Responsive Web Design - The Viewport

[❮ Previous](https://www.w3schools.com/css/css_rwd_intro.asp)[Next ❯](https://www.w3schools.com/css/css_rwd_grid.asp)

What is The Viewport?

The viewport is the user's visible area of a web page.

The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.

Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.

Then, when we started surfing the internet using tablets and mobile phones, fixed size web pages were too large to fit the viewport. To fix this, browsers on those devices scaled down the entire web page to fit the screen.

This was not perfect!! But a quick fix.

Setting The Viewport

HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.

You should include the following <meta> viewport element in all your web pages:

<meta name="viewport" content="width=device-width, initial-scale=1.0">

This gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:

[[](https://www.w3schools.com/css/example_withoutviewport.htm)  
  
**Without the viewport meta tag**](https://www.w3schools.com/css/example_withoutviewport.htm)

[[](https://www.w3schools.com/css/example_withviewport.htm)  
  
**With the viewport meta tag**](https://www.w3schools.com/css/example_withviewport.htm)

**Tip:** If you are browsing this page with a phone or a tablet, you can click on the two links above to see the difference.

Size Content to The Viewport

Users are used to scroll websites vertically on both desktop and mobile devices - but not horizontally!

So, if the user is forced to scroll horizontally, or zoom out, to see the whole web page it results in a poor user experience.

Some additional rules to follow:

**1. Do NOT use large fixed width elements -**For example, if an image is displayed at a width wider than the viewport it can cause the viewport to scroll horizontally. Remember to adjust this content to fit within the width of the viewport.

**2. Do NOT let the content rely on a particular viewport width to render well** - Since screen dimensions and width in CSS pixels vary widely between devices, content should not rely on a particular viewport width to render well.

**3. Use CSS media queries to apply different styling for small and large screens** - Setting large absolute CSS widths for page elements will cause the element to be too wide for the viewport on a smaller device. Instead, consider using relative width values, such as width: 100%. Also, be careful of using large absolute positioning values. It may cause the element to fall outside the viewport on small devices.

# Responsive Web Design - Grid-View

[❮ Previous](https://www.w3schools.com/css/css_rwd_viewport.asp)[Next ❯](https://www.w3schools.com/css/css_rwd_mediaqueries.asp)

## What is a Grid-View?

Many web pages are based on a grid-view, which means that the page is divided into columns:

Using a grid-view is very helpful when designing web pages. It makes it easier to place elements on the page.

A responsive grid-view often has 12 columns, and has a total width of 100%, and will shrink and expand as you resize the browser window.

[Example: Responsive Grid View](https://www.w3schools.com/css/tryresponsive_grid.htm)

ADVERTISEMENT

## Building a Responsive Grid-View

Lets start building a responsive grid-view.

First ensure that all HTML elements have the box-sizing property set to border-box. This makes sure that the padding and border are included in the total width and height of the elements.

Add the following code in your CSS:

\* {  
  box-sizing: border-box;  
}

Read more about the box-sizing property in our [CSS Box Sizing](https://www.w3schools.com/css/css3_box-sizing.asp) chapter.

The following example shows a simple responsive web page, with two columns:

25%

75%

### **Example**

.menu {  
  width: 25%;  
  float: left;  
}  
.main {  
  width: 75%;  
  float: left;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_webpage)

The example above is fine if the web page only contains two columns.

However, we want to use a responsive grid-view with 12 columns, to have more control over the web page.

First we must calculate the percentage for one column: 100% / 12 columns = 8.33%.

Then we make one class for each of the 12 columns, class="col-" and a number defining how many columns the section should span:

### **CSS:**

.col-1 {width: 8.33%;}  
.col-2 {width: 16.66%;}  
.col-3 {width: 25%;}  
.col-4 {width: 33.33%;}  
.col-5 {width: 41.66%;}  
.col-6 {width: 50%;}  
.col-7 {width: 58.33%;}  
.col-8 {width: 66.66%;}  
.col-9 {width: 75%;}  
.col-10 {width: 83.33%;}  
.col-11 {width: 91.66%;}  
.col-12 {width: 100%;}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_cols)

 All these columns should be floating to the left, and have a padding of 15px:

### **CSS:**

[class\*="col-"] {  
  float: left;  
  padding: 15px;  
  border: 1px solid red;  
}

Each row should be wrapped in a <div>. The number of columns inside a row should always add up to 12:

