

CSS Layout: Grid

Module 3 - 03

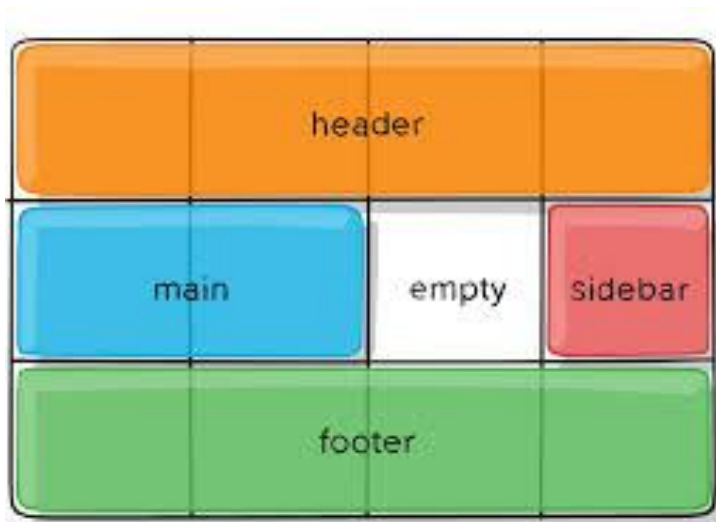
Objectives

1. CSS Variables
2. CSS Grid
3. Responsive Design

CSS Grid

CSS Grid Layout is a 2-dimensional system, meaning it can handle both columns and rows.

You work with Grid Layout by applying CSS rules both to a parent element (which becomes the Grid Container) and to that element's children (which become Grid Items).



Grid Terminology

Grid Container

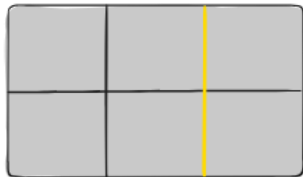
The element on which `display: grid` is applied.

Grid Item

The children (i.e. *direct* descendants) of the grid container.

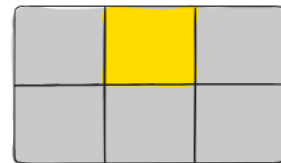
Grid Line

The dividing lines that make up the structure of the grid. They can be either vertical ("column grid lines") or horizontal ("row grid lines") and reside on either side of a row or column.



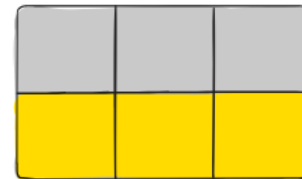
Grid Cell

The space between two adjacent row and two adjacent column grid lines. It's a single "unit" of the grid.



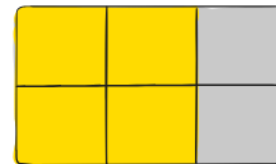
Grid Track

The space between two adjacent grid lines. You can think of them like the columns or rows of the grid.



Grid Area

The total space surrounded by four grid lines. A grid area may be composed of any number of grid cells.



Display

Defines the element as a grid container and establishes a new grid formatting context for its contents.

Values:

- **grid** – generates a block-level grid

```
.container {  
  display: grid;  
}
```

grid-template-columns

Defines the columns of the grid with a space-separated list of values. The values represent the track size, and the space between them represents the grid line

```
grid-template-columns: 4fr 1fr 2fr;
```

Columns can be a length, a percentage, or a fraction of the free space in the grid (using the `fr` unit)

Fractional Units (`fr`) is the preferred unit to allow the grid to correctly adjust for different display sizes.

grid-template-areas

Grid template areas is where a developer can set up names for different sections of the defined grid. `grid-template-areas` is defined as a series of strings. Each string will have as many sections as there are columns defined and there will be a row created for each string in the sequence.

```
grid-template-areas:  
    "header    header header"  
    "main      .      sidebar"  
    "footer footer copyright";
```

grid-area

Grid Areas are applied to the Grid Items and identify the area of the grid the element should be displayed in.

```
div.header {  
    grid-area: header;  
}  
  
div.main {  
    grid-area: main;  
}  
  
div.sidebar {  
    grid-area: main;  
}
```

