Lesson:

Introduction to Media Queries & RWD

PW SKILLS

Topics Covered

- What is Responsive web design(RWD)?
- Advantages of RWD
- Ways to implement Responsive web design
- What is the media query in CSS?
- Anatomy of a Media Query

What is Responsive Web Design(RWD)?

Responsive Web Design (RWD) is an approach to web design and development that aims to create websites that provide an optimal viewing and user experience across a wide range of devices and screen sizes. For example smartphones, tablets, and various other devices, it is important for websites to be adaptable and accessible on different platforms.

The primary goal of responsive web design is to ensure that a website's layout and content can automatically adjust and adapt to fit the screen size and orientation of the device being used. This means that regardless of whether someone is viewing a website on a desktop computer, a smartphone, or a tablet, the website should be visually appealing, easy to navigate, and provide a seamless user experience.

Advantages of RWD

1. Improved User Experience: RWD ensures that your website looks and functions well across different devices, providing a seamless and user-friendly experience for visitors.

2. Increased Mobile Traffic: With the rising use of mobile devices, RWD allows your website to cater to a larger audience, as it adapts to various screen sizes and resolutions.

3. Cost and Time Efficiency: Instead of creating separate websites or apps for different devices, RWD allows for a single codebase, reducing development time and costs associated with maintaining multiple versions.

4. Search Engine Optimization (SEO) Benefits: Having a responsive website is favored by search engines like Google, as it avoids duplicate content and provides a consistent URL structure. This can contribute to better search rankings and increased visibility.

Ways to implement Responsive Web Design

We have several ways to implement RWD, we will discuss a few common techniques to make your website Responsive.

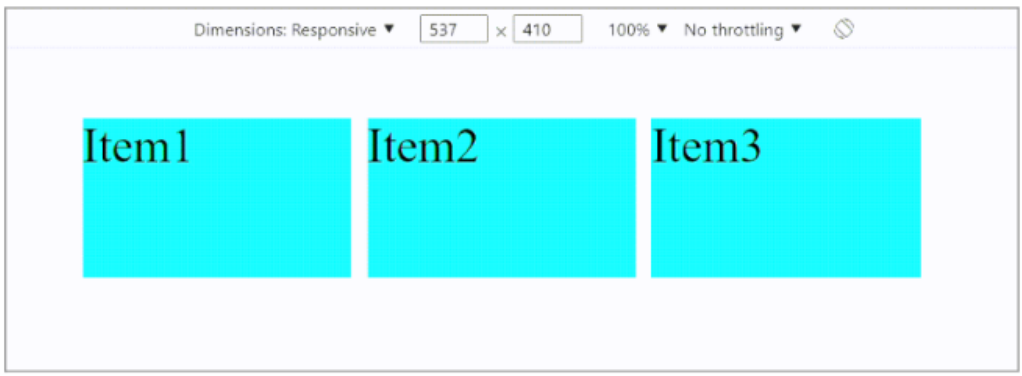
1. Responsive Layouts
By default, several layout methods such as Multiple-column layout, Flexbox, and Grid possess inherent responsiveness.

Although we have already studied flex and grid, let's take an example to recall it with respect to responsive design.

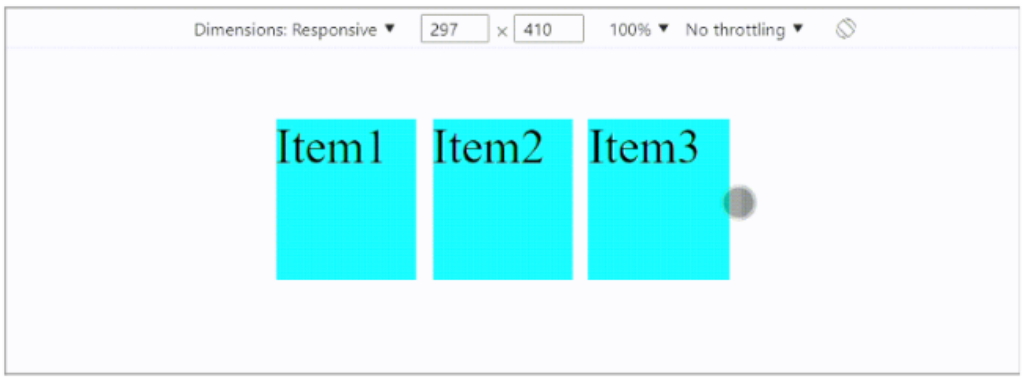
Example:- Check below responsive design below using flex layout.

```
<style>
  .container {
    display: flex;
    justify-content: space-around;
    gap: 10px;
  }

  .container div {
    width: 100%;
    height: 100px;
    background-color: aqua;
    font-size: 30px;
  }
</style>
<body>
  <div class="container">
    <div>Item1</div>
    <div>Item2</div>
    <div>Item3</div>
  </div>
</body>
```

