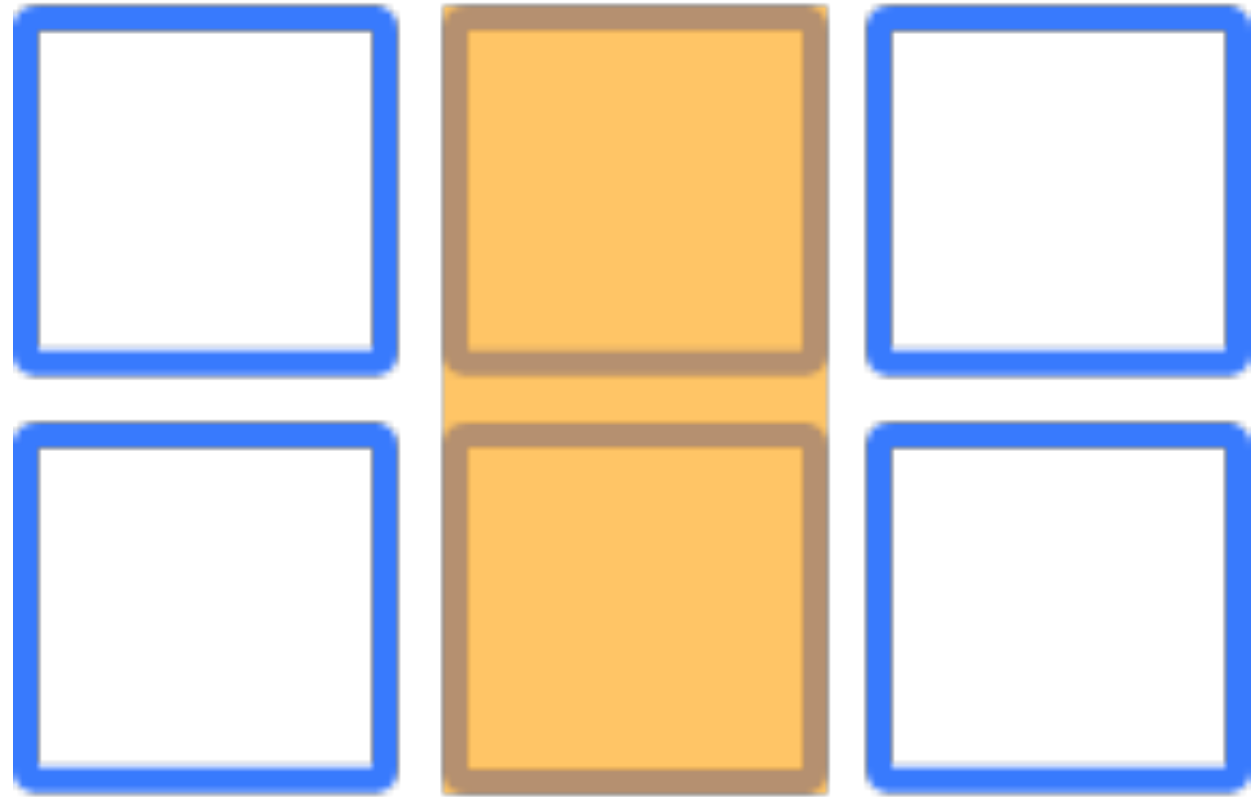


# Grid Track

The space between two grid lines.  
This space can be horizontal or vertical

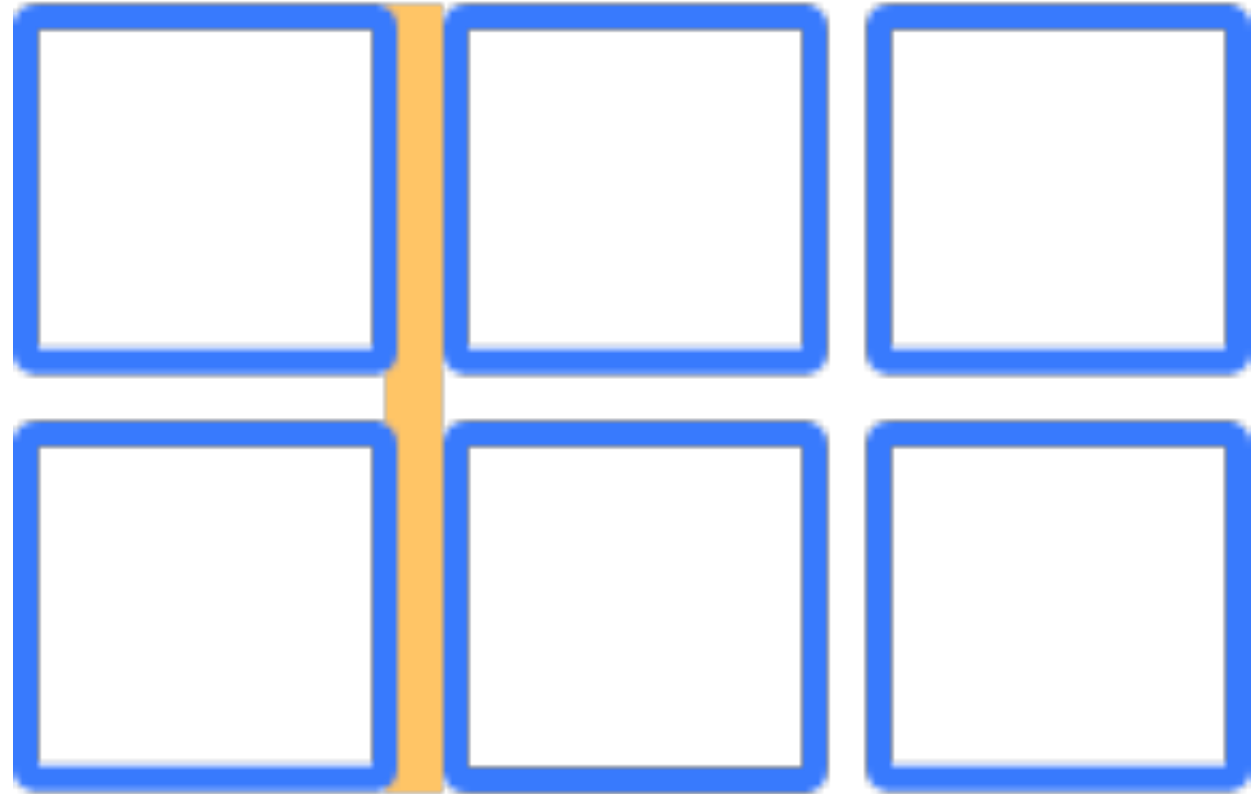
**Grid column:** a vertical track

**Grid row:** a horizontal track



# Grid Gutter

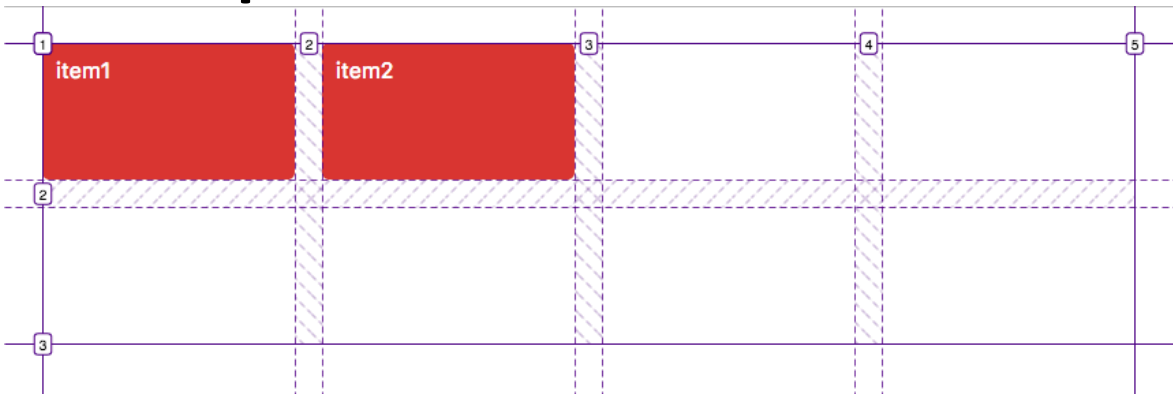
The optional space between rows and columns in a grid



# The explicit and implicit grid

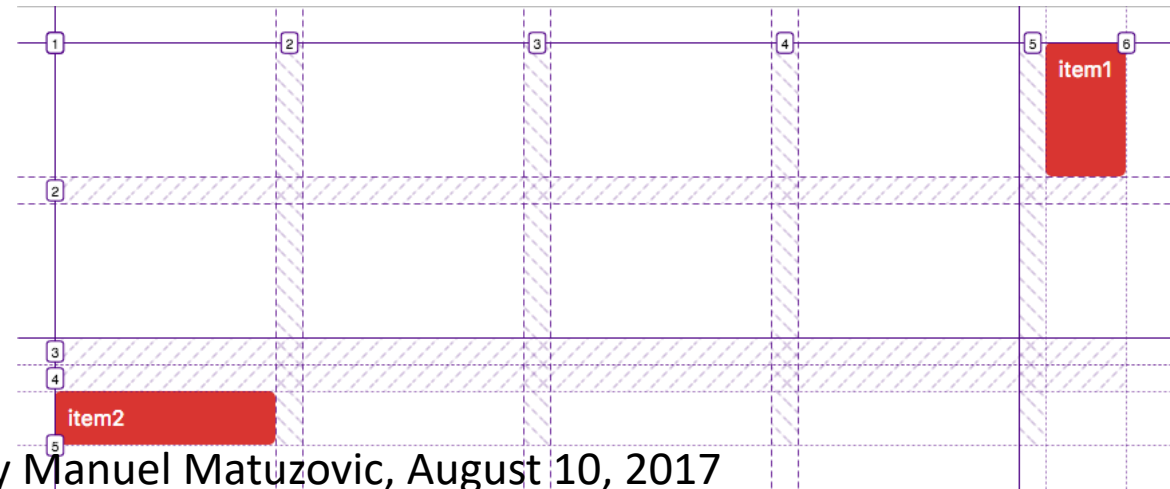
## Explicit Grid

- When you use:  
**grid-template-columns** and  
**grid-template-rows**
- Everything you setup *within* the column and row tracks are **explicit**



## Implicit Grid

- When you use:  
**grid-auto-columns** and  
**grid-auto-rows**  
everything is **implicit**



# Using the Grid Lines for Positioning

- Use CSS properties...  
**grid-column-start** and **grid-column-end**, or **grid-row-start** and **grid-row-end**  
...to have cells *span* tracks
- Or use their shortcut counterparts:  
**grid-column** and **grid-row**  
e.g.  
`grid-column: 1 / 4;`  
`grid-row: 1 / 3;`

