Responsive Designs

INTRODUCTION

Responsive Design is a graphic design approach that is used to create content that can adjust to different screen sizes so that it looks good on devices with different screen sizes.

You need responsive designs because you want your website to look good on different devices without having to code different websites for each device type. It offers an optimized browsing experience. Your website will look great and work well on a desktop (or laptop), a tablet, and a mobile phone's browser as it automatically scales its content and elements to match the screen size on which it is viewed.

You can achieve this by three methods:

1. Responsive Design – This means the browser will show the same HTML code, from the same URL, regardless of the device on which it's being viewed.
2. Dedicated Website for each device – You can make a dedicated separate website for mobile-only devices, which will serve the same content, on a different URL.
3. Dynamic Serving – With the use of Dynamic serving, your website will use different HTML and CSS codes for different users, depending on their device type, on the same URL.

Here we will see Responsive Design using HTML and CSS.

Measurement units for Responsive Design

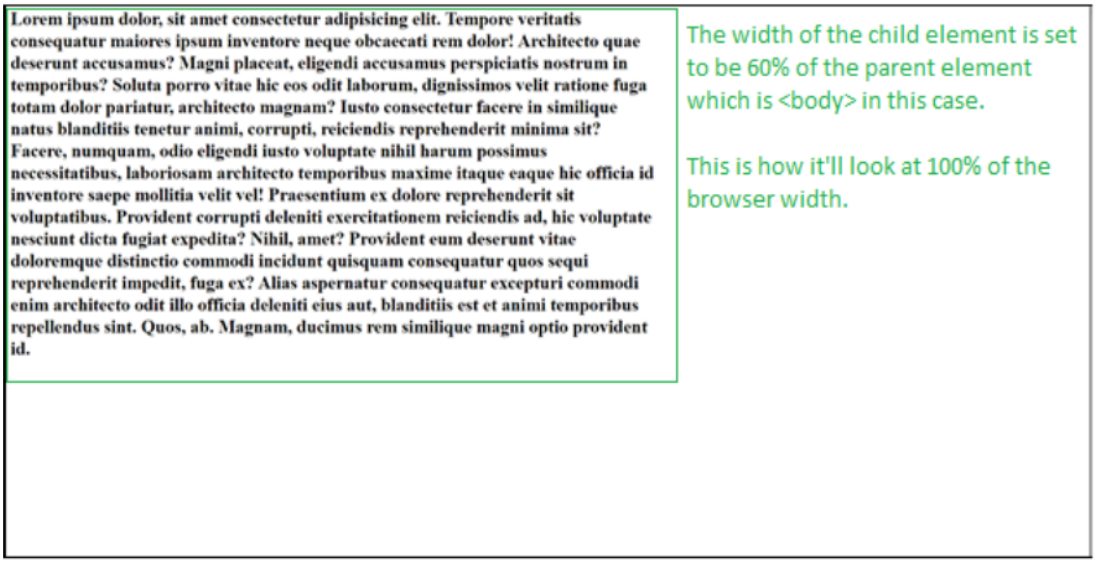
As we all know CSS units are of two types: Absolute and Relative

- Absolute Unit – It's a fixed-size unit across all devices and does not vary with different screen resolutions. For e.g., px(pixel), in(inch), mm(millimeter)
- Relative Unit – It doesn't have a fixed size. It's relative to something else like the font size of a parent element or the size of the viewport. E.g., % (relative

