AIMERZ.ai
Aim.Act.Achieve

NOTES

Media Query

What is Responsive Web Design ?

Responsive Web Design (RWD) is a way of designing websites so they look good and work well on all kinds of devices, like smartphones, tablets, and computers.

The main goal is to make sure that a website automatically changes its layout and content to fit the screen size and orientation of the device being used. This means whether you're on a desktop, a phone, or a tablet, the website should be easy to use, visually appealing, and provide a smooth experience.

Advantages of Responsive Web Design

1. **Better User Experience**: RWD makes sure your website looks good and works well on all devices, giving visitors a smooth and easy experience.
2. **More Mobile Visitors**: With more people using phones and tablets, RWD helps your website reach a larger audience by fitting different screen sizes.
3. **Saves Time and Money**: Instead of building separate websites or apps for each device, RWD uses one codebase. This means less time and money spent on development and maintenance.
4. **SEO Benefits**: Search engines like Google prefer responsive websites because they avoid duplicate content and keep a consistent URL. This can help improve your search rankings and make your site more visible.

Ways to Implement Responsive Web Design

There are several ways to make your website responsive. Here are a few common techniques:

1. **Responsive Layouts**
   Some layout methods, like multiple-column layouts, Flexbox, and Grid, are naturally responsive. Let's briefly revisit Flexbox

and Grid as examples of how they can be used in responsive design.

- **Flexbox**: This layout method allows you to arrange items in a flexible way. It can adjust the size and position of elements based on the available space, making it easy to create responsive designs.
- **Grid**: The Grid layout provides a more structured way to arrange content. You can create rows and columns that adapt to different screen sizes, allowing for a clean and organized look across devices.

Example:

index.html

```
JavaScript
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <div class="container">
      <div>BOX1</div>
      <div>BOX2</div>
      <div>BOX3</div>
    </div>
  </body>
</html>
```

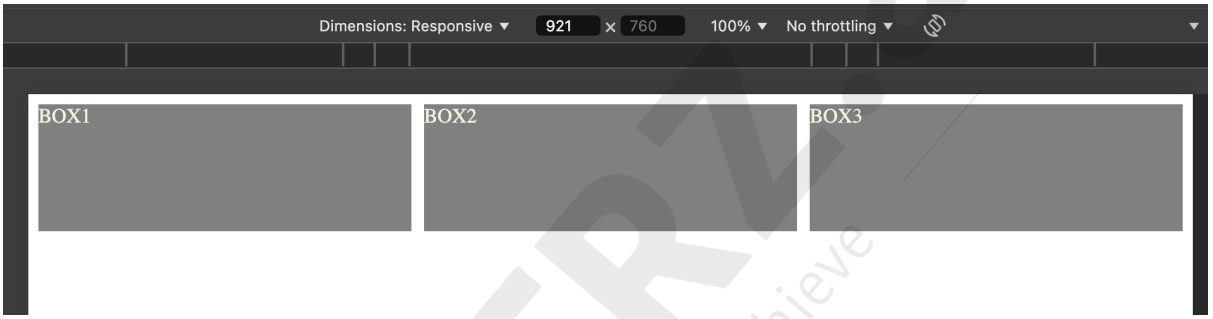
style.css

```
JavaScript
.container {
  display: flex;
  justify-content: space-between;
  gap: 10px;
}
```

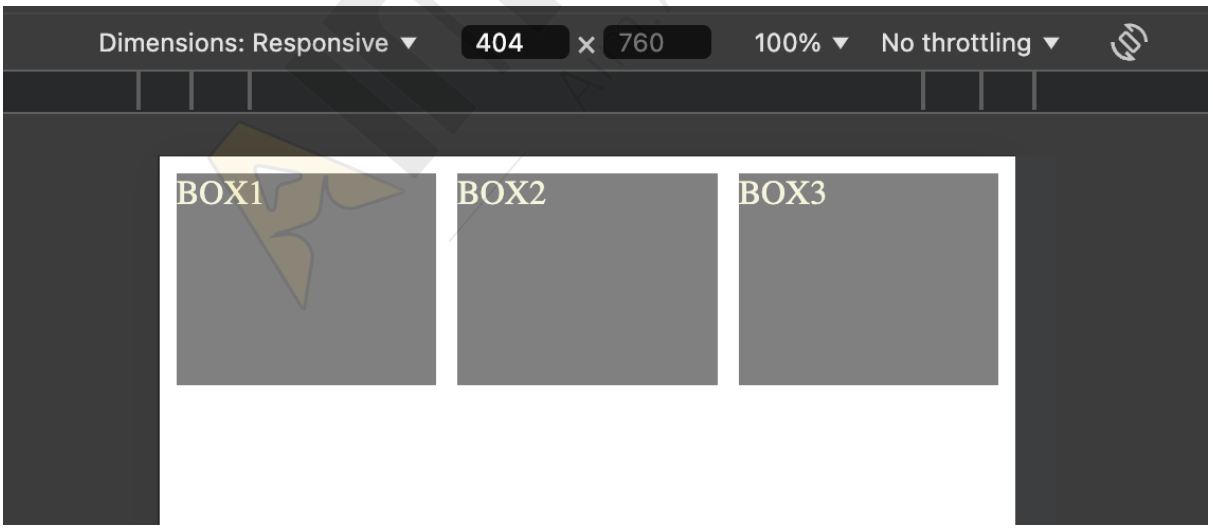
```
.container div {
  width: 100%;
  height: 100px;
  background-color: grey;
  color: beige;
}
```

