**HTML:**

TAGS-

<sub> - sub part of a word H2O3 (2 below h so we use sub)

<sup> -superior part of a word H2O3 (3 above O so we use sup)

LISTS -

Orderlist -- [1,2,3] <ol> <li> </li> </ol>

unorderedlist -- [. . .] <ul> <li> </li> </ul>

TABLE TAG -

<table>

<tr>

<th> </th> #bold data

<td> </td>

</tr>

</table>

colspan - divides a columns horizontally

rowspan - divides a row vertically

table>tr\*3>td\*6 - Creates a table with 3 rows and 6 columns

-----------------------------FORMS--------------------------------------

<!-- Form starts here -->

<form>

<!-- Input field for name -->

<label>Name</label> <input type="text" placeholder="Enter your name"/><br>

<!-- Input field for age --> [only allows numbers in input ]

<label>Age</label> <input type="number" placeholder="Enter your age"/><br>

<!-- Input field for password --> [Blocks the password in dotted manner]

<label>Password</label> <input type="password"/><br>

<!-- Radio buttons for gender --> [select only one option from the data]

<label>Gender:</label>

<label>Male</label>

<input type = "radio" name="Gender"/>

<label>Female</label>

<input type = "radio" name="Gender"/><br>

<!-- Dropdown list for department --> [options for data in a scroll down manner]

<label>Department</label>

<select>

<option>CSE</option>

<option>AIML</option>

<option>DS</option>

<option>IT</option>

</select><br>

<!-- Checkboxes for skills --> [Helps to select more than one option]

<label>Skills:</label>

<label>Data Structures</label>

<input type = "checkbox" name = "Skills"/>

<label>FSD</label>

<input type = "checkbox" name = "Skills"/>

<label>Animation Designing</label>

<input type = "checkbox" name = "Skills"/><br>

<!-- Range input field -->

<label>Range</label>

<input type="range"> <br>

<!-- Textarea field --> [Give a huge text box]

<label>Textarea:</label>

<Textarea></Textarea>

<!-- Reset and submit buttons --> [Reset the complete form]

<input type="reset" />

<input type="submit"/>

</form>

ANCHOR TAG = For attaching links <a></a>

CSS:

Inline Styling:

Internal Stying:

External Styling:

Inline Styling:

Styling is given in the line itself for content

<body>

� � <p style = "font:900; color: bisque; background-color: black;" > Inline Styling</p>

</body>

Internal Styling:

<head>

� � <title>INTERNAL STYLING</title>

� � <style>

� � � � p{

� � � � � � color: blue;

� � � � � � font-family:'Times New Roman', Times, serif;

� � � � � � background-color: burlywood;

� � � � }

� � � � #mytag{

� � � � � � color: red;

� � � � � � font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif;

� � � � � � background-color: wheat;

� � � � }

� � </style>

</head>

<body>

� � <p id = "mytag">MS DHONI</p>

� � <p id = "mytag">KOHLI</p>

� � <p>SURESH RAINA</p>

� � <p>ROHIT SHARMA</p>

</body>

External Styling:

We use a separate css file for Styling

HTML FILE:

<head>

� � <title>External Styling</title>

� � <link rel="stylesheet" href="CSS-External.css">

</head>

<body>

� � <p>Hey Buddy!Light Weight</p>

� � <p id ="tag1" class="tag2">QuadStomp</p>

� � <p class = "tag2">ARNOLD SCHWEZNEGGAR</p>

� � <p id = "tag1"class ="tag3 tag2">PHIL HEATH</p>

</body>

CSS FILE:

p{

� � color: rgb(228, 167, 88);

� � font: oblique;

}

#tag1{

� � background-color: rgb(0, 0, 0);

}

.tag2{

� � background-color: rgb(8, 133, 249) ;

}

.tag3{

� � background-color: chartreuse !important;

}

PREFERENCE:

