Assignment 02 Question & Answers:

Qn 1. What is CSS Media Query? Show the use of maxwidth & minwidth properties.

Answer: CSS Media Queries are a feature of CSS3 that allow developers to apply specific styles based on the characteristics of the device or browser, such as screen size, resolution, or orientation. They enable responsive web design by adapting layouts and styles to different devices, ensuring an optimal user experience across various screen sizes.

`max-width` **and** `min-width` **Properties**

- `max-width`: Applies styles when the viewport width is less than or equal to the specified value. It is commonly used for targeting smaller screens like mobile devices.
- `min-width`: Applies styles when the viewport width is greater than or equal to the specified value. It is often used for larger screens like tablets or desktops.

Example Code:

Here is a practical example demonstrating the use of max-width and min-width:

```
/* Default styles */
body {
  background-color: white;
  font-size: 16px;
}

/* Styles for screens smaller than or equal to 600px */
@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
    font-size: 14px;
  }
}

/* Styles for screens larger than or equal to 768px */
@media only screen and (min-width: 768px) {
  body {
    background-color: lightgreen;
    font-size: 18px;
  }
}

/* Styles for screens between 600px and 768px */
@media only screen and (min-width: 600px) and (max-width: 768px) {
  body {
    background-color: lightcoral;
    font-size: 16px;
  }
}
```

Explanation:

- The default styles apply to all devices unless overridden by a media query.
- The `max-width` query targets devices with a width of 600 px or less, applying a light blue background and smaller font size.
- The min-width query targets devices with a width of 768 px or more, applying a light green background and larger font size.
- A combined query with both min-width and max-width defines styles specifically for devices between 600 px and 768 px, applying a light coral background.

Qn 2. List the properties associated with 2D effects and transformation. Show examples.

Answer:

Properties Associated with 2D Effects and Transformations

The CSS `transform` property allows you to apply various 2D transformations to elements, such as translating, rotating, scaling, skewing, or combining these effects. Below are the key 2D transformation properties and their descriptions:

1. `translate()`
    - Moves an element from its current position.
    - Can be applied along the X-axis, Y-axis, or both using `translateX()`, `translateY()`, or `translate(x, y)`.

      Example:

      ```
      div {
          transform: translate(50px, 100px); /* Moves element 50px right and
      100px down */
      }
      ```

2. `rotate()`
    - Rotates an element around its origin by a specified angle *(in degrees)*.
    - Positive values rotate clockwise; negative values rotate counter-clockwise.

      Example:

      ```
      div {
          transform: rotate(45deg); /* Rotates element by 45 degrees */
      }
      ```

3. `scale()`
    - Resizes an element by scaling it along the X and Y axes.
    - Use `scaleX()` or `scaleY()` for individual axis scaling.

      Example:

```
div {
    transform: scale(1.5); /* Increases size by 1.5 times */
}
```

4. `skew()`
    - Skews an element along the X-axis, Y-axis, or both using angles.
    - Use `skewX()` or `skewY()` for individual axis skewing.

      Example:

```
div {
    transform: skew(30deg, 10deg); /* Skews 30 degrees on X-axis and 10 degrees on Y-axis */
}
```

5. `matrix()`
    - Combines all transformations (translate, rotate, scale, skew) into one function.
    - Takes six parameters: `matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())`.

      Example:

```
div {
    transform: matrix(1, -0.3, 0.3, 1, 50, 100); /* Combines transformations */
}
```

Examples of Multiple Transformations

You can combine multiple transformations in a single rule:

```
div {
  transform: translate(50px, 100px) rotate(45deg) scale(1.5);
}
```

This moves the element by `(50px, 100px)`, rotates it by `45°`, and scales it by `1.5x`.

Qn 3. What is CSS-3 animation? Demonstrate it by dragging a box in any one direction.

Answer:

What is CSS3 Animation?: CSS3 animations allow you to animate HTML elements without using JavaScript or Flash. They are defined using the `@keyframes` rule, which specifies the changes in CSS properties at different points in time (keyframes). These animations can be controlled with properties like `animation-name`, `animation-duration`, `animation-iteration-count`, and more.

Animations are more flexible than transitions because they allow for multi-step animations and greater control over timing.

Example: Dragging a box in 1 direction

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Animation Example</title>
  <style>
    /* Keyframes for moving the box */
    @keyframes moveRight {
      from {
        transform: translateX(0); /* Start at original position */
      }
      to {
        transform: translateX(300px); /* Move 300px to the right */
      }
    }

