CSS Grid

WEEK 6

CSS GRID

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

CSS GRID

Flexbox layout is most appropriate to the components of an application, and small-scale layouts*, while the Grid layout is intended for larger scale layouts.

FLEX (One-dimensional grid)
CSS Grid (Two-dimensional grid)

*not always true. bootstrap grid is flex-based.

CSS GRID

CSS-Tricks Complete Guide to CSS Grid >>

Great source for CSS Grid (background, break-down of the basic concepts)

W3Schools CSS Grid >>

Exercises and examples from W3Schools

Gridbyexample >>

Some simple layout examples in snippet form

BASIC CONCEPTS

Grid container vs. Grid item [>>]

Grid line
(column grid line or row grid line)

Grid track
(space between two grid lines; either rows or columns)

Grid cell

Grid area (a bunch of cells)

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

A grid container contains grid items. By default, a container has one grid item for each column, in each row, but you can style the grid items so that they will span multiple columns and/or rows [>>]

CONTAINER VS ITEM

Properties for the Grid Container

- display
- grid-template-columns
- grid-template-rows
- grid-template-areas
- grid-template
- grid-column-gap
- grid-row-gap
- grid-gap
- justify-items
- align-items
- place-items
- justify-content
- align-content
- place-content
- grid-auto-columns
- grid-auto-rows
- grid-auto-flow
- grid

Properties for the Grid Items

- grid-column-start
- grid-column-end
- grid-row-start
- grid-row-end
- grid-column
- grid-row
- grid-area
- justify-self
- align-self
- place-self

CSS-Tricks Complete Guide to CSS Grid >>

GRID CONTAINER

the flex container needs to have {display:grid;}

grid-template-columns:
either auto, or width in units

grid template-rows:
either auto, or width in units;

grid-column-gap
{grid-rcolumn-gap:50px;}

grid-row-gap
{grid-row-gap:50px;}

grid-gap *(shorthand for column and gap)*

justify-items *(aligning items horizontally)*:
start, end, center, stretch

align-items *(aligning items vertically)*:
start, end, center, stretch

place-items *(shorthand for align-items and justify-items)*

justify-content *(horizontal)*:
space-evenly, space-around, space-between, start, end, center

align-content *(vertical)*:
center, space-evenly, space-around, space-between, start, end

place-content *(shorthand for justify and align-content)*

GRID CONTAINER

the flex container needs to have {display:grid;}

grid-auto-columns:
default column width in units >> 150px etc

grid-auto-rows
default column width in units >> 150px etc

grid-auto-flow
Shorthand for gird-auto-columns and -rows

grid
Shorthand for a bunch of categories
{grid: 150px / auto auto auto;}

grid template-areas: *(defines a page through areas)*:

```
.grid-container {
    grid-template-areas:
        'header header header header header header'
        'menu main main main right right'
        'menu footer footer footer footer footer';
}
```

. means empty space. That's a point

```
.grid-container {
    grid-template-areas:
        'header header header header header header'
        'menu main main main . right'
        'menu footer footer footer footer footer';
}
```

[EXAMPLE >>]

GRID ITEM

 direct children of grid container are grid items

grid-column-start [>>]
grid-column-end [>>]
grid-row-start [>>]
grid-row-end [>>]
Determines a grid item's location;
<number> | <name> | span <number> |
span <name> | auto;

```
.item-a {
    grid-column-start: 2;
    grid-column-end: header;
    grid-row-start: row1-start;
    grid-row-end: span 3;
}
```

z

grid-column *(grid-column-start / grid-column-end)*
grid-row *(grid-row-start / grid-row-end)*
{grid-row: 1 / span 2;}

grid-area *defines grid area by name, that ties back to grid container properties or through span*

```
.item-b {
    grid-area: header;
}

.item-c {
  grid-area: 2 / 1 / span 2 / span 3;
}
/* start on row 2 column 1, and span 2 rows
and 3 columns */
```

GRID ITEM

 direct children of grid container are grid items

justify-self

Aligns a grid item inside a cell along the inline (row) axis >> horizontally;
start | end | center | stretch

align-self

Aligns a grid item inside a cell along the inline (column) axis >> vertically
start | end | center | stretch

place-self

place-self sets both the align-self and justify-self properties in a single declaration

```
.item-a {
    place-self: center stretch;
}
```