# Lesson:

# Introduction to CSS Units


PW SKILLS

# Topics Covered

- CSS units and their types
- Usage of different types
- Preferred CSS unit and why

## CSS units and their types

CSS units are used to specify the size, length, and other measurements for various properties like font size, margin, padding, and more. There are several types of units available in CSS, and each has its own unique characteristics and uses. For example, if you wanted to set the property margin of a paragraph, you would give it a specific value. This value includes the CSS unit.

Look at this simple example:

```
Unset
p {
    margin: 20px;
}
```

In this case, the **margin** is the **property, 20px**; is the **value**, and **px** (or "pixel") is the CSS **unit**.

Even though it's common to see units like px used, the big question is often, "What's the best unit to use here?"

When considering all the options for which units to use, it's important to consider the two categories of units: absolute and relative.

**Absolute** - These are the **fixed-length units**, and the length expressed using the absolute units will appear as **exactly that size**. It is **not recommended to use on-screen,** because the size of the screen varies too much. The Absolute unit types consist of Pixels(px), centimeters (cm), Millimetres(mm), inches(in), Points(pt), and Picas(pc).

**Relative -** Relative units are good for styling the responsive site because they scale relative to the window size or the parent (depending on the unit). They specify the length, which is relative to another length property.

The Relative unit types consist of percentage(%), parent element's font size(em), view height(vh), viewport minimum(vmin), viewport maximum(vmax), character(ch), X-height(ex)

## Usage of different types

**Absolute unit Usage**

Absolute units can be useful when working on a project where responsiveness is not being considered. For example, desktop apps that can't be resized can be styled to the default dimensions. If the window doesn't scale, you don't need the content either.

**Pixels (px):** It is used to define the measurement in pixels and is one of the most common length units in CSS. Example: "font-size: 16px" sets the font size to 16 pixels.

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

**Centimeter (cm):** It is used to define the measurement in centimeters.

Example: “width: 5 cm," 1 cm is roughly equivalent to 37.8 pixels

```
Unset
 html-element {
      font-size: 5cm;
}
```

**Millimeters (mm)**: It is used to define the measurement in millimeters.

Example: “width: 5mm;” 1 mm is roughly equivalent to 3.78 pixels.

```
Unset
 html-element {
      font-size:5mm;
}
```

**Inches (in):** It is used to define the measurement in inches.
Example: “height: 1in;” 1 in(one inch) is roughly equivalent to 96 pixels or about 2.54 cm.

```
Unset
 html-element {
      font-size:1in;
 }
```

**Points (pt):** A unit of length commonly used in typography to specify the size of the text. One point is equal to 1/72nd of an inch, and the size of a point is typically defined as the height of the letter "M" in a given font.
Example: “ font-size: 12pt;” 1pt is roughly equivalent to 1.3333 pixels, or 1/72th of an inch

```
Unset
 html-element { font-size:  12pt; }
```

**Picas (pc):** It is commonly used in typography to specify the size of the text. One pica is equal to 12 points, or 1/6th of an inch.
Example: “font-size: 1pc”. 1pc is roughly 16 pixels or ⅙ of an inch.

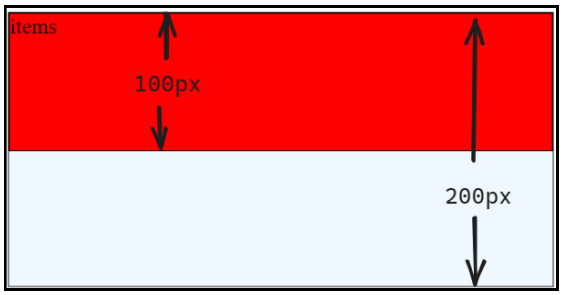
```
Unset
 html-element {
      font-size:1pc;}
```

**Relative unit Usage**

Relative units are widely used in web development because they provide a flexible and responsive way to size elements and create layouts that adapt to different screen sizes and devices.
This unit can be a little more difficult to determine than absolute units, so let's go through your options in detail.

**percentage (%):**

It is used to define the measurement as a percentage that is relative to the parent element's value.
**Example:** If the height of the parent element is 200px, **"height: 50%"** sets the height of the child element to 100 pixels (50% of 200 pixels).




**parent element's font size (em):**

It is relative to the font size of the parent element.
if the font size of the parent element is 16 pixels, then 1em would be equal to 16 pixels. If the font size of the parent element is 20 pixels, then 1em would be equal to 20 pixels.