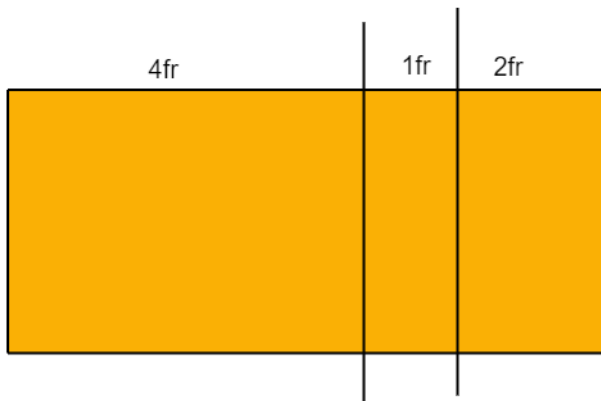
grid-template-columns

Defines the columns of the grid with a space-separated list of values. The values represent the track size, and the space between them represents the grid line

```
grid-template-columns: 4fr 1fr 2fr;
```

Columns can be a length, a percentage, or a fraction of the free space in the grid (using the `fr` unit)

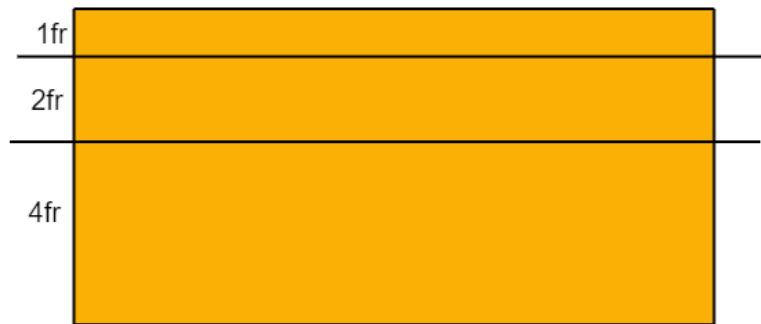
Fractional Units (`fr`) is the preferred unit to allow the grid to correctly adjust for different display sizes.



grid-template-rows

Defines the rows of the grid with a space-separated list of values. The values represent the track size, and the space between them represents the grid line. ***Rarely used as the default is to auto size, which sizes the rows to the size of the content.***

```
grid-template-rows: 1fr 2fr 4fr;
```



Grid Gap

Grid Gap is area between the grid tracks.

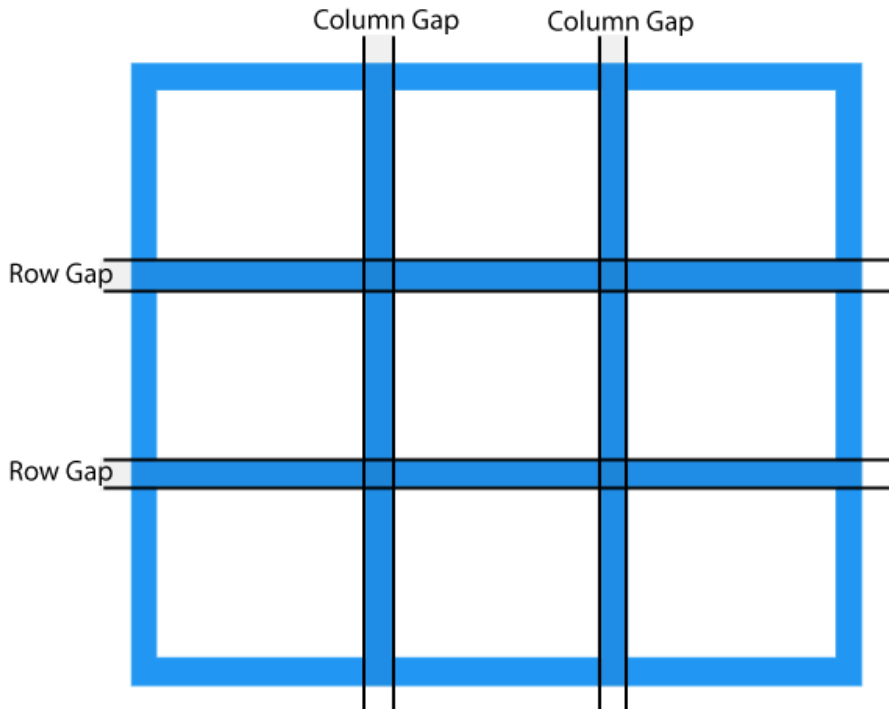
It can be defined as the gap between columns, the gap between rows, or for both columns and rows. Grid gap is applied to the container.

