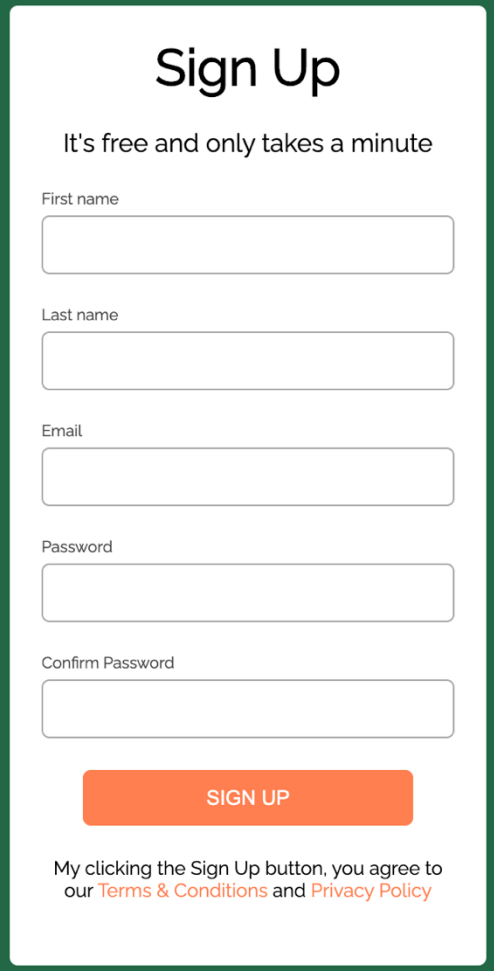
# HTML and CSS

Margit Tennosaar

# Task


Sign Up
It's free and only takes a minute
First name
Last name
Email
Password
Confirm Password
SIGN UP
My clicking the Sign Up button, you agree to our Terms & Conditions and Privacy Policy
Already have an account? Login Here

https://cantunsee.space/

# Units

# Absolute Lengths

| | |
|---|---|
| **px (pixels)** | The most common unit for web design. Fixed size. |
| **cm, mm, in** | Centimeters, millimeters, inches – used for print. |
| **pt, pc** | Points and picas – mainly for print design. |

* Pixels (px) are relative to the viewing device. For low-dpi devices, 1px is one device pixel (dot) of the display. For printers and high resolution screens 1px implies multiple device pixels.

# Relative Lengths

| | |
|---|---|
| % | **Percentage of the parent element's size.** |
| em | Relative to the font size of the parent. |
| rem | Relative to the font size of the root (<html>). |
| vh, vw | Viewport width and height (1vw = 1% of screen width). |
| vmin, vmax | Based on the smaller (vmin) or larger (vmax) viewport dimension. |
| ch | Width of the 0 character in the chosen font. |
| ex | Height of the lowercase x in the font. |

* Viewport = the browser window size. If the viewport is 50cm wide, 1vw = 0.5cm.

```
h1 {
  font-size: 24px; /* Fixed size, does not change */
}

.container {
  width: 80%; /* Takes 80% of its parent */
}

h1 {
  font-size: 2rem; /* 2x the root font size */
}

.box {
  height: 50vh; /* 50% of the viewport height */
}
```

```
html {
  font-size: 16px; }

body {
  font-size: 1rem;
  padding: 2vw; }

.container {
  width: 80%;
  max-width: 1200px;
  margin: auto;
  background: lightgray;
  padding: 2rem;
  text-align: center;}

.title {
  font-size: 2rem;
  margin-bottom: 1rem; }

.text {
  font-size: 1.2rem;
  width: 60%;
  margin: auto; }

.button {
  display: inline-block;
  padding: 1em 2em;
  font-size: 1rem;
  background: steelblue;
  color: white;
  border: none;
  cursor: pointer;}

.section {
  width: 100vw;
  height: 30vh;
  background: lightcoral;
  display: flex;
  align-items: center;
  justify-content: center;}
```

# EM vs REM vs PX

**Select your body font size**

*Conversions based on 16px browser default size*

| Pixels | EMs | Percent | Points |
|---|---|---|---|
| 6px | 0.375em | 37.5% | 5pt |
| 7px | 0.438em | 43.8% | 5pt |
| 8px | 0.500em | 50.0% | 6pt |
| 9px | 0.563em | 56.3% | 7pt |
| 10px | 0.625em | 62.5% | 8pt |
| 11px | 0.688em | 68.8% | 8pt |
| 12px | 0.750em | 75.0% | 9pt |
| 13px | 0.813em | 81.3% | 10pt |
| 14px | 0.875em | 87.5% | 11pt |
| 15px | 0.938em | 93.8% | 11pt |
| 16px | 1.000em | 100.0% | 12pt |
| 17px | 1.063em | 106.3% | 13pt |
| 18px | 1.125em | 112.5% | 14pt |
| 19px | 1.188em | 118.8% | 14pt |
| 20px | 1.250em | 125.0% | 15pt |
| 21px | 1.313em | 131.3% | 16pt |
| 22px | 1.375em | 137.5% | 17pt |
| 23px | 1.438em | 143.8% | 17pt |
| 24px | 1.500em | 150.0% | 18pt |

http://pxtoem.com/

# When to use?

- Use absolute units when you need precise sizing that won't change, like borders or print styling.
- Use relative units for scalable and responsive layouts.
- **Use `rem` for consistent scaling across the entire website.**
- **Use viewport units when designing full-screen sections and dynamic layouts.**

# Display

- Use **block** for structural elements (e.g., , <section>).
- Use **inline** for text elements inside a paragraph.
- Use **inline-block** for buttons or interactive elements.
- Use **flex** when aligning elements dynamically.
- Use **grid** for structured layouts.
- Use **none** to hide elements dynamically.

# Flexbox

Flexbox allows you to align items horizontally and vertically with minimal CSS. It is particularly useful for:

- Centering elements
- Creating dynamic layouts
- Building responsive navigation bars
- Handling equal-height columns

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

# Display: flex;

Container properties

- flex-direction
- flex-wrap
- justify-content
- align-items
- align-content

Items properties

- flex
- order
- align-self

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

```
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
</div>
```

```
.container {
  display: flex;
  background-color: lightgray;
}

.item {
  padding: 20px;
  background-color: steelblue;
  color: white;
  margin: 5px;
}
```

# Flex direction

| | |
|---|---|
| row | Default. Items are placed left to right. |
| row-reverse | Items are placed right to left. |
| column | Items are placed top to bottom. |
| column-reverse | Items are placed bottom to top. |

```
.container {
  display: flex;
  flex-direction: row; /* Main axis: left to right */
}
```

# Justify content

| | |
|---|---|
| flex-start | Items align to the start (default). |
| flex-end | Items align to the end. |
| center | Items are centered. |
| space-between | Space between items. |
| space-around | Space around items. |
| space-evenly | Equal spacing around all items. |

# Align items

| | |
|---|---|
| stretch | Default. Items stretch to fill the container. |
| flex-start | Items align at the top. |
| flex-end | Items align at the bottom. |
| center | Items align at the middle. |
| baseline | Items align by their text baselines. |

# Flex wrap

| | |
|---|---|
| nowrap | Default. Items stay in a single line. |
| wrap | Items wrap to the next line if needed. |
| wrap-reverse | Items wrap in reverse order. |

# Summary of flex

- display: flex; enables flexbox.
- flex-direction controls row/column layout.
- justify-content aligns on the main axis.
- align-items aligns on the cross axis.
- flex-wrap controls item wrapping.

# Homework

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

https://www.markhendriksen.com/interactive-css-flexbox-tool

https://flexboxfroggy.com/