If the font size of a parent element is not explicitly set in CSS, the child elements using the em unit will inherit the font size from the nearest ancestor that has a font size explicitly defined. If no ancestor has a defined font size, then the browser's default font size will be used whereas the default font size in most web browsers is set to 16px.

**Example:** if the font size of the parent element is 16px, **"height: 2em"** sets the height to 32 pixels (2 times 16 pixels). **"width:4em"** sets the width to 64 pixels (4 times 16 pixels) of the child element.

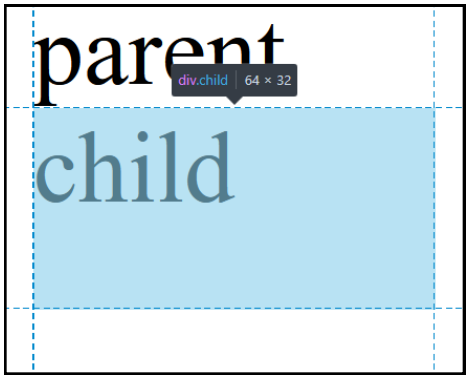
**index.html**

```
Unset
<div class="parent">parent<div class="child">child</div>
</div>
```

**style.css**

```
Unset
.parent { font-size: 16px; }
.child {
  height: 2em;  // 2 * 16 = 32
  width: 4em;   // 4 * 16 = 64 }
```

**Browser output:**




**The root element's font size (rem):**
it is relative to the font- size of the root element. The **<html>** tag is the root element.

**Example:** If the font size of the root element is 16px,
**"font-size: 1.5rem"** sets the font size to 24 pixels. I.e. 1.5 * 16 = 24px

**index.html**

```
Unset
<ul class="rems">
  <li>One</li>
  <li>Two</li>
  <li>Three
    <ul>
      <li>Three A</li>
      <li>Three B
        <ul>
          <li>Three B 2</li>
        </ul>
      </li>
    </ul>
  </li>
</ul>
```

**style.css**

```
Unset
html {
  font-size: 16px;
}
.rems li {
  font-size: 1.5rem; // i.e 16 * 1.5 = 24px
}
```

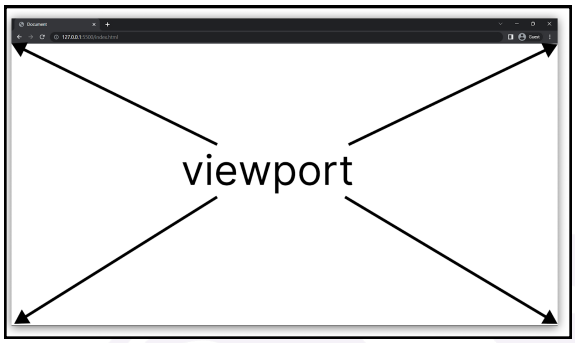
From the above example, the root element is set in the **<html> tag**, with **font size** of **16px**

**Browser output:**

- One
- Two
- Three
  - Three A
  - Three B
    - Three B 2

To understand CSS viewport properties like **vh, vw, vmin**, and **vmax**, let's first understand what the viewport is.

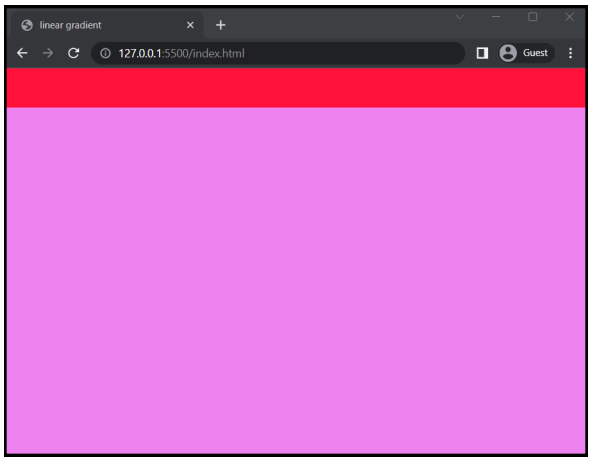
**"Viewport"** is a term used in web development that refers to the visible area of a web page on a device's screen. It's basically the portion of the screen that you can see when you open a website on your phone, tablet, or computer.




On a desktop device, the viewport size matches the **browser's window size, excluding the toolbar and other elements** that aren't part of the page. On a mobile device, the viewport is generally the size of the device's screen.

Let's say we have a webpage that has a header and a main section. The header is set to have a height of 10% of the viewport height, and the main section is set to have a height of 90% of the viewport height.