Grid Gap adds space between columns and rows, but not on the outer edges of the grid.

```
grid-column-gap: 10px;
```

```
grid-row-gap: 15px;
```

```
grid-gap: 20px;
```



CSS Grids: Defining

To define a grid we must specify a display attribute with a value of grid:

```
.myGrid {  
  
    display: grid;  
  
}
```

In this example, the CSS code will specify using a selector by class, that an html element with a class name of myGrid be defined as a grid. All of the direct children of the container will become grid items.

CSS Grids: Columns and grid-gap

grid-template-columns: This property defines the number of columns (and their respective width).

grid-gap: Creates a gutter, setting a width of space between the columns and rows. (note – not around outside of container)

```
body {  
  display: grid;  
  grid-template-columns: 1fr 2fr 2fr 2fr 2fr 1fr;  
  gap: 40px;  
}
```

fr stands for fractional unit. The 1st column will occupy 10% of the width, the second 20%, etc.

Adds a buffer between the cells of the grid.

CSS Grids: Template Areas

grid-template-areas: Matches each area of the grid to a specific HTML element. By virtue of how this works, it also defines the number of rows.

```
body {  
  display: grid;  
  grid-template-columns: 1fr 2fr 2fr 2fr 2fr 1fr;  
  gap: 20px;  
  grid-template-areas:  
    ". header header nav nav ."  
    ". main main main main ."  
    ". fall-festival fall-festival store store .";  
}
```

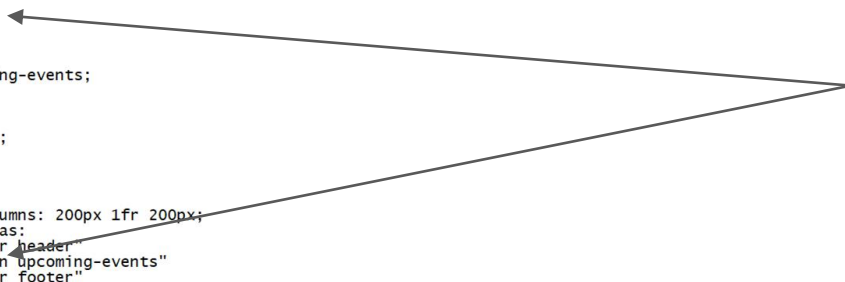
The period represents an empty space in the final grid layout.

CSS Grids: Template Areas

grid-areas: Allows you to name each element specifically to be able to access it, or allows to give a name for each element to be displayed in the grid.

```
header {  
  grid-area: header;  
}  
  
nav#menu-nav {  
  grid-area: menu-nav;  
}  
  
main {  
  grid-area: main;  
}  
  
aside {  
  grid-area: upcoming-events;  
}  
  
footer {  
  grid-area: footer;  
}  
  
.container {  
  display: grid;  
  grid-template-columns: 200px 1fr 200px;  
  grid-template-areas:  
    "header header header"  
    "menu-nav main upcoming-events"  
    "footer footer footer"  
  ;  
  height: 100vh;  
  gap: 10px;  
}
```

grid-template-area
property matches up to
the grid-area property

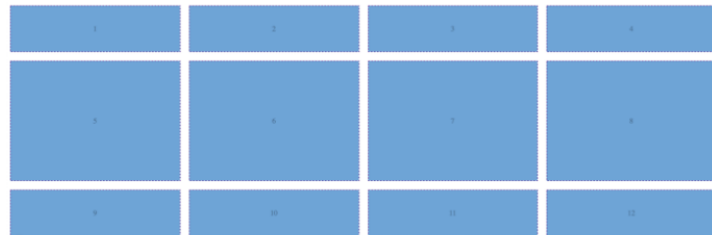


CSS Grids: Template Rows

grid-template-rows: Specifically allow you to specify how many rows.

```
body {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr 1fr;  
  grid-template-rows: 100px 1fr 100px;  
  grid-template-areas:  
    "header header nav nav "  
    "main main main main"  
    "fall-festival fall-festival store store";  
}
```

