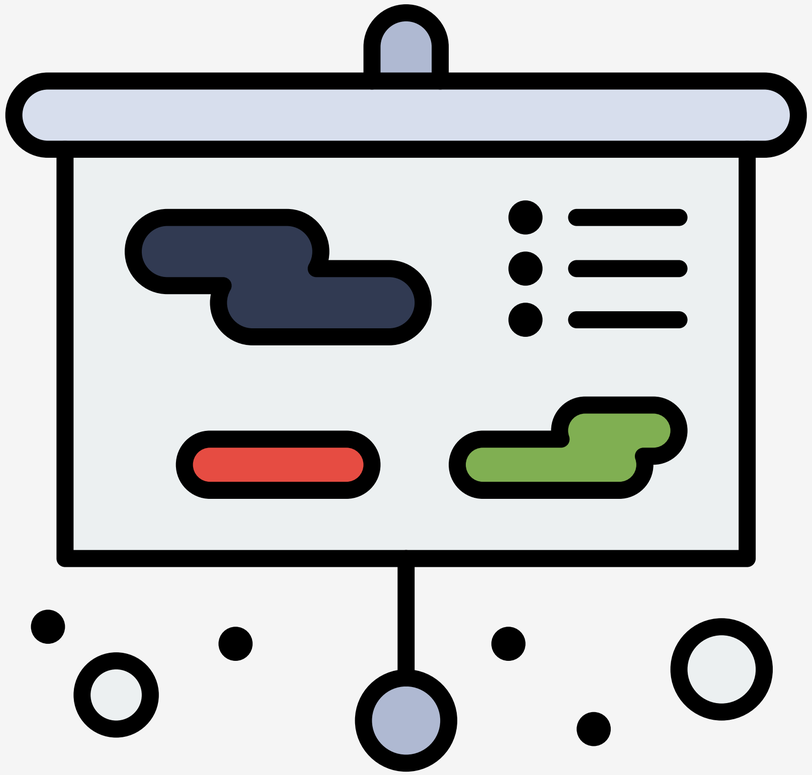
RESPONSIVE
WEB DESIGN II

Lecture 5

TODAY'S TOPICS



- CSS Grid
- **Exercise:** On the Grid

ANNOUNCEMENTS

- Sign-in Sheet
- Recordings

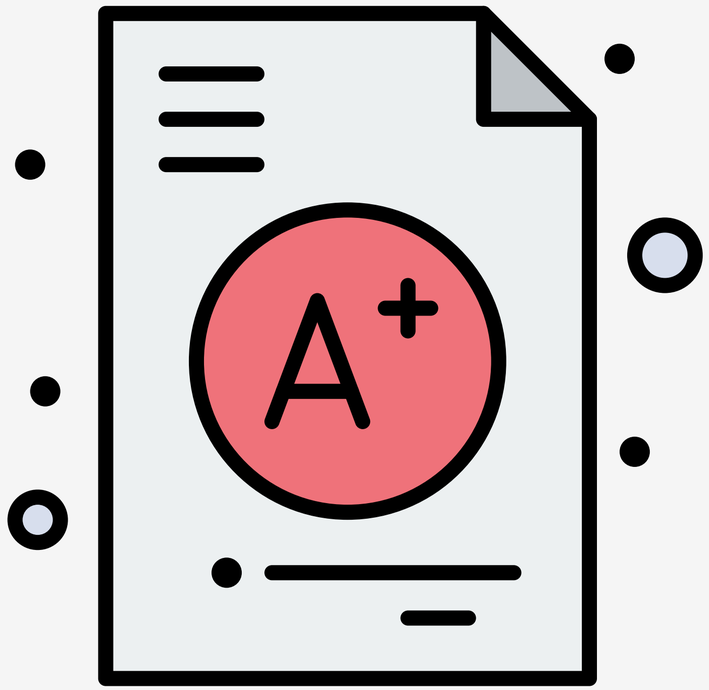
QUESTIONS

CSS GRID

CSS GRID

- CSS Grid is a two-dimensional grid system
- CSS Grid enables alignment of elements into columns and rows