When we open the webpage in a maximized browser window, the viewport will be the full height and width of the scene. The header will take up 10% of the scene height, and the main section will take up the remaining 90%.




However, if we resize the browser window to be smaller, the viewport will also become smaller. As a result, the header and main section will also become smaller, since their heights are based on the viewport height.

# CSS relative units based on the viewport

**Viewport height (vh):**

It is relative to the height of the viewport.

1 vh = 1%, or 1/100 of the height of the viewport.

**Example: "height: 50vh"** sets the height of the element to 50% of the viewport height.

**index.html**

```
Unset
<div></div>
```

**style.css**

```
Unset
div {
  background-color: violet;
  height: 50vh;
}
```

**Browser output:**

**Viewport width (vw):**
It is relative to the width of the viewport.
1 vw = 1%, or 1/100 of the width of the viewport.
**Example: "width: 50vw"** sets the width of the element to 50% of the viewport width.

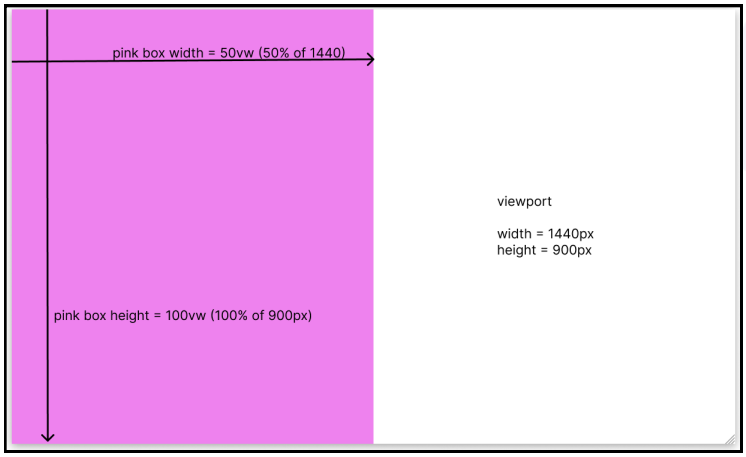
**index.html**

```
Unset
<div></div>
```

**style.css**

```
Unset
div {
  background-color: violet;
  width: 50vw;
  height: 100vh;
}
```

**Browser output:**




# Viewport minimum (vmin):

it is relative to the smaller dimension of the viewport (either the height or width, whichever is smaller) of the viewport.

# Vmin example in mobile:

If the viewport is 400 pixels wide and 800 pixels tall, the smaller dimension would be the width (400 pixels). Therefore, the width and height of the "div" element would be set to 400 pixels (100% of 400 pixels).

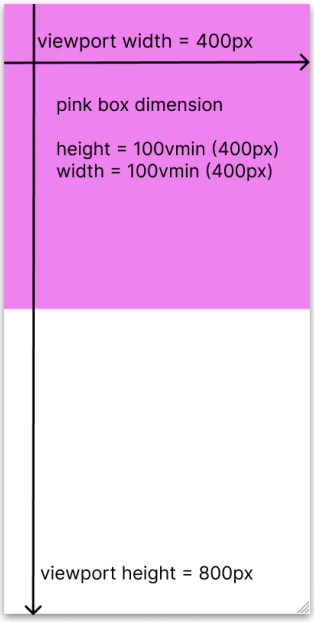
**index.html**

```
Unset
<div><div>
```

**style.css**

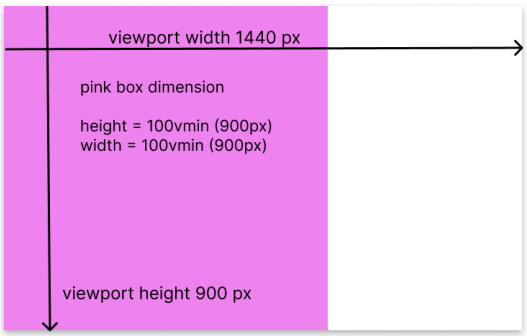
```
Unset
div {
  background-color: pink;
  height: 100vmin;
  width: 100vmin;
}
```

**Browser output:**




**vmin example in desktop:**
If the viewport is **1440 pixels wide** and **900 pixels tall**, the **smaller dimension would be the height** (900 pixels). Therefore, the **width and height of the "div"** element would be **set to 900 pixels** (100% of 900 pixels).