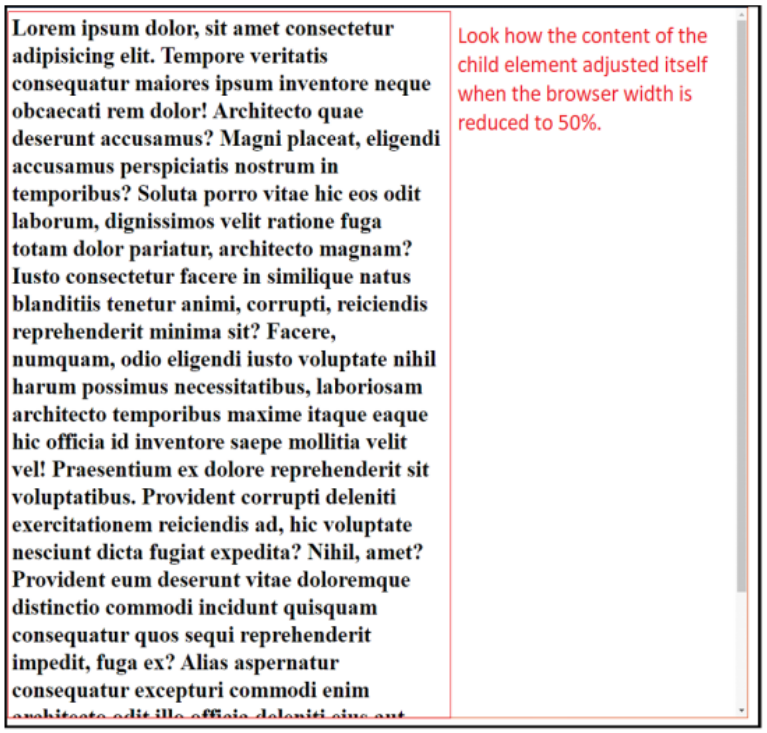
to parent element), rem (relative to the root element), em(relative to the parent element),

We should use relative units in fonts and images while keeping responsiveness in mind. To make things a bit clearer, we have explained the use of the percentage unit below.

The percentage unit is useful when creating a responsive fluid layout. We can adjust the percentage of width an element occupies which will be set according to the width of the parent element. Let's say we have a <p> tag inside the <body>, and we adjust the width of the child element to be 60%, then if we reduce the browser width, it should adjust itself and scale accordingly so that we'll be able to see the entire content of the element.



If we reduce the browser width, the child element will adjust and down-scale itself accordingly.



Viewport meta element

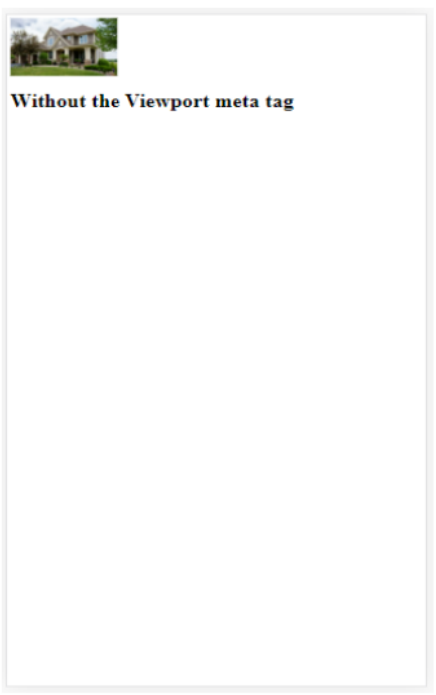
First of all, you'll need to add a viewport meta tag, without which, the mobile devices first render the pages at a typical desktop resolution and then scale them down, which makes it difficult to read as all the text and images will shrink and we'll have to zoom in to see the content.

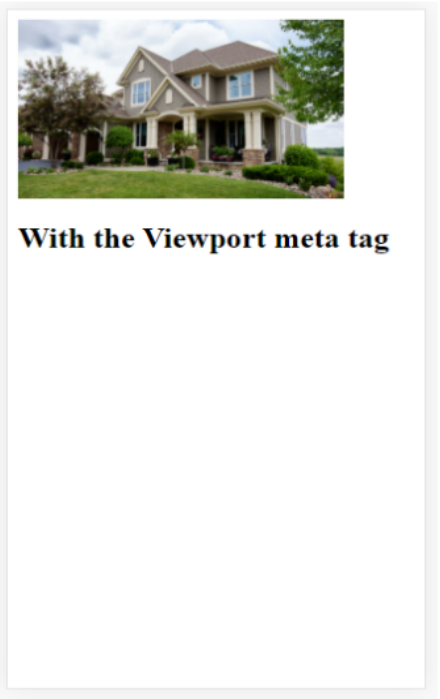
Setting the viewport meta tag, lets us control the width and scaling of the pages so that they are sized according to the device size.