GRID CONTAINER

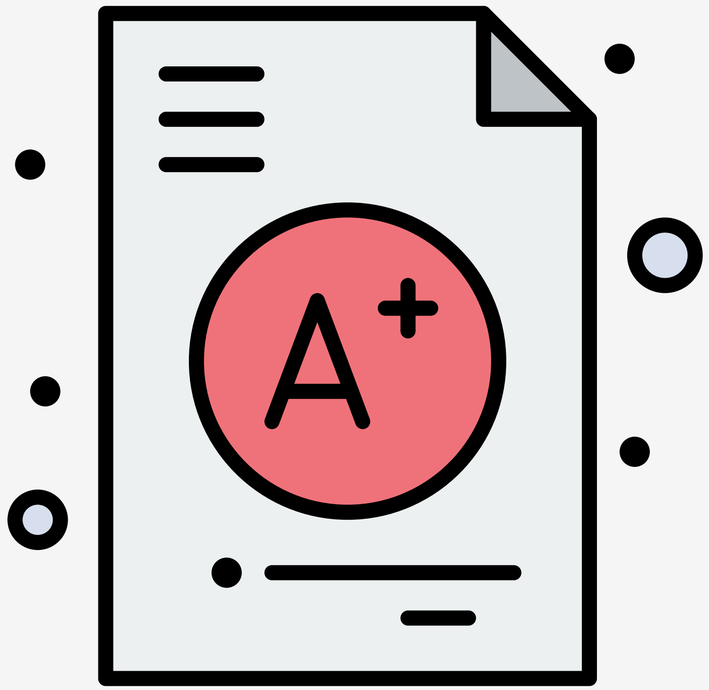


- Grid containers are created using `display: grid`
- Columns are defined using `grid-template-columns`
- Rows are defined using `grid-template-rows`
- The size of a column or row can be defined using common CSS units (`px`, `em`) or use the fraction (`fr`) unit
- The `repeat()` function can be used to create multiple columns or rows of the same size

GRID CONTAINER

```css
/* Creates a grid container */
.grid {
  display: grid;
}

/* Creates a grid with 2 columns */
.grid {
  display: grid;
  grid-template-columns: 50px 100px;
}

/* Creates a 3 x 3 grid */
.grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(4, 1fr);
}
```

IMPLICIT VS EXPLICIT



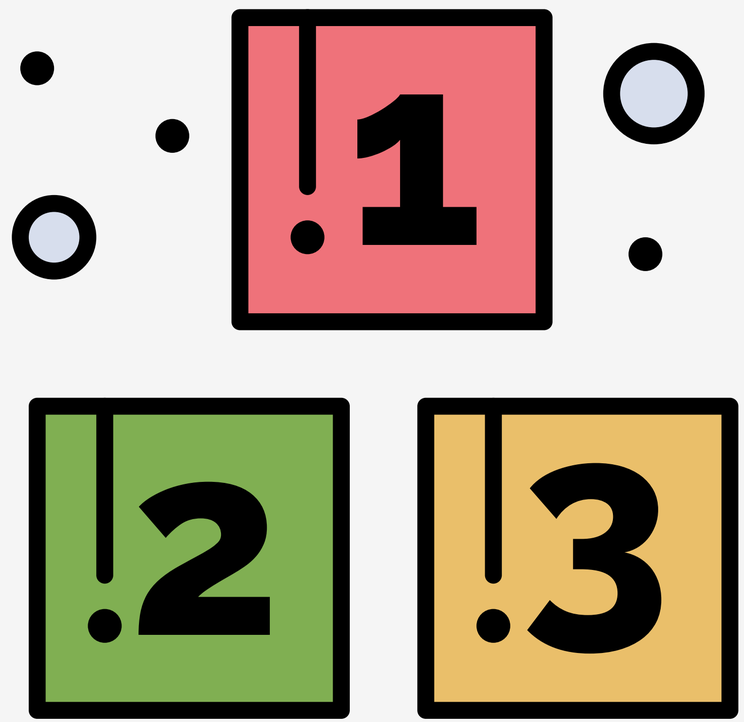
- The explicit grid is the grid that has been defined using `grid-template-columns` and `grid-template-rows`
- The implicit grid is the grid that CSS automatically creates when there more elements then defined cells