Output:

Large Screen:



Small Screen:



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

2. Responsive images:

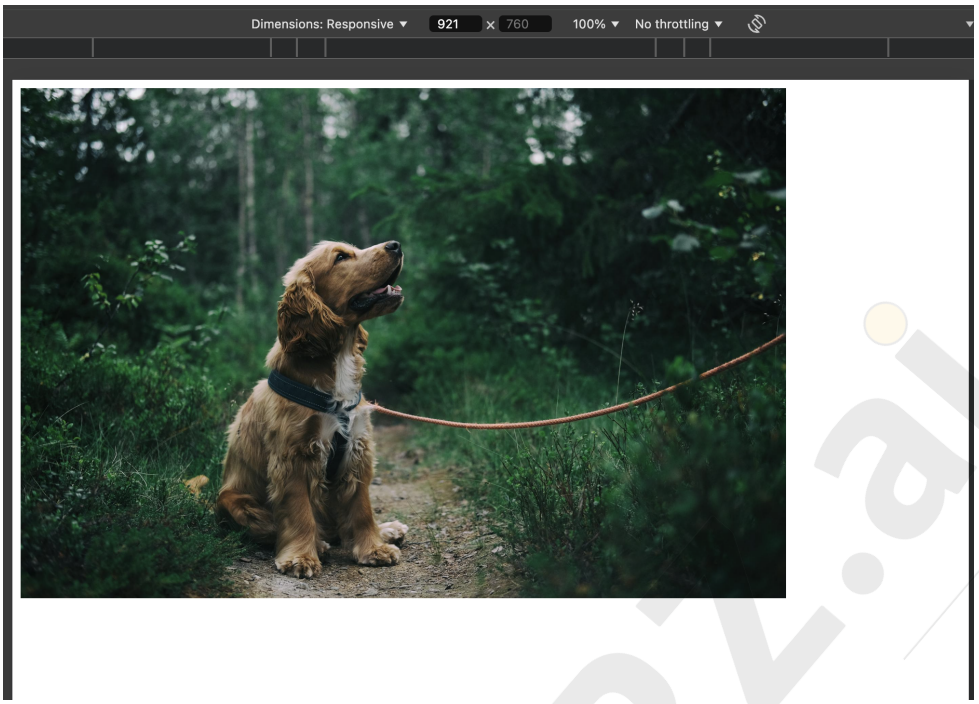
To make the image size responsive, we can use **max-width:100%**, so that image never overflows its container and adjusts itself as the container size increases or decreases.

Example:

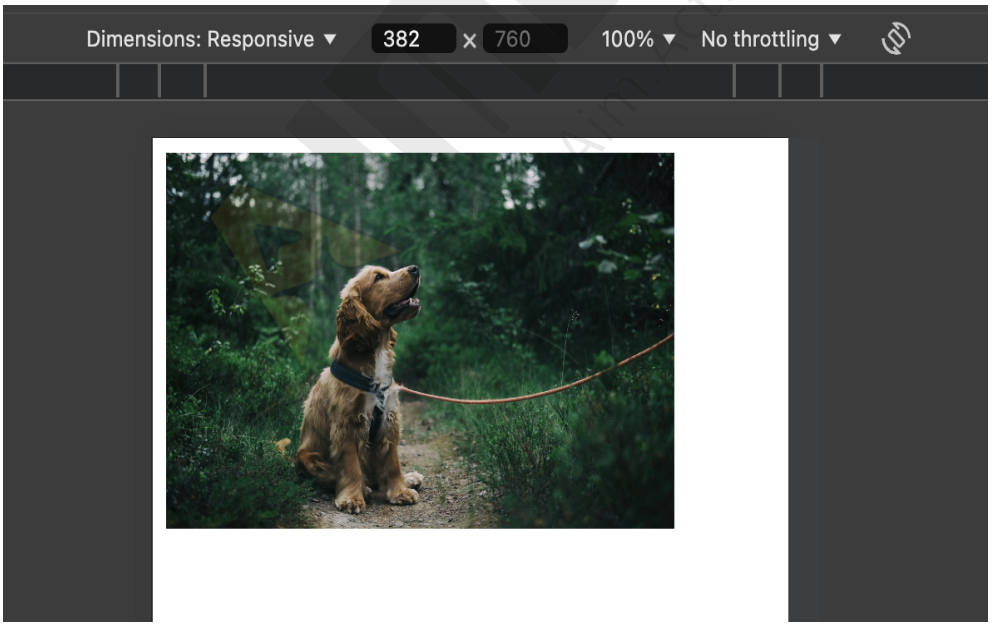
```JavaScript
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <style>
      .container {
        width: 80vw;
      }

      img {
        max-width: 100%;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <img

src="https://images.pexels.com/photos/1254140/pexels-photo-1254140.jpeg?auto
=compress&cs=tinysrgb&w=1260&h=750&dpr=2"
        alt="responsive image"
      />
    </div>
  </body>
</html>
```

Output:

Large Screen



Small Screen:



3. Media query:

The most common way to create mobile-friendly web designs is by using media queries. Media queries help us check things like the width of the user's screen. Based on this information, we can apply different CSS styles to make the webpage look good on all devices.

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:
index.html

```
JavaScript
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="styles.css" />
    <title>Responsive Typography Example</title>
  </head>
  <body>
    <h1 class="heading">Responsive Typography</h1>
    <p class="paragraph">This text will adjust based on the base font
size.</p>
  </body>
</html>
```

Style.css

```
JavaScript
/* Base font size */
html {
  font-size: 16px; /* 1rem = 16px */
}

.heading {
  font-size: 2rem; /* 32px */
}
```

```
.paragraph {
  font-size: 1rem; /* 16px */
}
```

Here, we have set the font size of the text as 2rem (2*16px root = 32px).

Output:

5. The Viewport meta tag

In the HTML source code of a responsive webpage, it's common to see the following <meta> tag within the <head> section:

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

Let's break down the attributes and their meanings:

name="viewport" : This attribute indicates that the <meta> tag is defining the viewport settings for the webpage. It tells the browser that the content that follows will specify how to control the dimensions and scaling of the viewport.

content="width=device-width, initial-scale=1": This attribute provides specific instructions for how the viewport should behave. It consists of two main parts:

width=device-width :This instructs the browser to set the width of the viewport to match the width of the device's screen. It ensures that the layout of the webpage adapts to the screen size, allowing content to fit properly without requiring horizontal scrolling. This is essential for creating a responsive design that works seamlessly on various devices, from smartphones to tablets and desktops.

initial-scale=1: This sets the initial zoom level of the webpage to 100%. It ensures that the webpage is displayed at its natural size (1:1 scale) when first loaded, without any automatic zooming or scaling applied. This helps maintain the intended design and improves user experience by preventing content from appearing too small or too large.

Media Query in CSS

Media queries in CSS let you create different styles for different devices, screen sizes, and orientations. This helps in making websites look good on all devices, from phones to large screens.

You use the **@media** rule to define these styles, followed by conditions in parentheses. These conditions can check things like the width or height of the device, its orientation (portrait or landscape), or the screen's pixel density. It's a key tool for building responsive web designs that adapt to different screen sizes and devices.