Large Screen



Small Screen



In the above example, we have 3 three flex items, and if you observe they are by default responsive because of display:flex

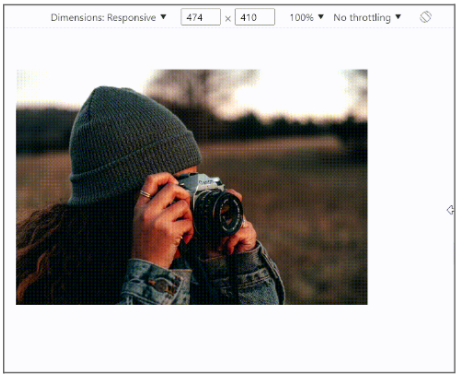
2. Responsive images

To make the image size responsive, we can use max-width:100%, so that image never overflows its container and adjusts itself as the container size increases or decreases.

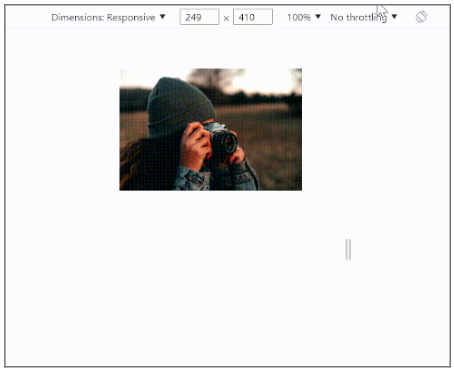
Example:

```
<style>
  .container{
    width: 80vw;
  }

  img {
    max-width: 100%;
  }
</style>

<body>
   <div class="container">
      <img src="./assets/photo.jpg" alt="responsive image"></div>
   </div>
</body>
```

Large Screen



Small Screen



To analyze the output please open inspect mode**(CTRL + SHIFT + I).**

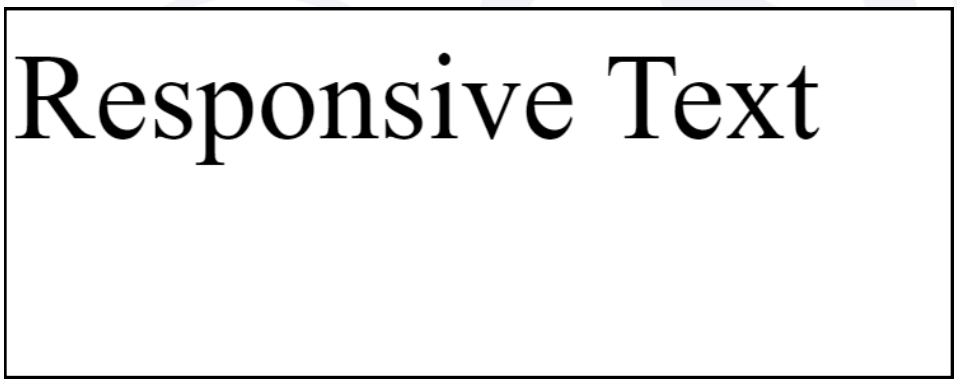
3. Media Query
It is the most prominent method for creating mobile responsive web designs. Media queries enable us to perform a set of evaluations, such as determining if the user's screen exceeds a specific width and selectively applying CSS to appropriately style the webpage based on the user's requirements.

4. Responsive Typography
It refers to the technique of adjusting font sizes based on either media queries or relative units like rem which sets font size relative to base font size.

Example

```
<style>
   html {
      font-size: 10px;
    }
    div {
      font-size: 4rem;
    }
</style>
<body>
    <div>Responsive Text</div>
</body>
```

Here, we have set the font size of the text as **4rem** (4*10px root = 40px).



5. The Viewport meta tag
In the HTML source code of a responsive webpage, it is common to observe the presence of the following <meta> tag within the <head> section of the document.

```
<meta name="viewport" content="width=device-width,initial-scale=1" />
```

Let's break down the attributes and their meanings:

- **name="viewport":** This attribute specifies that the <meta> tag is defining the viewport settings for the web page.
- **content="width=device-width, initial-scale=1":** This attribute defines the content, or the actual instructions, for the viewport settings. It consists of two parts:

- **width=device-width:** This instructs the browser to set the width of the viewport to the width of the device's screen. It ensures that the webpage's layout adapts to the screen size of the device being used, allowing content to fit properly.

- **initial-scale=1:** This sets the initial zoom level of the webpage to 100%. It ensures that the webpage is displayed at a 1:1 scale, without any initial zooming or scaling applied.

What is the media query in CSS?