GRID CONTAINER

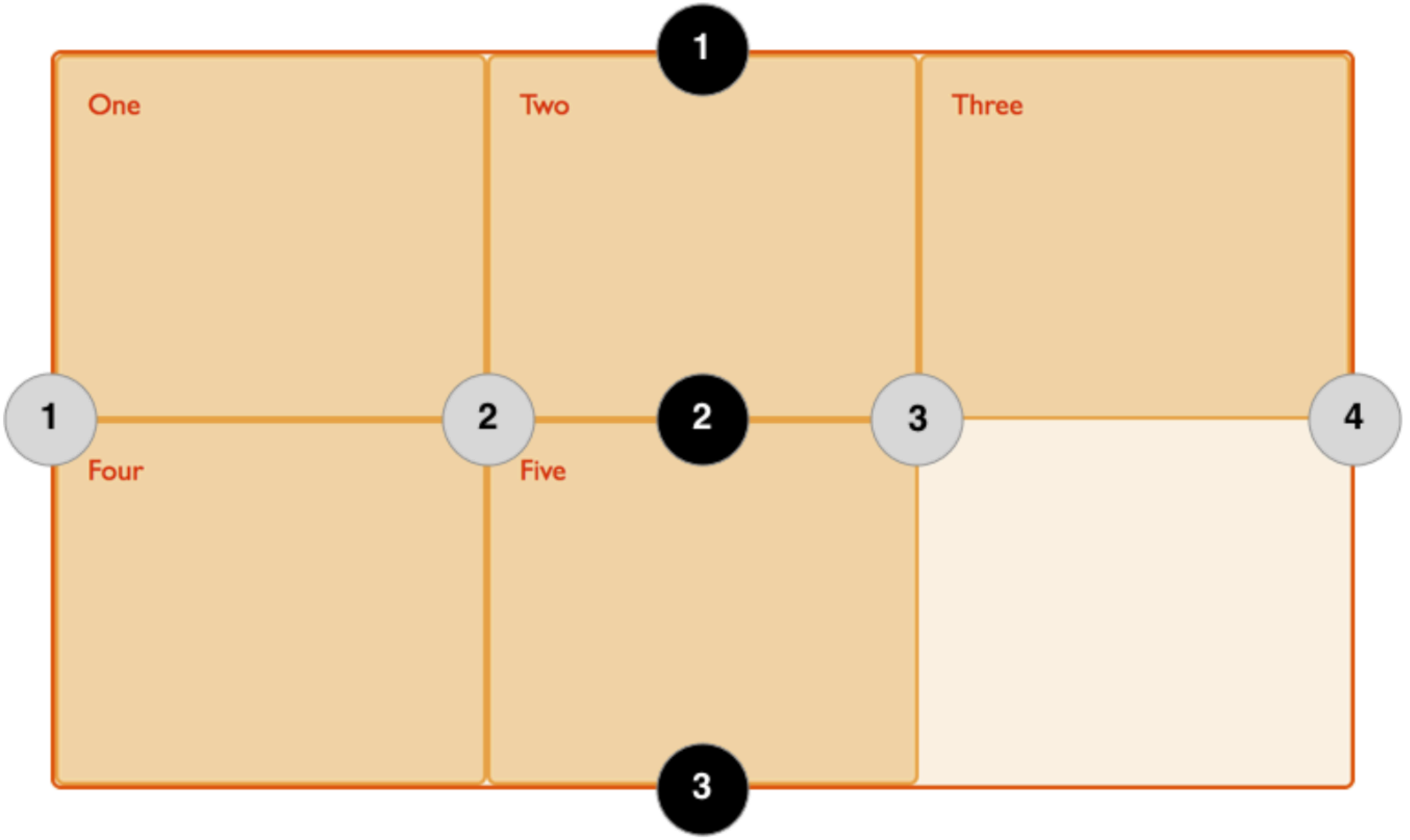
```
/* Assuming 9 elements inside of the grid...*/

/* Implicit grid: 1 x 9 */
.grid {
  display: grid;
}

/* Explicit grid: 2 x 1, Implicit grid: 2 x 4 */
.grid {
  display: grid;
  grid-template-columns: 50px 100px;
}

/* Explicit grid: 3 x 3 */
.grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 1fr);
}
```