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).

```
Unset
@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:

Syntax:

```
JavaScript
@media screen and (max-width: 600px) {
  /* Styles for screens 600px wide or smaller */
}
```

index.html

```
JavaScript
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Responsive Navigation Menu</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <nav>
            <ul class="nav-menu">
                <li><a href="#">Home</a></li>
                <li><a href="#">About</a></li>
                <li><a href="#">Services</a></li>
                <li><a href="#">Contact</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <h1>Welcome to Our Website</h1>
        <p>This is a simple example of a responsive navigation menu using
media queries.</p>
    </main>
</body>
</html>
```

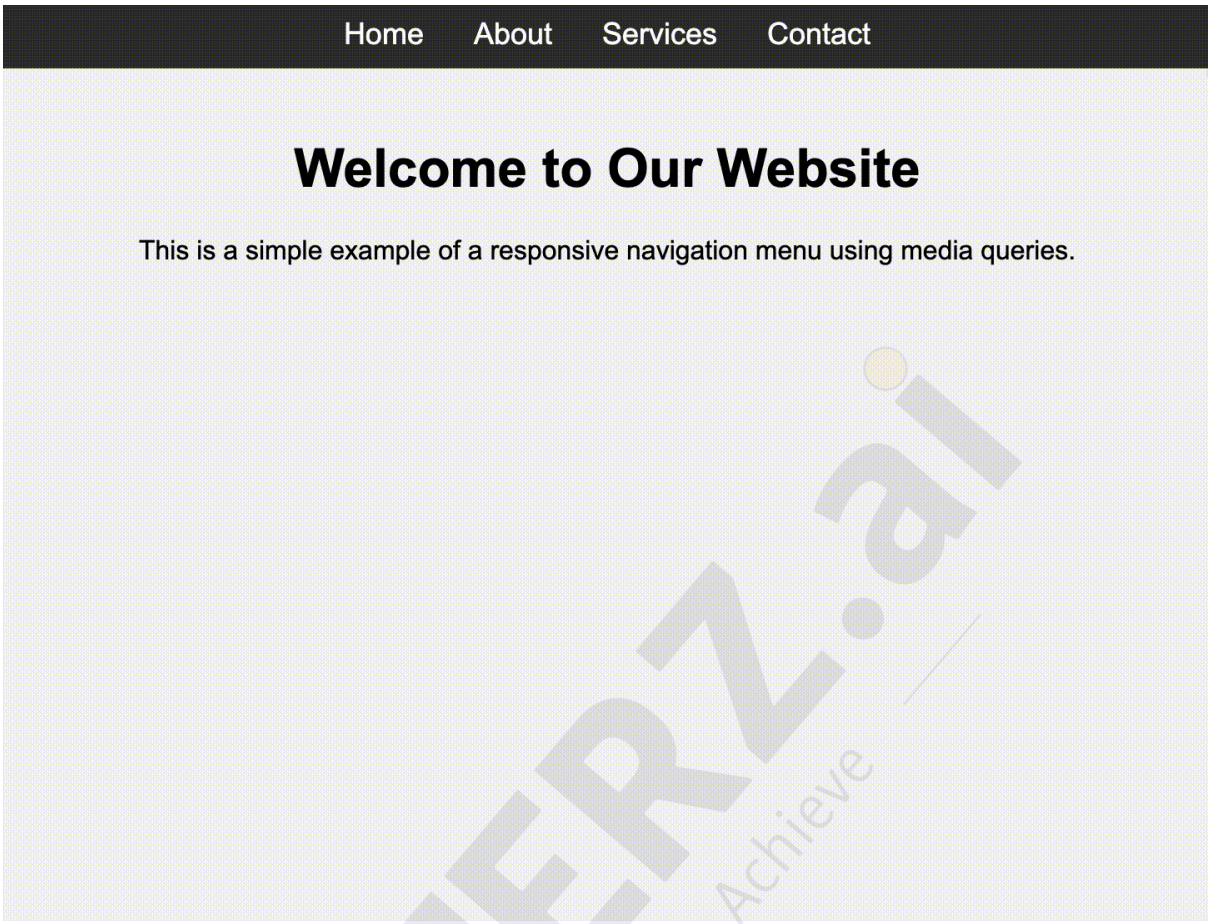
style.css

JavaScript

```
/* Base styles for larger screens */
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f4f4f4;
}

header {
  background-color: #333;
  padding: 10px 0;
}

.nav-menu {
  list-style-type: none;
  margin: 0;
  padding: 0;
```

```
  display: flex;
  justify-content: center;
}

.nav-menu li {
  margin: 0 15px;
}

.nav-menu li a {
  color: white;
  text-decoration: none;
  font-size: 18px;
}

main {
  text-align: center;
  padding: 20px;
}

/* Media query for screens 600px wide or smaller */
@media screen and (max-width: 600px) {
  .nav-menu {
    flex-direction: column;
    align-items: center;
  }

  .nav-menu li {
    margin: 10px 0;
  }

  .nav-menu li a {
    font-size: 20px;
  }
}
```

Output:



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.

THANK YOU

AIMERZ.ai
Aim.Act.Achieve