Media queries in CSS allow you to define different styles for different device types, screen sizes, and orientations. This is particularly useful for creating responsive web designs that adapt to different devices and screen sizes.

Media queries are written using the @media rule, followed by one or more conditions in parentheses. The conditions can be based on various criteria, including the device width, device height, orientation, and pixel density.

Anatomy of a Media Query

Media

The @media rule is the starting point for defining media queries in CSS. It is used to define a block of styles that should only be applied under certain conditions (i.e. based on media query conditions).

```
@media [media-type] ([media-feature]) {
  /* Styles! */
}
```

Media Type

The media type is a keyword that specifies the type of device or media being targeted. There are several media types that can be used in a media query, including

- **all:** The media type targets all devices and media types.
- **screen:** This media type targets devices with a screen, such as desktops, laptops, tablets, and smartphones.
- **print:** This media type targets device that are used for printing, such as printers and PDF generators.
- **speech**: This media type targets speech-based devices, such as screen readers.

Here is an example of a media query that targets screens with a maximum width of 600 pixels:

```
@media screen and (max-width: 600px) {
  /* Styles for screens with a maximum width of 600px */
}
```

Media feature
A media feature is used to test specific characteristics of the device or viewport, such as width, height, orientation, and resolution.

Here are some common media features that can be used in a media query:
width: specifies the width of the viewport.
height: specifies the height of the viewport.
Orientation: Specifies the orientation of the device.
Aspect-ratio: specifies the aspect ratio of the viewport.
Resolution: specifies the resolution of the device.
Color: Specifies the number of bits per color channel.
hover: specifies whether the device has a pointing device or not.
pointer: specifies the type of pointing device available.

Operators
Logical operators such as and, or, and not can be used to combine multiple media features or media types to create more complex conditions for when styles should be applied.

Here's a brief overview of the logical operators:

"and": This operator combines two or more media features or media types and requires that all conditions be true for the styles to apply.

```
Matches screen between 320px AND 768px

@media screen (min-width: 320px) and (max-width: 768px) {
  .element {
    /* Styles! */
  }
}
```

"or": This operator combines two or more media features or media types and requires that at least one condition must be true for the styles to apply.

```
/*  matches screens where either the user prefers dark mode or the
screen is at least 1200px wide */

@media screen (prefers-color-scheme: dark), (min-width 1200px) {
  .element {
    /* Styles! */
  }
}
```

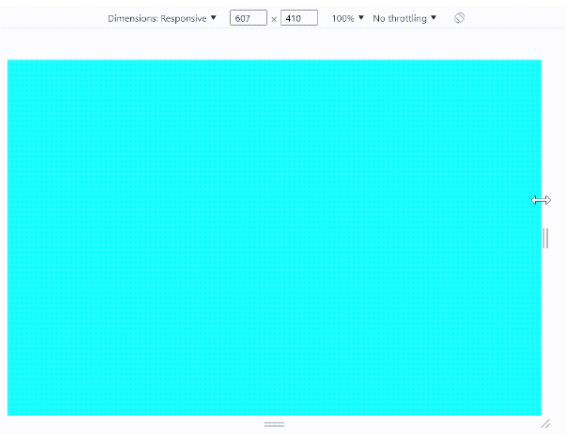
"not": This operator negates the condition that follows it, and requires that the condition be false for the styles to apply

```
@media print and ( not(color) ) {
  body {
    background-color: none;
  }
}
```

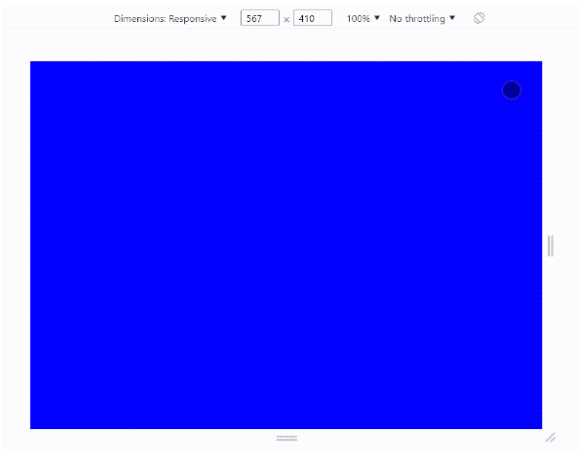
Example

```
<style>
  body {
     background-color: aqua;
  }

/* Apply below CSS when screen width reaches ⩾ 600px */
  @media (max-width: 600px) {
    body {
      background-color: blue;
    }
  }
</style>
<body>
    <div class="container"></div>
</body>
```

Large Screen



Small Screen



In the above example, if you observe background colour changes, when it reaches a screen width of more than or equal to **600px**. To analyze the output please open inspect mode**(CTRL + SHIFT + I).**