### **HTML:**

<div class="row">  
  <div class="col-3">...</div> <!-- 25% -->  
  <div class="col-9">...</div> <!-- 75% -->  
</div>

The columns inside a row are all floating to the left, and are therefore taken out of the flow of the page, and other elements will be placed as if the columns do not exist. To prevent this, we will add a style that clears the flow:

### **CSS:**

.row::after {  
  content: "";  
  clear: both;  
  display: table;  
}

We also want to add some styles and colors to make it look better:

### **Example**

html {  
  font-family: "Lucida Sans", sans-serif;  
}  
  
.header {  
  background-color: #9933cc;  
  color: #ffffff;  
  padding: 15px;  
}  
  
.menu ul {  
  list-style-type: none;  
  margin: 0;  
  padding: 0;  
}  
  
.menu li {  
  padding: 8px;  
  margin-bottom: 7px;  
  background-color :#33b5e5;  
  color: #ffffff;  
  box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);  
}  
  
.menu li:hover {  
  background-color: #0099cc;  
}

# Responsive Web Design - Media Queries

[❮ Previous](https://www.w3schools.com/css/css_rwd_grid.asp)[Next ❯](https://www.w3schools.com/css/css_rwd_images.asp)

## What is a Media Query?

Media query is a CSS technique introduced in CSS3.

It uses the @media rule to include a block of CSS properties only if a certain condition is true.

### **Example**

If the browser window is 600px or smaller, the background color will be lightblue:

@media only screen and (max-width: 600px) {  
  body {  
    background-color: lightblue;  
  }  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_mediaquery)

## Add a Breakpoint

Earlier in this tutorial we made a web page with rows and columns, and it was responsive, but it did not look good on a small screen.

Media queries can help with that. We can add a breakpoint where certain parts of the design will behave differently on each side of the breakpoint.

**  
Desktop**

**  
Phone**

Use a media query to add a breakpoint at 768px:

### **Example**

When the screen (browser window) gets smaller than 768px, each column should have a width of 100%:

/\* For desktop: \*/  
.col-1 {width: 8.33%;}  
.col-2 {width: 16.66%;}  
.col-3 {width: 25%;}  
.col-4 {width: 33.33%;}  
.col-5 {width: 41.66%;}  
.col-6 {width: 50%;}  
.col-7 {width: 58.33%;}  
.col-8 {width: 66.66%;}  
.col-9 {width: 75%;}  
.col-10 {width: 83.33%;}  
.col-11 {width: 91.66%;}  
.col-12 {width: 100%;}  
  
@media only screen and (max-width: 768px) {  
  /\* For mobile phones: \*/  
  [class\*="col-"] {  
    width: 100%;  
  }  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_breakpoints)

ADVERTISEMENT

## Always Design for Mobile First

Mobile First means designing for mobile before designing for desktop or any other device (This will make the page display faster on smaller devices).

This means that we must make some changes in our CSS.

Instead of changing styles when the width gets smaller than 768px, we should change the design when the width gets larger than 768px. This will make our design Mobile First:

### **Example**

/\* For mobile phones: \*/  
[class\*="col-"] {  
  width: 100%;  
}  
  
@media only screen and (min-width: 768px) {  
  /\* For desktop: \*/  
  .col-1 {width: 8.33%;}  
  .col-2 {width: 16.66%;}  
  .col-3 {width: 25%;}  
  .col-4 {width: 33.33%;}  
  .col-5 {width: 41.66%;}  
  .col-6 {width: 50%;}  
  .col-7 {width: 58.33%;}  
  .col-8 {width: 66.66%;}  
  .col-9 {width: 75%;}  
  .col-10 {width: 83.33%;}  
  .col-11 {width: 91.66%;}  
  .col-12 {width: 100%;}  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_mobilefirst)

## Another Breakpoint

You can add as many breakpoints as you like.

We will also insert a breakpoint between tablets and mobile phones.

