**Note:** Responsive means making the same web element fit to different screen sizes automatically without creating a completely separate design for each screen size.

**The Two Major Types of Units in CSS:**

CSS units can be classified into two major categories:

* Absolute
* Relative

**Absolute Units:**

These units are the ones whose values are fixed irrespective of any other factors like parent element or viewing window i.e the screen size won't affect the size of the element.

**CSS px:**

px stands for Pixel. Pixels can be defined as 1/96th part of an inch.

**Syntax:**

font-size: Kpx;

**Note:**

* In most syntaxes, we have used the font-size property but any HTML element that has some length like width, height, margin, padding, etc will follow the same syntax for the CSS unit.
* All 'K' used in syntax is any numeric value (non-negative Integer or decimal).

**Use case:** Pixels are widely used in websites to make elements of fixed sizes (ex: in a logo) i.e we don’t want them to change size with screen size variation.

**CSS pt:**

pt stands for point. 1 CSS pt is defined as 1/72th of an inch.

**Syntax:**

font-size: Kpt;

**Use case:** This unit is mainly used in printers for printing the output on paper and not so widely used for on-screen outputs.

**CSS pc:**

pc stands for pica or picas. 1 CSS pt is defined as 1/6th of an inch.

**Syntax:**

font-size: Kpc;

**Use case:** This unit is mainly used in printers for printing the output on paper and not so widely used for on-screen outputs.

**CSS cm:**

This is our good old centimeter that we all know :).

**Syntax:**

font-size: Kcm;

**Use case:** This unit is mainly used in printers for printing the output on paper and not so widely used for on-screen outputs.

**CSS mm:**

mm stands for millimeter.

**Syntax:**

font-size: Kmm;

Where K is any numeric value.

**Use case:** This unit is mainly used in printers for printing the output on paper and not so widely used for on-screen outputs.

**CSS in:**

in stands for inch.

**Syntax:**

font-size: Kin;

**Use case:**This unit is mainly used in printers for printing the output on paper and not so widely used for on-screen outputs.

**Relation between the Absolute Units in CSS:**

|  |  |
| --- | --- |
| **Unit** | **equivalence** |
| Pixel | 1px = 1/96th of an inch |
| Point | 1pt = 1/72th of an inch |
| Pica | 1pc = 1/6th of an inch |
| centimeter | 2.54 cm = 1 inch |
| milimeter | 10mm = 1cm |

**Relative Units:**

Relative units are useful for styling responsive sites because they scale relative to the parent or window size (depending on the unit).

In relative units, we talk in terms of the same property, like, if we are talking about width of an element in relative units then it is relative to the Width of the parent element/viewport.

Relative units, if used correctly, are suitable for making elements scale properly with respect to other things on the same page i.e to make the element responsive.

**CSS %:**

This is the percentage unit. The measurement of the element is relative to the dimensions of the parent. Like if we set the width of an HTML element to be K% then its width is calculated as follows:

***WidthOfHTMLelement=K/100​∗WidthOfParentElement***

**Syntax:**

width: K%;

**Use case:** % is widely used for making responsive sites. This allows us to size HTML elements dynamically relative to the size of the viewing window.

**Example: HTML:**

<body>

<p>Mine is the default font size (16px).</p>

<div class="relativeUnits"><p>Mine is 200% of the default font size</p></div>

</body>

**CSS:**

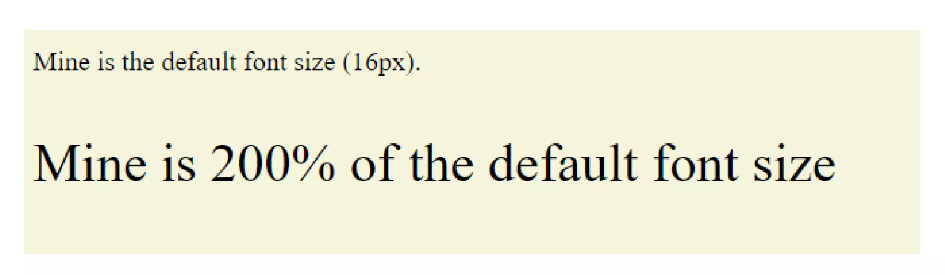
.relativeUnits {

font-size:200%;

}

The above CSS code changes the size of the font inside a div is exactly 32px which is twice relative to the parent (i.e body). The default font size of plain text is 16px.

**Output:**



The size of the font inside a div is exactly 32px which is twice relative to the parent (i.e body). The default font size of plain text is 16px.

**CSS em:**

1em refers to the default size of the property. So precisely, 1em is equivalent to 100%. (i.e 1em in case of plain text would be 16px if the root value is unaltered)

**Interesting fact about em:** em in CSS is another widely used relative unit. It has got the phonetic pronunciation of the English alphabet ‘M’. This is because as per the olden days’ golden rules of typesetting 1em were used to signify the width of the capital letter ‘M’. 2em would mean twice the width of the capital ‘M’. But this is no longer the case in the modern era.