First and third row have a set height of 100px.
Second row will take up the rest of the available space.



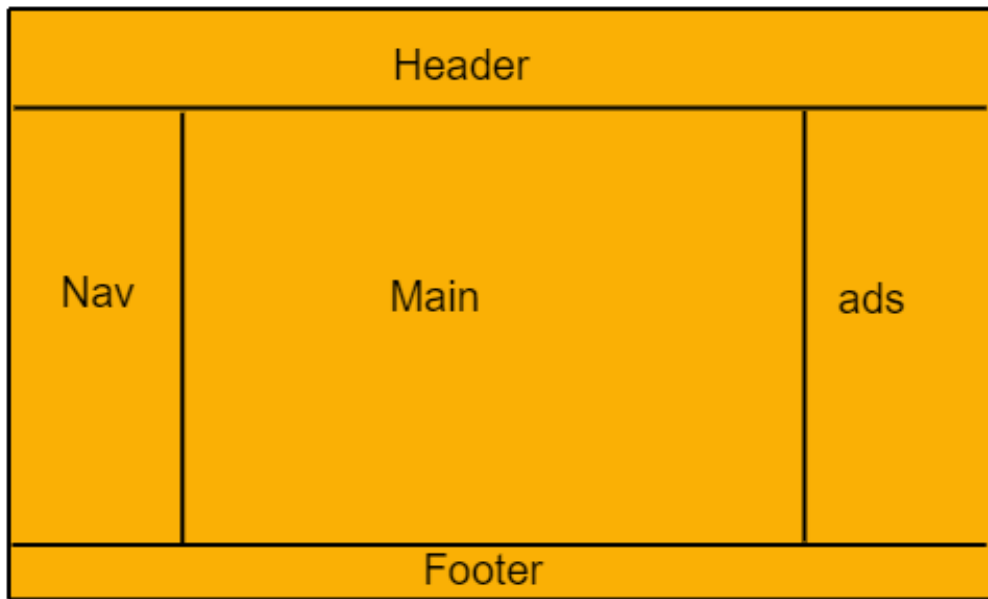
CSS Units: viewport vs. pixels vs. fractional units

- Pixels are fixed length units. 100px is the same size no matter how big the display (viewport)
- vh (viewport height) and vw (viewport width) can be used to set the height and/or width the container (1vh is 1% of viewport height)
- fr (fractional units) typically used to specify how much of the container each grid item is allotted

```
body {  
  margin: 0;  
  padding: 0;  
}  
  
.container {  
  height: 100vh;  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr 1fr;  
  grid-template-rows: 1fr 1fr 1fr;  
  grid-gap: 20px;  
}
```

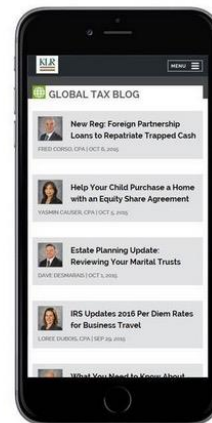
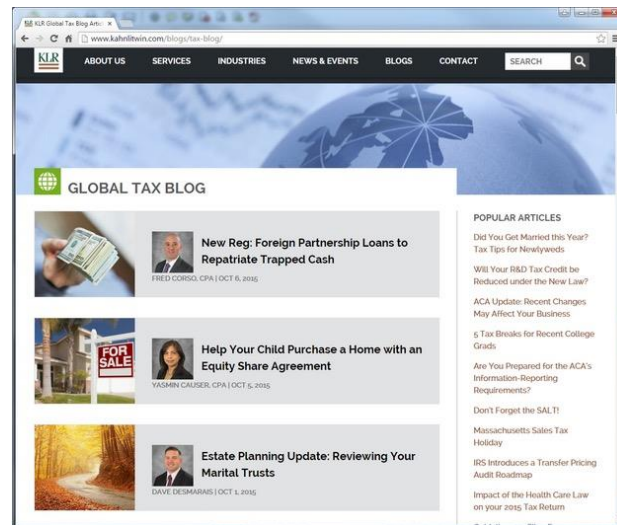
The “Holy Grail” Layout