**  
Desktop**

**  
Tablet**

**  
Phone**

We do this by adding one more media query (at 600px), and a set of new classes for devices larger than 600px (but smaller than 768px):

### **Example**

Note that the two sets of classes are almost identical, the only difference is the name (col- and col-s-):

/\* For mobile phones: \*/  
[class\*="col-"] {  
  width: 100%;  
}  
  
@media only screen and (min-width: 600px) {  
  /\* For tablets: \*/  
  .col-s-1 {width: 8.33%;}  
  .col-s-2 {width: 16.66%;}  
  .col-s-3 {width: 25%;}  
  .col-s-4 {width: 33.33%;}  
  .col-s-5 {width: 41.66%;}  
  .col-s-6 {width: 50%;}  
  .col-s-7 {width: 58.33%;}  
  .col-s-8 {width: 66.66%;}  
  .col-s-9 {width: 75%;}  
  .col-s-10 {width: 83.33%;}  
  .col-s-11 {width: 91.66%;}  
  .col-s-12 {width: 100%;}  
}  
  
@media only screen and (min-width: 768px) {  
  /\* For desktop: \*/  
  .col-1 {width: 8.33%;}  
  .col-2 {width: 16.66%;}  
  .col-3 {width: 25%;}  
  .col-4 {width: 33.33%;}  
  .col-5 {width: 41.66%;}  
  .col-6 {width: 50%;}  
  .col-7 {width: 58.33%;}  
  .col-8 {width: 66.66%;}  
  .col-9 {width: 75%;}  
  .col-10 {width: 83.33%;}  
  .col-11 {width: 91.66%;}  
  .col-12 {width: 100%;}  
}

It might seem odd that we have two sets of identical classes, but it gives us the opportunity in HTML, to decide what will happen with the columns at each breakpoint:

### **HTML Example**

**For desktop:**

The first and the third section will both span 3 columns each. The middle section will span 6 columns.

**For tablets:**

The first section will span 3 columns, the second will span 9, and the third section will be displayed below the first two sections, and it will span 12 columns:

<div class="row">  
  <div class="col-3 col-s-3">...</div>  
  <div class="col-6 col-s-9">...</div>  
  <div class="col-3 col-s-12">...</div>  
</div>

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_col-s)

## Typical Device Breakpoints

There are tons of screens and devices with different heights and widths, so it is hard to create an exact breakpoint for each device. To keep things simple you could target five groups:

### **Example**

/\* Extra small devices (phones, 600px and down) \*/  
@media only screen and (max-width: 600px) {...}  
  
/\* Small devices (portrait tablets and large phones, 600px and up) \*/  
@media only screen and (min-width: 600px) {...}  
  
/\* Medium devices (landscape tablets, 768px and up) \*/  
@media only screen and (min-width: 768px) {...}  
  
/\* Large devices (laptops/desktops, 992px and up) \*/  
@media only screen and (min-width: 992px) {...}  
  
/\* Extra large devices (large laptops and desktops, 1200px and up) \*/  
@media only screen and (min-width: 1200px) {...}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_mediaquery_breakpoints)

## Orientation: Portrait / Landscape

Media queries can also be used to change layout of a page depending on the orientation of the browser.

You can have a set of CSS properties that will only apply when the browser window is wider than its height, a so called "Landscape" orientation:

### **Example**

The web page will have a lightblue background if the orientation is in landscape mode:

@media only screen and (orientation: landscape) {  
  body {  
    background-color: lightblue;  
  }  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_mediaquery_orientation2)

## Hide Elements With Media Queries

Another common use of media queries, is to hide elements on different screen sizes:

I will be hidden on small screens.

### **Example**

/\* If the screen size is 600px wide or less, hide the element \*/  
@media only screen and (max-width: 600px) {  
  div.example {  
    display: none;  
  }  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss_mediaqueries_hide2)

## Change Font Size With Media Queries

You can also use media queries to change the font size of an element on different screen sizes:

# Variable Font Size.

### **Example**

/\* If the screen size is 601px or more, set the font-size of <div> to 80px \*/  
@media only screen and (min-width: 601px) {  
  div.example {  
    font-size: 80px;  
  }  
}  
  
/\* If the screen size is 600px or less, set the font-size of <div> to 30px \*/  
@media only screen and (max-width: 600px) {  
  div.example {  
    font-size: 30px;  
  }  
}

# Responsive Web Design - Images

[❮ Previous](https://www.w3schools.com/css/css_rwd_mediaqueries.asp)[Next ❯](https://www.w3schools.com/css/css_rwd_videos.asp)



Resize the browser window to see how the image scales to fit the page.

## Using The width Property

If the width property is set to a percentage and the height property is set to "auto", the image will be responsive and scale up and down:

### **Example**

img {  
  width: 100%;  
  height: auto;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image2)

Notice that in the example above, the image can be scaled up to be larger than its original size. A better solution, in many cases, will be to use the max-width property instead.