**Syntax:**

font-size: Kem;

**Use case:** This is mostly used to achieve the same values dynamically as is the case with % but applicable specifically to font size.

**Example:**

**HTML:**

<body>

<p>Mine is the default font size (16px).</p>

<div class="relativeUnits"><p>Mine is 2em of the default font size</p></div>

</body>

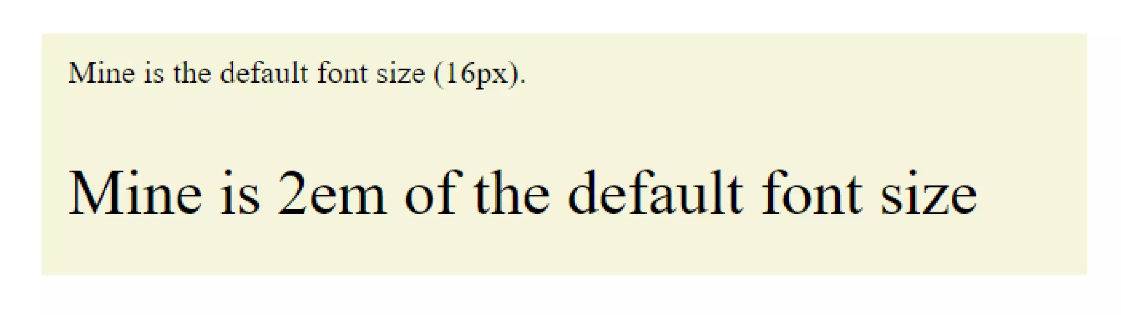
**CSS:**

.relativeUnits {

font-size:2em;

}

**Output:**



**NOTE:** Both % and em are inherited as they are RELATIVE TO THEIR PARENT! So, if we put parent font size as 2em then again we put child font size as 2em it would be added up to 4em (i.e 400% of the default value)! This might cause issues in the case of large CSS codes.

**CSS rem:**

This unit counters the adding-up effect of units like % and em. rem rather stands for Root em.  
This means the browser will ignore all the adjustments made to the parent elements and will scale the HTML element, to which the property is applied, based on the root/default value for that particular element.

**Syntax:**

font-size: Krem;

**Use case:** This is used to achieve the values relative to the root/default value of the HTML elements. This is usually used for font-size property.

**Example:**

**HTML:**

<body>

<p>My font size is set to 40px.</p>

<div class="relativeUnits"><p>I have a font size of 2rem</p></div>

</body>

**CSS:**

body{

font-size: 40px;

}

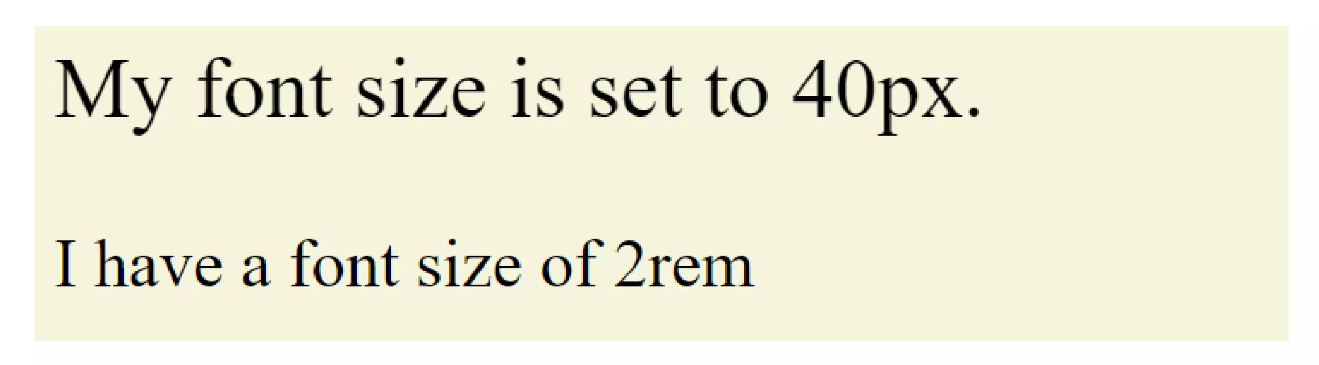
.relativeUnits {

font-size:2rem;

}

In this CSS code, the font-size of the div is set to 2rem (i.e twice the base font size ignoring all changes) and thus is (2 x 16px=) 32px. It ignores the change in font size of 40px made to its parent (here, the body).