!important >inline>id>class

**DIVISIONS:**

**DIVISION BORDER:**

1. **border-top-width**, **border-bottom-width**, **border-left-width**, **border-right-width**: These properties set the width of the top, bottom, left, and right borders, respectively. In this example, the top border is 10px wide, and the other borders are 20px wide.
2. **border-top-style, border-bottom-style, border-left-style, border-right-style:** These properties set the style of the top, bottom, left, and right borders, respectively. In this example, the top border is a groove, the bottom border is a ridge, the left border is dashed, and the right border is inset.
3. **border-left-color**, **border-right-color**, **border-top-color**, **border-bottom-color**: These properties set the color of the left, right, top, and bottom borders, respectively. In this example, the left border is aqua, the right border is yellow, the top border is orange, and the bottom border is black.
4. **border-color**: This property sets the color of all four borders at once. In this example, it is commented out.
5. **background-color**: This property sets the background color of the element. In this example, it is commented out.
6. **border**: This property sets the width, style, and color of all four borders at once. In this example, it is set to 5px solid black, which means all four borders are 5px wide, solid, and black.
7. **width**: This property sets the width of the element. In this example, it is set to 100%, which means the element will take up the full width of its parent container.
8. **max-width**: This property sets the maximum width of the element. In this example, it is set to 1000px, which means the element will not be wider than 1000 pixels.
9. **max-height**: This property sets the maximum height of the element. In this example, it is set to 1000px, which means the element will not be taller than 1000 pixels.

**CSS FILE:**

.parent{

    border-top-width: 10px;

    border-bottom-width: 20px;

    border-left-width: 20px;

    border-right-width: 20px;

    border-top-style: groove;

    border-bottom-style:ridge;

    border-left-style: dashed;

    border-right-style: inset;

    border-left-color: aqua;

    border-right-color: yellow;

    border-top-color: orange;

    border-bottom-color:black;

    border: 5px solid black;

    width: 100%;

    max-width:1000px;

    max-height: 1000px;

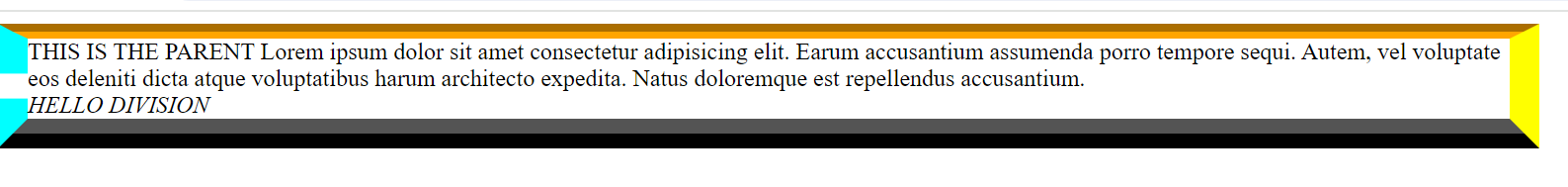
}

.child1{

    font-style: oblique;

}

**OUTPUT:**



**PADDING**:

Space between border of division and content

**MARGIN:**

Space given outside of the division

**BACKGROUND:**

Certainly! Let's break down the CSS properties used in your example for a better understanding:

### CSS Explanation:

1. \*\*#container\*\*:

- This is an ID selector targeting an HTML element with `id="container"`. IDs are unique identifiers in HTML, used to apply specific styles to a particular element.

2. \*\*height: 20rem;\*\* and \*\*width: 20rem;\*\*:

- Sets the height and width of the container to 20 "root em" units, which is relative to the root font size of the document.

3. \*\*border: 10px solid rgb(0, 255, 255);\*\*:

- Creates a border around the container. It's 10 pixels thick, solid (not dashed or dotted), and colored with an RGB value of cyan (`rgb(0, 255, 255)`).

4. \*\*font-size: 50px;\*\*:

- Sets the font size of the text inside the container to 50 pixels.

5. \*\*font-weight: 100;\*\*:

- Sets the font weight to 100, which is typically an ultra-light or thin weight, depending on the font family.

6. \*\*background-image: url(...);\*\*:

- Sets the background image of the container using a URL. In this case, it's fetching an image from Unsplash.

7. \*\*background-size: 100% 100%;\*\*:

- Specifies how the background image should be sized within the container. Here, `100% 100%` means the image should cover the entire container without stretching.

8. \*\*color: transparent;\*\*:

- Sets the text color to transparent, making the text invisible.

9. \*\*background-clip: text;\*\*:

- Clips the background image to the shape of the text. This means the background image is visible only where there is text content.