    /* Styling the box */
    .box {
      width: 100px;
      height: 100px;
      background-color: lightblue;
      position: relative; /* Required for transform to work */
      animation-name: moveRight; /* Link to keyframes */
      animation-duration: 2s; /* Animation lasts for 2 seconds */
      animation-timing-function: ease-in-out; /* Smooth movement */
      animation-fill-mode: forwards; /* Keeps the box at the end position */
    }
  </style>
</head>
<body>
  <div class="box"></div>
</body>
</html>
```

Qn 4. Show the use of active, focus and hover effects.

Answer:

CSS Pseudo-Classes: Active, Focus, and Hover

CSS provides pseudo-classes like `:hover`, `:focus`, and `:active` to style elements based on user interactions. Here's a brief explanation of each:

1. `:hover`: Applies styles when the user hovers over an element with a mouse.
2. `:focus`: Applies styles to an element that has received focus, typically through keyboard navigation or mouse click (e.g., form inputs).
3. `:active`: Applies styles to an element when it is being clicked or activated.

Example Code

Below is an example demonstrating the use of `:hover`, `:focus`, and `:active` effects on a button:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Pseudo-Classes</title>
  <style>
    /* Default button styles */
    button {
      background-color: lightblue;
      color: black;
      border: 2px solid blue;
      padding: 10px 20px;
      font-size: 16px;
      cursor: pointer;
      transition: all 0.3s ease; /* Smooth transitions for hover/focus/active */
    }

    /* Hover effect */
    button:hover {
      background-color: blue;
      color: white;
    }

    /* Focus effect */
    button:focus {
      outline: none; /* Removes default browser outline */
      border-color: green;
      box-shadow: 0 0 5px green; /* Adds a glowing effect */
    }

    /* Active effect */
    button:active {
```

```
      background-color: darkblue;
      transform: scale(0.95); /* Slightly shrinks the button when clicked */
    }
  </style>
</head>
<body>
  <button>Interact with Me!</button>
</body>
</html>
```

Qn 5. Using CSS, draw the following objects:

- Triangle
- Circle
- Oval
- Triangle upon rectangle
- Two circles joint together *(X-axis)*

Answer:

- Triangle:

```
<div class="triangle"></div>

<style>
.triangle {
  width: 0;
  height: 0;
  border-left: 50px solid transparent;
  border-right: 50px solid transparent;
  border-bottom: 100px solid red; /* Triangle color */
}
</style>
```

- Circle

```
<div class="circle"></div>

<style>
.circle {
  width: 100px;
  height: 100px;
  background-color: blue; /* Circle color */
  border-radius: 50%;
}
</style>
```

- Oval

```
<div class="oval"></div>

<style>
.oval {
  width: 150px;
  height: 100px;
  background-color: green; /* Oval color */
  border-radius: 50%;
}
</style>
```

- Triangle Upon Rectangle

```
<div class="triangle-up"></div>
<div class="rectangle"></div>

<style>
.rectangle {
  width: 100px;
  height: 50px;
  background-color: orange; /* Rectangle color */
}

.triangle-up {
  width: 0;
  height: 0;
  border-left: 50px solid transparent;
  border-right: 50px solid transparent;
  border-bottom: 50px solid red; /* Triangle color */
  margin-bottom: -1px; /* Adjusts the position to overlap with the rectangle */
}
</style>
```

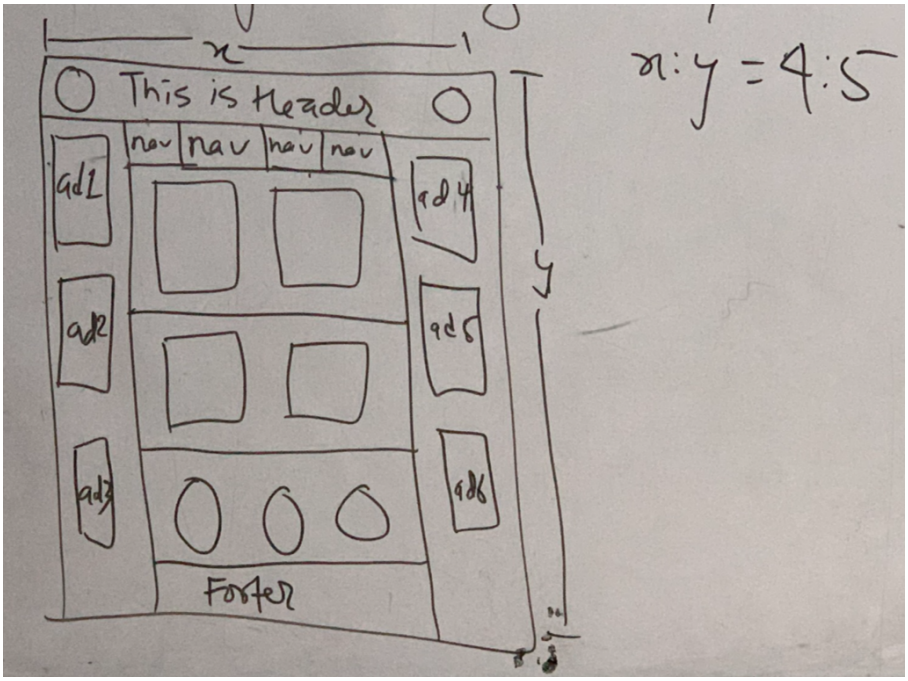
- Two Circles Joint Together *(X-Axis)*

```
<div class="circle1"></div>
<div class="circle2"></div>

<style>
.circle1, .circle2 {
  width: 50px;
  height: 50px;
  background-color: purple; /* Circle color */
  border-radius: 50%;
  display: inline-block;
  margin-right: -4px; /* Adjusts the margin to make circles touch */
```

```
}
</style>
```

Qn. 6 Design the following looking template:



Answer:

> *Click Here to redirect towards the HTML file.*

> *Click Here to redirect towards the CSS file.*

End of assignment 02