POSITIONING

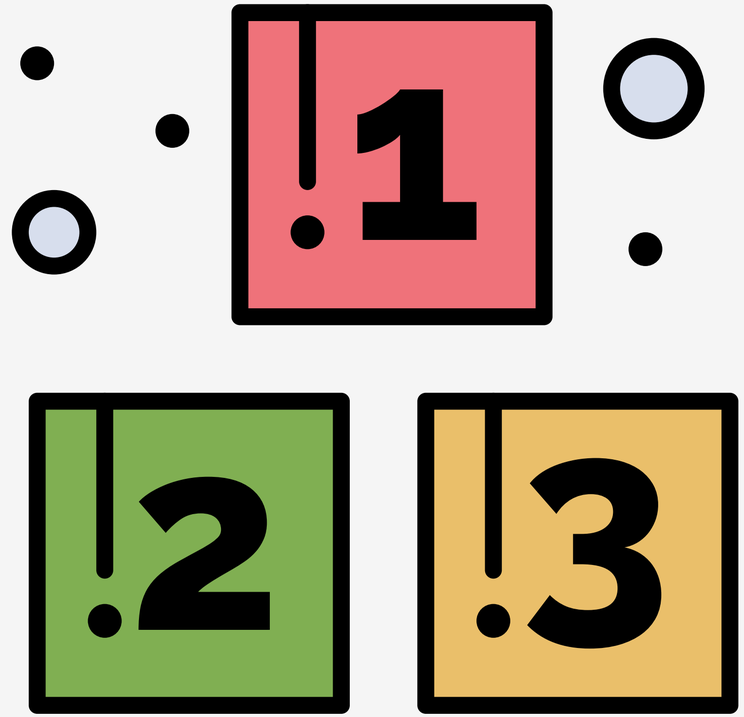
GRID LINES



- Grid lines are the dividing lines that make up the grid structure
- Grid track refers to the space between the two grid lines (each column or row)
- Numbers are assigned to each grid line
- Positive numbers move left to right, top to bottom
- Negative numbers move right to left, bottom to top


1
One
Two
Three
1
2
2
3
4
Four
Five
3

POSITIONING



- Grid items are position using the grid-line numbers
- Each grid item has a start and end point in both horizontal (columns) and vertical (rows)
- The following properties can be used:
  - `grid-column-start`
  - `grid-column-end`
  - `grid-row-start`
  - `grid-row-end`

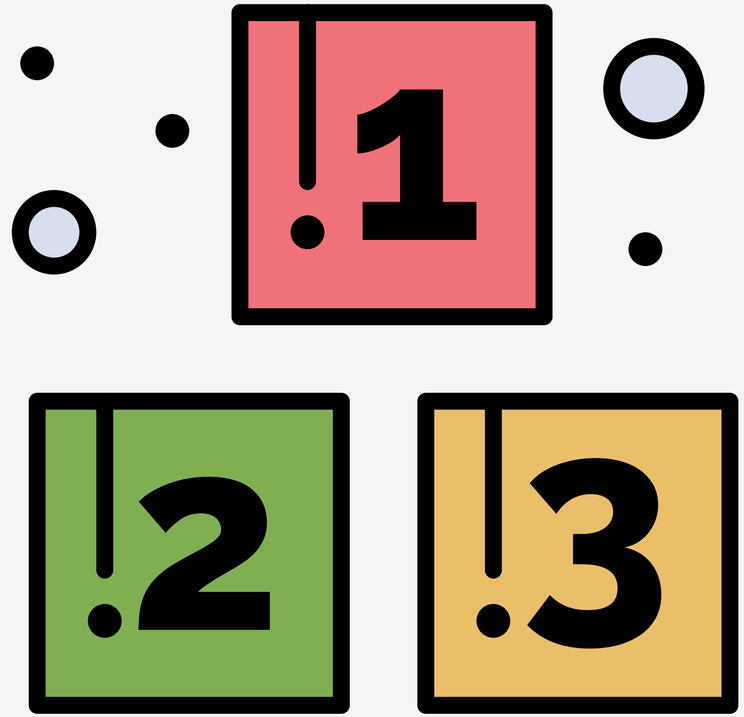
POSITIONING

```css
.grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 1fr);
}

/* item1 will fill 2 columns and 1 row */
.item1 {
  grid-column-start: 1;
  grid-column-end: 3;
  grid-row-start: 1;
  grid-row-end: 2;
}
```

POSITIONING

```css
.grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 1fr);
}

/* SHORTHAND */
/* item1 will fill 2 columns and 1 row */
.item1 {
  grid-column: 1 / 3;
  grid-row: 1;
}
```

POSITIONING WITH SPAN



- The span keyword can be used to provide the number of tracks an item should fill
- The span keyword can be more intuitive than referencing grid lines

