

## Flexbox Properties

### Parent (Flex Container)

**display:** flex | inline-flex;

**flex-direction:** row | row-reverse | column | column-reverse;

**flex-wrap:** wrap | nowrap | wrap-reverse;

**flex-flow** (shorthand for flex-direction and flex-wrap)

**justify-content** (main axis): flex-start | flex-end | center | space-between | space-around | space-evenly;

**align-items** (cross axis - adjust to individual sizes): flex-start | flex-end | center | baseline | stretch;

**align-content** (cross axis - adjust to largest item): flex-start | flex-end | center | stretch | space-between | space-around;

### Children (Flex Items)

**order:** <integer>;

**flex-grow:** <number>;

**flex-shrink:** <number>;

**flex-basis:** <length> | auto;

**flex:** shorthand for grow, shrink, and basis (default: 0 1 auto)

**align-self:** overrides alignment set on parent

## Grid Properties

### Parent (Grid Container)

**display:** grid | inline-grid;

**grid-template-columns**

**grid-template-rows:** [optional: line name, in square brackets] <track-size> | <repeat>;  
track-size: length, %, fr, auto  
line name: an arbitrary name for this item. If no name assigned, a number is used

#### EXAMPLES:

```
.myClass {
  grid-template-columns: [col1] 40px [col2] 3fr;
  grid-template-rows: 50% 25vh auto;
}

.anotherClass {
  grid-template-rows: repeat(2, 350px [name]) 10%;
}
translates to
.anotherClass {
  grid-template-rows: 350px [name] 350px [name] 10%;
}
```

#### grid-template-areas:

List of names of areas. First, name areas via selector. Then specify layout via this property. Area name must be specified for each column/row. A . indicates no content in this row/column.

Note: in this example, the lines are named automatically: header-start, header-end, article-start, article-end, etc.

#### EXAMPLES:

```
.class1 {
  grid-area: header;
}
.class2 {
  grid-area: article;
}
.class3 {
  grid-area: aside;
}
.wrapper {
  grid-template-columns: 1fr 3fr;
  grid-template-rows: auto;
  grid-template-areas:
    "header header header header"
    "aside . article article";
}
```

#### grid-template:

Shorthand for grid-template-rows, grid-template-columns, and grid-template-areas in 1 declaration. Not covered in class.

**grid-column-gap:** <number>;  
**grid-row-gap:** <number>;  
Distance between rows and/or columns.

**grid-gap:**  
Shorthand for grid-column-gap and grid-row-gap.  
1 number = same in all directions  
2 numbers = row column

**justify-items:** start | end | center | stretch;  
align grid items on row axis  
stretch is default

**align-items:** start | end | center | stretch;  
align grid items on column axis  
stretch is default

**justify-content:** start | end | center | stretch | space-around | space-between | space-evenly;

If size of grid container is bigger than total of grid items, you can align grid items within the container (like flexbox). This works on row axis.

**align-content:** start | end | center | stretch | space-around | space-between | space-evenly;

If size of grid container is bigger than total of grid items, you can align grid items within the container (like flexbox). This works on column axis.

**grid-auto-columns**

**grid-auto-rows:** <track-size>;

If you create grid cells beyond those specified in grid-template-columns and grid-template-rows, this specifies how big these extra rows/columns should be.

**grid:** shorthand for all of the above properties. Not covered in class.

## Children (Grid Items)

**grid-column-start**

**grid-column-end**

**grid-row-start**

**grid-row-end:** <number> | <name> | span <number> | span <name> | auto;

This is the longhand for declaring individual values for start and end points for rows and columns.

### EXAMPLES:

```
.class1 {  
  grid-column-start: 1;  
  grid-column-end: span 4;  
  grid-row-start: 3;  
  grid-row-end: span footer-end;  
}
```

**grid-column**

**grid-row:** <start-line> / <end-line> | <start-line> / span <value>;

Combines start and end values, as used extensively in class.

### EXAMPLES:

```
.class1 {  
  grid-column: 1 / span 4;  
  grid-row: 3 / span footer-end;  
}
```

**grid-area:** <name> | <row-start> / <column-start> / <row-end> / <column-end>;

OR

<name>;

If you're confused, no wonder. grid-area can be used in 2 different ways:

a. Assign a name for the grid-template-areas property (see above example under grid container/grid-template-areas)

b. Assign a name AND the dimensions for a grid-template-areas property. If you use this methodology, you would not necessarily need a grid-template-rows and grid-template-columns declaration, depending on other factors.

### EXAMPLES:

```
.class1 {  
  grid-area: 1 / name3 / namedline / 4;  
}
```

**justify-self:** start | end | center | stretch;

Aligns content in a grid item on the row axis. Overrides justify-items.

**align-self:** start | end | center | stretch;

Aligns content in a grid item on the column axis. Overrides align-items.