**Browser output:**




**Viewport maximum (vmax):**
It is **relative to the larger dimension of the viewport** (either the height or width, whichever is larger) of the viewport.
1 vmax = 1% or 1/100 of the viewport's larger dimension

**Vmax example in mobile:**
If the viewport is **400 pixels wide** and **800 pixels tall**, the **larger dimension would be the height** (800 pixels). Therefore, the width and height of the "div" element would be set to 400 pixels (50% of 800 pixels).

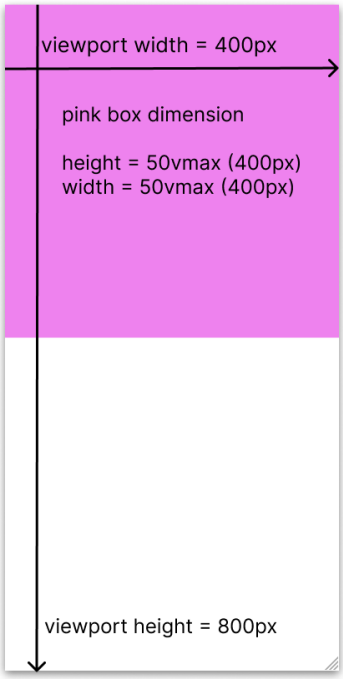
**index.html**

```
Unset
<div><div>
```

**style.css**

```
Unset
div {
  background-color: pink;
  height: 50vmax;
  width: 50vmax; }
```

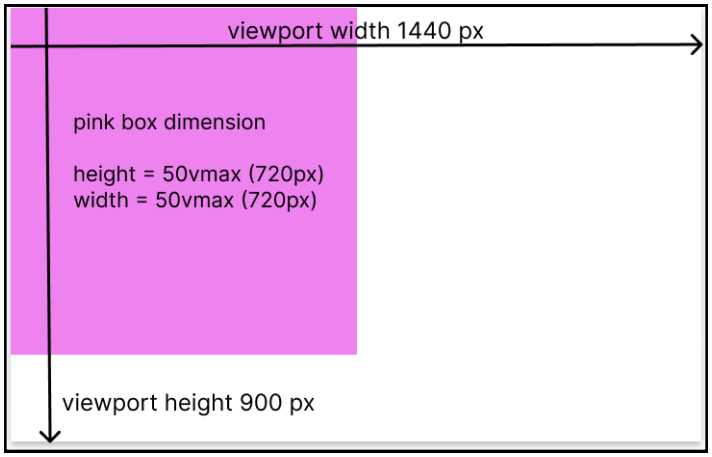
**Browser output:**




**vmax example in desktop:** If the viewport is **1440 pixels wide** and **900 pixels tall,** the **larger dimension would be the width** (1440 pixels). Therefore, the width and height of the "div" element would be set to 720 pixels (50% of 1440 pixels).

**Browser output:**

**Browser output:**




**character (ch):** This unit is based on the width of the "0" character in the font used for the element.
**Example:** "width: 10ch" sets the width of the element to the width of 10 "0" characters.

**X-height (ex):** It is relative to the x-height of the font of the element. It is rarely used, the x-height is determined by the height of the lowercase letter "x".
**Example: "font-size: 2ex"** sets the font size to twice the x-height of the font.

## Preferred CSS unit and why

In CSS the px is a commonly and widely used CSS unit, as it provides a fixed measurement for specifying lengths and sizes and is ideal for elements with dimensions like borders and margins

However, when it comes to preferred CSS units it depends on the specific use cases and design goals of the website being non-responsive and responsive.

For Non-responsive webpages - For designing or creating a web page that is not responsive a pixels unit **(px)** is mostly used as it has the advantage when there is a need for precise control over an element's sizes and positions, especially for designs that don't adapt to different screens

For Responsive webpages - For designing or creating a web page that is responsive, the **Relative** units are always preferred as it allow elements to scale proportionally with the viewport. Some of the most frequently used relative units can be -

- Percentage - Percentage units are relative to the parent container's dimensions. They are commonly used for creating fluid layouts that adapt to different screen sizes.
- Viewport Percentage (vw, vh, vmin, vmax) - These units are relative to the viewport dimensions. They are excellent for responsive designs because they scale with the viewport size.
- em - The **em** unit is relative to the font size of the nearest parent element. It's useful for creating scalable typography and spacing. This is the most used/recommended out of all.
- rem - The **rem** unit is similar to **em**, but it's relative to the root element's font size. This makes it easier to create consistent scaling across the entire webpage.