Why is it called “Holy Grail Layout” - it used to be very difficult, almost impossible to achieve, but everyone tried with different levels of success. Grid makes this layout easy.



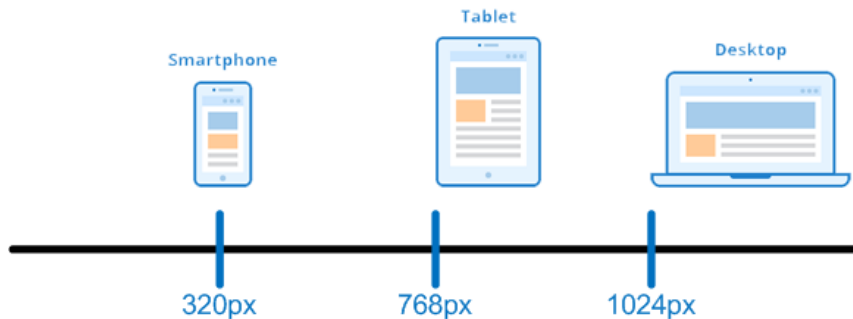
Responsive Design

1. Design for mobile first
2. Avoid fixed dimensions
3. Design for specific devices
 - a. phone
 - b. tablet
 - c. computer
4. Prioritize content differently for different devices
5. Use responsive sizing like em, rem, vh, and wh instead of absolute sizing like px
6. Use either min-width/height or max-width/height, but not both



CSS Media Query

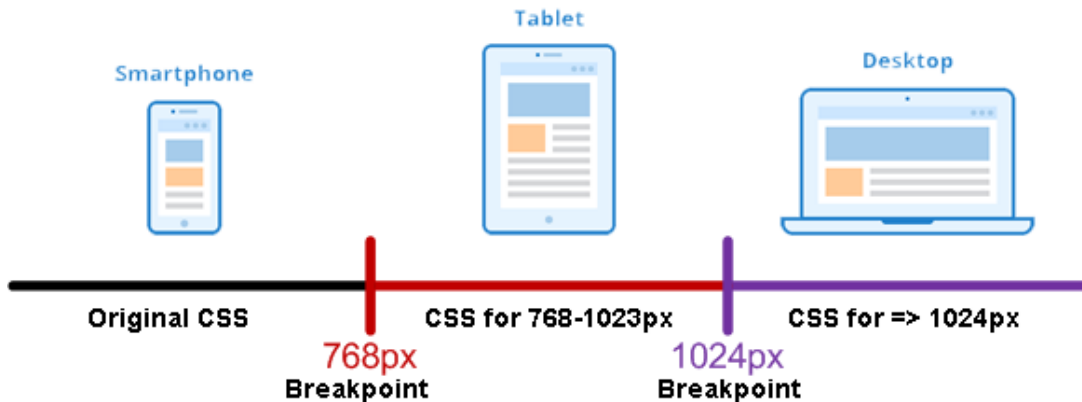
- Creates conditional CSS that will be applied for different sizes of viewports, for example css that will be applied to a phone and different CSS that will be applied to a computer.
- Done by creating **breakpoints** where the CSS being applied changes
- [Documentation](#)
- Common Sizes:
 - Small (smartphones): 320px
 - Medium (tablets): 768px
 - Large (computer): 1024px
- Media Query Examples:
 - `@media screen and (max-width: 768px) { .. }`
 - `@media screen and (max-width: 450px) { .. }`



Breakpoints

Breakpoints are pixel sizes where the CSS “breaks” and changes to a different media query to adjust the presentation of the site.