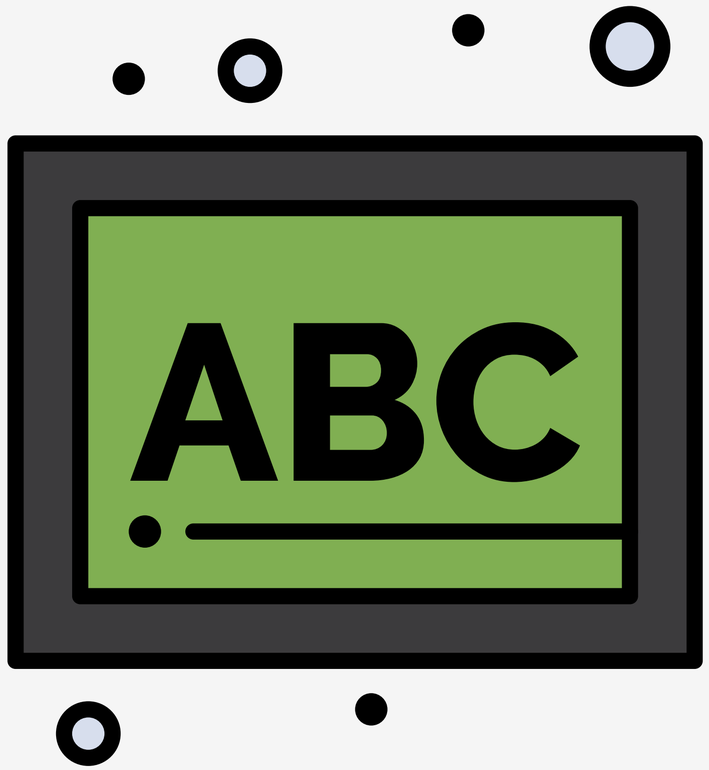
POSITIONING WITH SPAN

```css
.grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 1fr);
}

/* item1 will fill 2 columns and 1 row */
.item1 {
  grid-column: 1 /span 2;
  grid-row: 1;
}
```

HANDS-ON

TEMPLATE AREAS

TEMPLATE AREAS



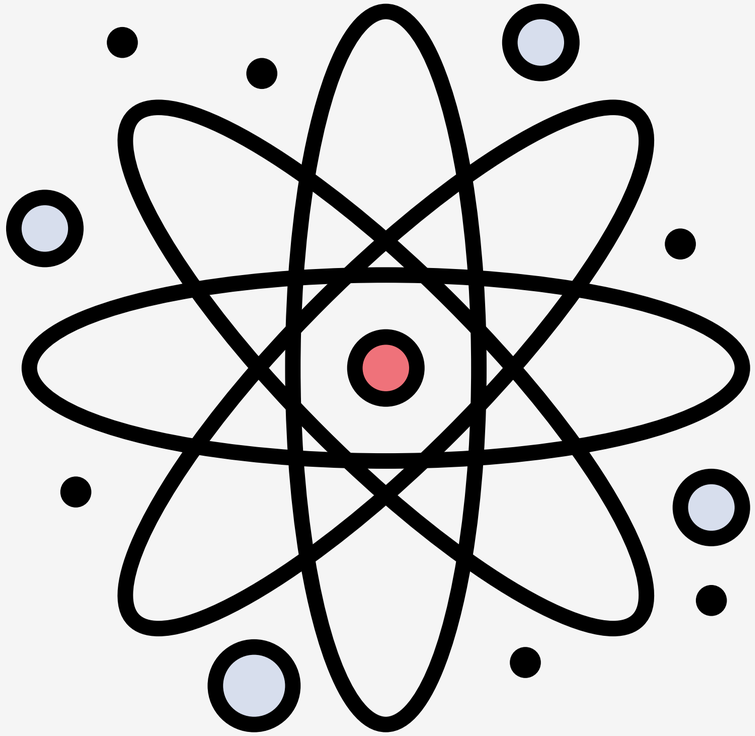
- The `grid-template-areas` property is used to defined named areas of the grid
- Areas are defined using strings that will visually represent the grid and must be in a rectangle
- Area names must start with a letter
- Spaces or tabs can be used to separate grid cells
- A `.` can be used to indicate that no area resides in that grid cell
- Each grid item is assigned a corresponding `grid-area` name

TEMPLATE AREAS

```
.grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 1fr);
  grid-template-area:
  "i1 i1 i2"
  "i3 i3 i2"
  "i3 i3 . "
}

.item1 { grid-area: i1; }
.item2 { grid-area: i2; }
.item3 { grid-area: i3; }
```