**Output:**

****

**CSS ch:**

This unit sizes relative to the width of the digit ‘0’ as in parent. For plain text 1ch=8px (this is only true if the font-size of the parent element is unchanged from the default value of 16px.);

**Syntax:**

font-size: Kch;

**Use case:** This is used to achieve the values relative to the width of the digit ‘0’. This is not very commonly used.

**Example:**

**HTML:**

<body>

<div class="relativeUnits"><p>My font size is set to 2ch</p></div>

</body>

**CSS:**

.relativeUnits {

font-size:2ch;

}

The output is similar to plain text i.e 16px.

**CSS vh:**

This stands for view height. If we want our element to have exactly the same height as your viewport/ view window then use 100vh to denote that.

**Syntax:**

height: Kvh;

**Use case:** Mainly used for pages that occupy the entire height of the viewport.

**CSS vw:**

vw stands for View Width. 100vw means 100% the width of the viewport/view window.

**Syntax:**

width: Kvw;

**Use case:** Mainly used when the element width needs to be framed w.r.t the width of the viewport.

**Example of using vh and vw units:**

**HTML:**

<body>

<div class="vhvw">Example for view height and view width</div>

</body>

**CSS:**

.vhvw {

text-align: center;

height: 100vh;

width: 50vw;

background-color: yellowgreen;

font-size: 2rem;

}

The above CSS code sets the height to 100vh, the width to 50vh, and font-size as 2 rem.

**Output as viewed in the browser:**



**CSS vmin:**

100vmin means the element changes are relative to 100% of the smaller dimension of the viewport (i.e 100% of the minimum of view height and view width).

**Syntax:**

font-size: Kvmin;

**Use case:** Used for responsive views. If we want to fit an element within the viewport we would prioritize the minimum of the two vh and vw.

**CSS vmax:**

100vmax means the element changes are relative to 100% of the larger dimension of the viewport (i.e 100% of maximum of the view height and the view width).

**Syntax:** font-size: Kvmax;

**Use case:**Used for responsive views. If we want to fit an element to scale relative to larger of vh and vw.

**Example of vmin and vmax:**

**HTML:**

<body>

<div class="vminvmax">Example for vmin and vmax</div>

</body>

**CSS:**

.vminvmax {

text-align: center;

height: 70vmax;

width: 50vmin;

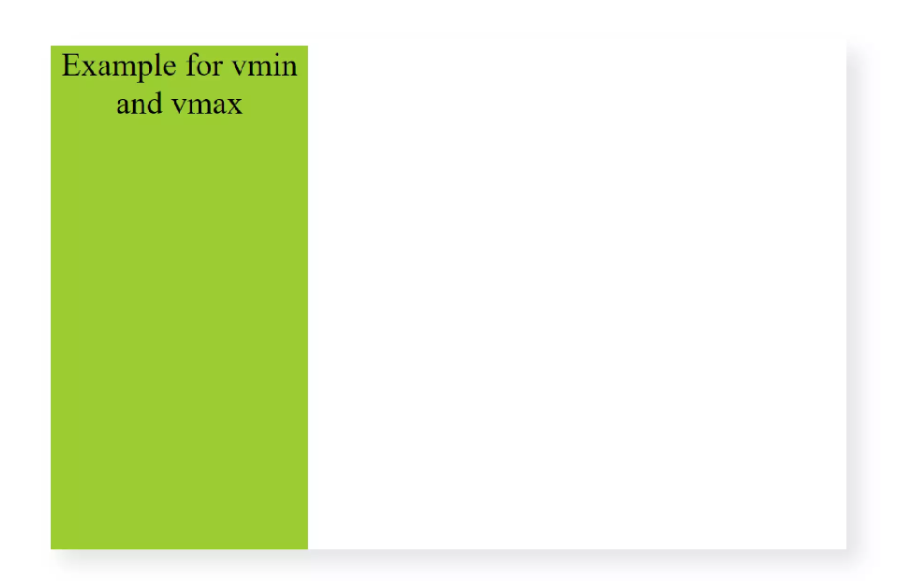
background-color: yellowgreen;

font-size: 2rem;

}

The above CSS code sets the height to 70vmax and the width to 50vmin, alongside a font-size of 2rem.

**Output:**

****

The width of the element is 50% of the minimum of vh and vw (here the smaller one is vh). The height of the element is 70% of the maximum of vh and vh (here the maximum is vw).

**CSS ex:**

Relative to the height of the letter ‘x’, 1ex would be equal to the length from the base of the letter ‘x’ to the intersection point.

**Syntax:**

font-size: Kex;

**Use case:** very rarely used.

**Absolute Unit Vs Relative Unit:**

* The values for absolute units do not depend on any other factor and remain the same irrespective of any change. Whereas in relative units the values depend on some factor and changing that factor will cause the value to change.
* Relative units are preferred over absolute units for making responsive sites because when HTML elements are set to relative unit they scale with respect to the size of the viewing window thus the size of elements will change automatically to fit the size of the screen i.e. become responsive.