- Design for the smallest size first, then create a breakpoint for each device size where you want to change the layout.



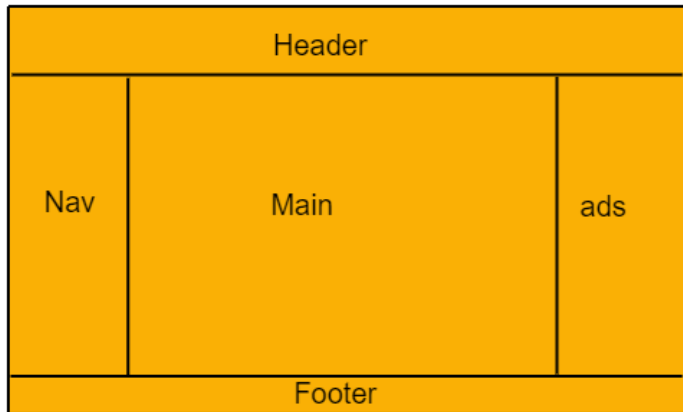
HTML Viewport Meta Tag

- Added in HTML5 - instructs the browser how to control page dimensions
- Required for mobile/tablets to properly size, resize, and zoom a page
- Current recommendation is to keep it minimal and use MediaQuery for sizing, unless you are working for an IOT (Internet of Things) application
- Added to the <head> of an HTML document, and will almost always be the following:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

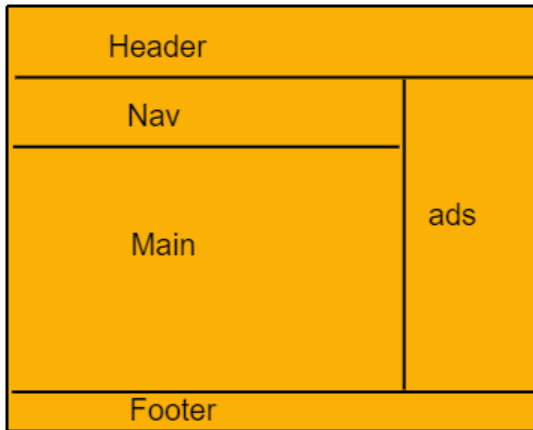
The responsive holy grail layout

Computer - >1074px



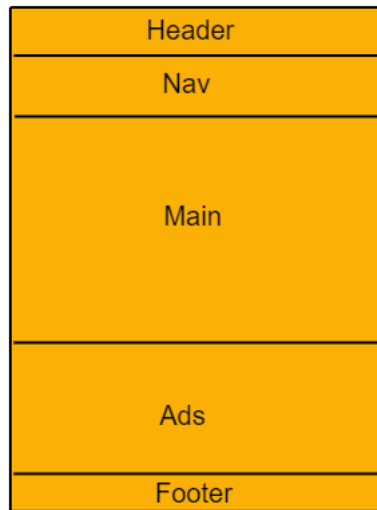
```
.container {  
  grid-template-columns: 200px 1fr  
  200px;  
  grid-template-areas:  
    "header header header"  
    "menu-nav main upcoming-events"  
    "footer footer footer"  
};  
}
```

Computer - 768-1074px



```
@media (max-width: 1024px) {  
  .container {  
    grid-template-areas:  
      "header header"  
      "menu-nav menu-nav"  
      "main upcoming-events"  
      "footer footer";  
    grid-template-columns: 4fr 1fr;  
  }  
}
```

Mobile- <768px



```
@media (max-width: 768px) {  
  .container {  
    grid-template-areas:  
      "header"  
      "menu-nav"  
      "main"  
      "upcoming-events"  
      "footer";  
    grid-template-columns: 1fr;  
  }  
}
```

CSS Grid Links

[CSS Tricks: Complete Guide to Grid](#)

comprehensive documentation with examples

[CSS Grid Garden](#)

game to learn Grid

[Layoutit!](#)

grid layout tool