10. \*\*justify-content: center;\*\* and \*\*align-items: center;\*\*:

- These properties are typically used in flex containers. They horizontally (`justify-content`) and vertically (`align-items`) center the content inside the container.

**USAGE:**

- The CSS defines a container (`#container`) with specified dimensions, a cyan border, and an image background from Unsplash. The text inside the container is transparent, revealing the background image only within the text shapes. The text is centered both horizontally and vertically within the container. The font size is large (`50px`) and the font weight is light (`100`).

This combination of styles creates a visually interesting effect where the text appears filled with the background image while the text itself remains invisible.

Example:

<!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 id = "container">

        BACKGROUND IMAGE

    </div>

</body>

</html>

#container{

    height: 20rem;

    width: 20rem;

    border: 10px solid rgb(0, 255, 255);

    font-size: 50px;

    font-weight: 100;

    background-image: url(https://images.unsplash.com/photo-1556827739-8cdac8bc6855?q=80&w=871&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D);

    background-size: 100% 100%;

    color: transparent;

    background-clip:text;

    justify-content: center;

    align-items: center;

}

OUTPUT: 

**POSITION:**

**HTML:**

<!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="/POSITION\_CSS/style.css">

</head>

<body>

    <div id="parent">

        <div id="static">Static</div>

        <div id="relative">Relative</div>

        <div id="absolute">Absolute</div>

        <div id="fixed">Fixed</div>

        <div id="sticky">Sticky</div>

    </div>

</body>

</html>

**CSS:**

#parent {

    height: 1600px;

    width: 600px;

    background-color: aqua;

    border: 1px solid black;

    position: relative;

    /\* padding-top: 200px;  use this and see the difference \*/

  }

#static {

    height: 100px;

    width: 600px;

    background-color: yellow;

    border: 1px solid black;

    position: static;

  }

#relative {

    height: 100px;

    width: 600px;

    background-color: green;

    border: 1px solid black;

    position: relative;

    top: 20px;

    left: 60px;

}

#absolute {

    height: 100px;

    width: 600px;

    background-color: blue;

    border: 1px solid black;

    position: absolute;

    top: 20px;

    left: 60px;

}

#fixed{

    height: 100px;

    width:600px;

    background-color: bisque;

    position: fixed;

    top: 130px;

}

#sticky{

    height: 100px;

    width: 600px;

    background-color: darkcyan;

    top:10px;

    position: sticky;

}

**LINK FOR ICONS:**

Font awesome

**DISPLAY FLEX:**

 **display: flex;:** Turns the parent container into a flex container, allowing flexible layout of its child elements.

 **align-items: flex-start;:** Aligns the flex items (children) to the start of the cross axis (top in this case).

**flex-direction**

The flex-direction property in CSS determines the direction of the main axis of a flex container and specifies how flex items are laid out in that container.

**Values:**

* **row**: Items are laid out horizontally in the order they appear in the source code (left to right).
* **row-reverse:** Items are laid out horizontally in reverse order (right to left).
* **column**: Items are laid out vertically from top to bottom.
* **column-reverse**: Items are laid out vertically in reverse order (bottom to top).

### justify-content

The justify-content property aligns flex items along the main axis of the flex container.

#### Values:

* **flex-start**: Items are packed toward the start of the main axis.
* **flex-end**: Items are packed toward the end of the main axis.
* **center**: Items are centered along the main axis.
* **space-between**: Items are evenly distributed with the first item at the start and the last item at the end.
* **space-around**: Items are evenly distributed with equal space around them.
* **space-evenly**: Items are evenly distributed with equal space around them, including before the first and after the last item.

### flex-wrap

The flex-wrap property controls whether flex items are forced onto a single line or can wrap onto multiple lines.

#### Values:

* **nowrap**: All flex items are forced onto a single line.
* **wrap**: Flex items can wrap onto multiple lines if needed.
* **wrap-reverse**: Flex items wrap onto multiple lines in reverse order **\**

**TRANSFORMS:**

transform: rotateX(45deg) rotateY(45deg) rotateZ(45deg);

rotate X:Rotates our div based on x axis

rotate Y: Rotates our div based on y-axis

rotate Z:Rotates our div based on Z-axis

**BUTTONS:**

https://uiverse.io/buttons

**3-D transforms:**