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

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

Here is an example of a web page without the viewport meta tag, and the same web page with the viewport meta tag, when viewed on an iPhone SE:





Responsive Images

Images can be responsive if they scale and adapt nicely to fit any browser window. We can set the width and height in percentages after which, the image will be responsive and it'll scale up and down according to the size of the browser.

The max-width and min-width properties can also be used, which will make sure the image never scales more than and less than its original size respectively. By setting the max-width property to 100%, an image would scale down smaller if its container becomes narrower than the image's intrinsic size, but never grows larger. This enables an image to scale down to fit in a flexibly-sized container, rather than overflow it, and become pixelated if the column becomes wider than the image.

```
<div>
    <img src="image.jpg" alt="" style="max-width: 100%;">
</div>
```

After setting the max-width of <img> tag to 100%, the image scales down if we reduce the browser width, but it doesn't scales up if we increase the browser width

Media Queries

Media Queries were introduced in CSS3 and they are very useful for creating responsive websites, as they give the designer a lot of freedom and flexibility.

Media queries are useful when you want to modify your website or web app depending on the device type and characteristics.
It involves the use of the @media rule to include a block of CSS properties only if a certain condition is true.

A media query consists of two parts: media types and media features.

Media types: They describe the category of devices. Except for using logical operators, the media type is optional, and all types will be set as default.

- all: Default. Used for all media type devices
- print: Used for printers
- screen: Used for laptops, computer screens, tablets, smartphones, etc.
- speech: Used for screen-readers that “reads” the page content out loud.

Media features: Media features are used in media queries, which allow us to apply different styles depending on the capabilities of the output device. They must be surrounded by parentheses.
E.g., height - Height of the viewport.

Syntax:

```
@media not|only mediatype and (mediafeature and|or|not mediafeature) {
  CSS-Code;
}
```

The not, only, and are logical keywords.

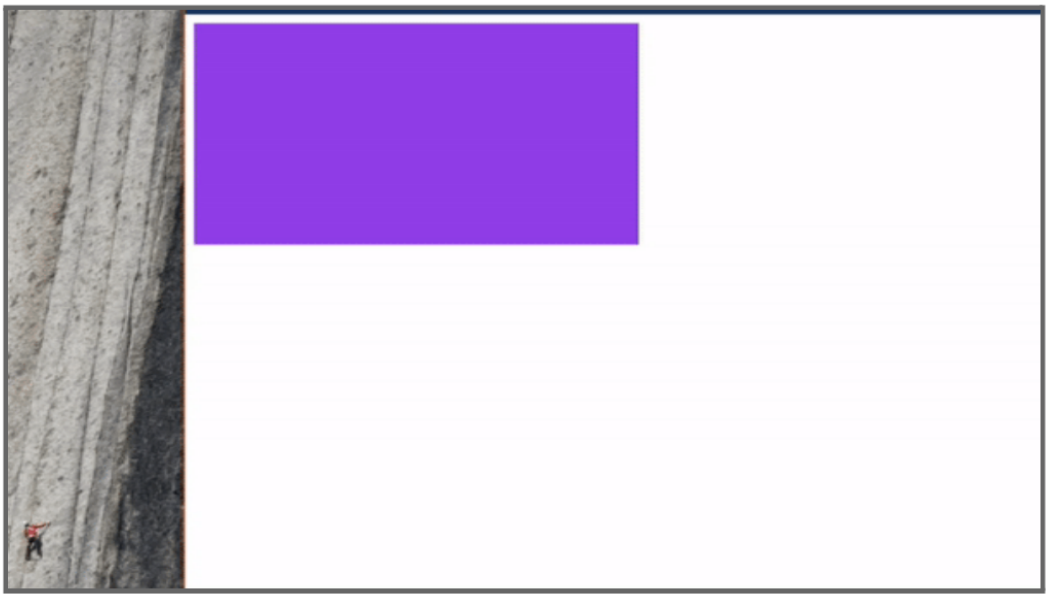
- not: It inverts the meaning of an entire media query.
- only: It will keep older browsers from applying the specified styles if they don't support media queries.
- and: it is used to specify a media feature and media type together

These all are optional and if you use not and only, you'll need to specify a media type.

