Responsive Design | WeWeb Documentation

|  |  |
| --- | --- |
| URL source | <https://docs.weweb.io/css-and-styling/responsive-design.html> |
| Date de scraping | 2025-04-08T15:53:12.454788+00:00 |

## Responsive design ​

Responsive design consists in designing web applications that adapt to smaller devices, like mobile and tablets.  
WeWeb handles responsive design out of the box.

## Three breakpoints ​

In WeWeb, you can customize the design at three breakpoint levels: desktop, tablet, and mobile:

### Images

<https://docs.weweb.io/assets/responsive1.Ck9PCVzg.gif>

## Top down approach ​

When you customize a CSS property on a breakpoint, that style will be applied to all the smaller breakpoints:  
setting a font color on desktop will also apply on tablets and mobile devices setting a font weight on tablet will also apply on mobile, but not on desktop setting a font size on mobile will only apply on mobile, not on desktop or tablet

## Add responsive style ​

To design responsively, switch between breakpoints in the top navigation menu, then change a style in the right sidebar.  
You'll be able to see responsive styles when hovering them. A green icon will appear next to the property name and, if you click on it, you'll see on what breakpoint the change was made:

### Images

<https://docs.weweb.io/assets/responsive3.KjXw4_LD.png>

<https://docs.weweb.io/assets/responsive4.p5uzJehJ.png>

## Delete responsive style ​

If you hover a CSS style and click on one of the tablet or mobile icons, you'll remove the responsive style for that breakpoint:

### Images

<https://docs.weweb.io/assets/responsive2.DeSwRCPW.gif>

## Make your designs responsive ​

To make your designs adapt to different screen sizes, ensure that your layout and components adjust smoothly across all breakpoints. Here are some best practices to help you achieve responsive designs in WeWeb:

## Use Relative Units ​

Instead of using fixed pixel values ( px ) for widths, heights, margins, and paddings, opt for relative units like percentages ( % ), viewport width ( vw ), viewport height ( vh ), or em and rem . This allows elements to scale proportionally with the screen size.

## Leverage Flexbox and Grid Layouts ​

Utilize CSS Flexbox and Grid to create flexible and responsive layouts:  
Flexbox is ideal for one-dimensional layouts (rows or columns). Grid is perfect for two-dimensional layouts (both rows and columns). These layout models adapt naturally to different screen sizes, ensuring your content remains organized.

## Avoid Fixed Widths and Heights ​

Setting fixed widths and heights can cause elements to overflow or not resize properly on smaller screens. Instead, use max-width , min-width , max-height , and min-height properties to allow elements to resize within specified limits.

## Check Parent Containers ​

If you notice that only part of your page resizes while another part remains the same, it might be because:  
Fixed Dimensions on Parent Elements : Ensure parent containers don't have fixed widths or heights that prevent child elements from resizing. Overflow Properties : Check if overflow is set to hidden , which can restrict content from adjusting its size. Positioning Issues : Elements with position: absolute or position: fixed might not respond to changes in screen size.

## Hide elements on mobile ​

To not display an element on tablet or mobile, DO NOT delete it .  
Instead, we invite you to follow standard web development practices and edit:  
CSS display property, or conditional rendering setting.