ALIGNMENT

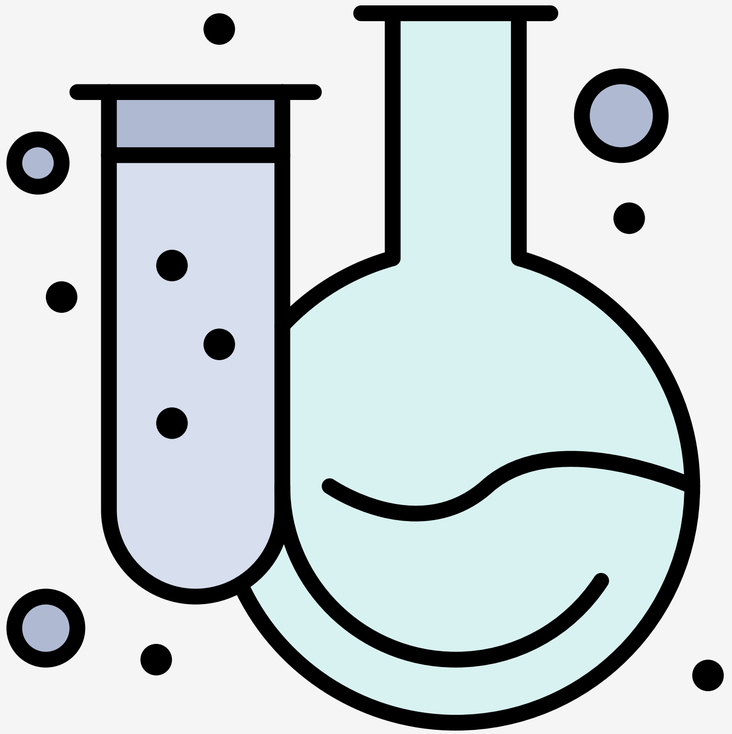
ALIGNMENT



- CSS Grid can control alignment of grid items
- The `justify-items` and `justify-self` properties are used for horizontal alignment
- The `align-items` and `align-self` properties is used for vertical alignment
- The `justify-content` and `align-content` are used to align the grid itself.

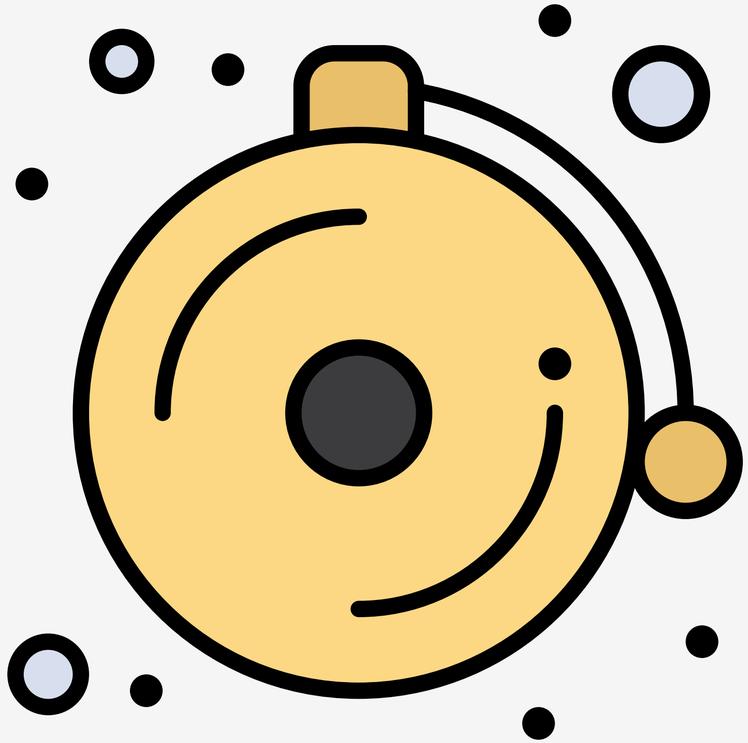
HANDS-ON

ON THE GRID



- *FORK THE PEN!*
- Use CSS Grid to stack the fields as shown
- Use the properties suggested for each field
- Do *NOT* change the HTML
- Submit the URL to your pen
- *DUE:* **Tue. Jan 28 @ 11:59 PM**

NEXT TIME...



- CSS Grid Auto Placement
- CSS Grid Demonstration
- **Midterm Project:** Prototype