Examples:

Change an element's color when the browser's width is 500px wide or less:

```
@media screen and (max-width: 500px) {
  #element {
    background-color: royalblue;
  }
}
```



2. Use media queries to set the background-color of an element to **pink** if the viewport is 700 pixels wide or wider, to **darkgreen** if the viewport is between 300 and 699 pixels wide. If the viewport is smaller than 300 pixels, the background-color is **grey:**

```
#element {
    background-color: grey;
    width: 50px;
    height: 50px;
}

@media screen and (min-width: 300px) {
   #element {
      background-color: darkgreen;
   }
}

@media screen and (min-width: 700px) {
   #element {
     background-color: pink;
   }
}
```

Hide an element when the browser's width is 800px wide or less and show it when the browser’s width is 1600px or more:

```
@media screen and (max-width: 800px) {
      #element {
        display: none;
      }
}
@media screen and (min-width: 1600px) {
      #element {
       display: block;
      }
}
```

Responsive Typography

Typography is the technique of arranging text to make written language legible, readable and appealing when displayed. There are different ways to achieve this. Generally, you can use pixels when the website you're working on, has a fixed width.

All the text and font in a responsive website are also responsive. ie It should change according to the size of the device browser. The font of your website should have a size that is related to its parent's container width so that it can adapt to the screen of the client and be easily readable on mobile devices.

CSS3 includes different relative units which can help us achieve relative typography.

- You can use 'vw' to set the text size. It means the 'viewport width' and its 1 unit's value is 1% of the total viewport width. As you'll use this, the size of the text gets itself adjusted according to the size of the browser window as the viewport is the browser window size.

```
<h2 style="font-size: 5vw;">
  Hello World!
</h2>
```

- We can also change font size with media queries. We can use rem. They are relative to the HTML element, which makes them a lot easier to use. We can define responsive font sizes as shown below:

```
@media (min-width: 640px) { body {font-size:1rem;} }
@media (min-width:960px) { body {font-size:1.2rem;} }
@media (min-width:1100px) { body {font-size:1.5rem;} }
```

Mobile-first Vs Desktop first

Desktop-first Responsive Strategy - This strategy involves designing a display that can show the maximum amount of information so that you can communicate with your audience as much as you want, provided that the target audience uses big screen devices like laptops and desktop PCs.

Mobile-first Responsive Strategy - As smaller screens have less space, you'll need to hide and adjust some elements while working with them. Generally, there are no changes in the functionality of the website. We'll have to focus on the user experience along with the proper implementation of the website on mobile devices. The mobile-first strategy needs a lot of research as compared to the Desktop-first strategy so that we can organize the content according to priority and space.

Most web content today is consumed with mobile devices so it just makes sense to develop mobile-first. Whether your app is mobile accessible or not, it should depend on your target audience.

Responsive design is the standard practice. Once upon a time, people called it mobile-first. It's easy to make your website responsive nowadays with CSS frameworks like bootstrap. The advantage of responsive design is that you don't have to try to add messy code to retrofit your website if you decide to make it mobile later on which makes it simpler to grow and maintain. Therefore, it all depends on the priority you'll give to the target audience. You can also use certain frameworks that can help you make responsive websites with much ease.

Responsive Frameworks

There are many free front-end frameworks that offer responsive design like Bootstrap and Foundation.

You can read about them in detail from the links given below:

1. https://getbootstrap.com/
2. https://get.foundation/