## Using The max-width Property

If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

### **Example**

img {  
  max-width: 100%;  
  height: auto;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image)

## Add an Image to The Example Web Page

### **Example**

img {  
  width: 100%;  
  height: auto;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image3)

ADVERTISEMENT

## Background Images

Background images can also respond to resizing and scaling.

Here we will show three different methods:

1. If the background-size property is set to "contain", the background image will scale, and try to fit the content area. However, the image will keep its aspect ratio (the proportional relationship between the image's width and height):

Here is the CSS code:

### **Example**

div {  
  width: 100%;  
  height: 400px;  
  background-image: url('img\_flowers.jpg');  
  background-repeat: no-repeat;  
  background-size: contain;  
  border: 1px solid red;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image_background1)

2. If the background-size property is set to "100% 100%", the background image will stretch to cover the entire content area:

Here is the CSS code:

### **Example**

div {  
  width: 100%;  
  height: 400px;  
  background-image: url('img\_flowers.jpg');  
  background-size: 100% 100%;  
  border: 1px solid red;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image_background2)

3. If the background-size property is set to "cover", the background image will scale to cover the entire content area. Notice that the "cover" value keeps the aspect ratio, and some part of the background image may be clipped:

Here is the CSS code:

### **Example**

div {  
  width: 100%;  
  height: 400px;  
  background-image: url('img\_flowers.jpg');  
  background-size: cover;  
  border: 1px solid red;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image_background3)

## Different Images for Different Devices

A large image can be perfect on a big computer screen, but useless on a small device. Why load a large image when you have to scale it down anyway? To reduce the load, or for any other reasons, you can use media queries to display different images on different devices.

Here is one large image and one smaller image that will be displayed on different devices:





### **Example**

/\* For width smaller than 400px: \*/  
body {  
  background-image: url('img\_smallflower.jpg');  
}  
  
/\* For width 400px and larger: \*/  
@media only screen and (min-width: 400px) {  
  body {  
    background-image: url('img\_flowers.jpg');  
  }  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image_mediaq)

You can use the media query min-device-width, instead of min-width, which checks the device width, instead of the browser width. Then the image will not change when you resize the browser window:

### **Example**

/\* For devices smaller than 400px: \*/  
body {  
  background-image: url('img\_smallflower.jpg');  
}  
  
/\* For devices 400px and larger: \*/  
@media only screen and (min-device-width: 400px) {  
  body {  
    background-image: url('img\_flowers.jpg');  
  }  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image_mediaq2)

## The HTML <picture> Element

The HTML <picture> element gives web developers more flexibility in specifying image resources.

The most common use of the <picture> element will be for images used in responsive designs. Instead of having one image that is scaled up or down based on the viewport width, multiple images can be designed to more nicely fill the browser viewport.

The <picture> element works similar to the <video> and <audio> elements. You set up different sources, and the first source that fits the preferences is the one being used:

### **Example**

<picture>  
  <source srcset="img\_smallflower.jpg" media="(max-width: 400px)">  
  <source srcset="img\_flowers.jpg">  
  <img src="img\_flowers.jpg" alt="Flowers">  
</picture>

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_image_picture)

The srcset attribute is required, and defines the source of the image.

The media attribute is optional, and accepts the media queries you find in [CSS @media rule](https://www.w3schools.com/cssref/css3_pr_mediaquery.asp).

You should also define an <img> element for browsers that do not support the <picture> element.

# Responsive Web Design - Videos

[❮ Previous](https://www.w3schools.com/css/css_rwd_images.asp)[Next ❯](https://www.w3schools.com/css/css_rwd_frameworks.asp)

## Using The width Property

If the width property is set to 100%, the video player will be responsive and scale up and down:

### **Example**

video {  
  width: 100%;  
  height: auto;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_video2)

Notice that in the example above, the video player can be scaled up to be larger than its original size. A better solution, in many cases, will be to use the max-width property instead.

## Using The max-width Property

If the max-width property is set to 100%, the video player will scale down if it has to, but never scale up to be larger than its original size:

### **Example**

video {  
  max-width: 100%;  
  height: auto;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_video)

## Add a Video to the Example Web Page

We want to add a video in our example web page. The video will be resized to always take up all the available space:

### **Example**

video {  
  width: 100%;  
  height: auto;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_video3)