FRONT-END WEB APPLICATION DEVELOPMENT

ASSIGNMENT -2

SUBMITTED BY:

AYUSH KUMAR JHA

SAP ID - 500086400

Enrollment no - R200220083

B.C.A -I.O.T.

Q1. Write HTML code to create two buttons “coin” and “dice”. Also, write javascript functions for both. For example, when a user presses the button “dice” it must return a random number from 1 to 6.

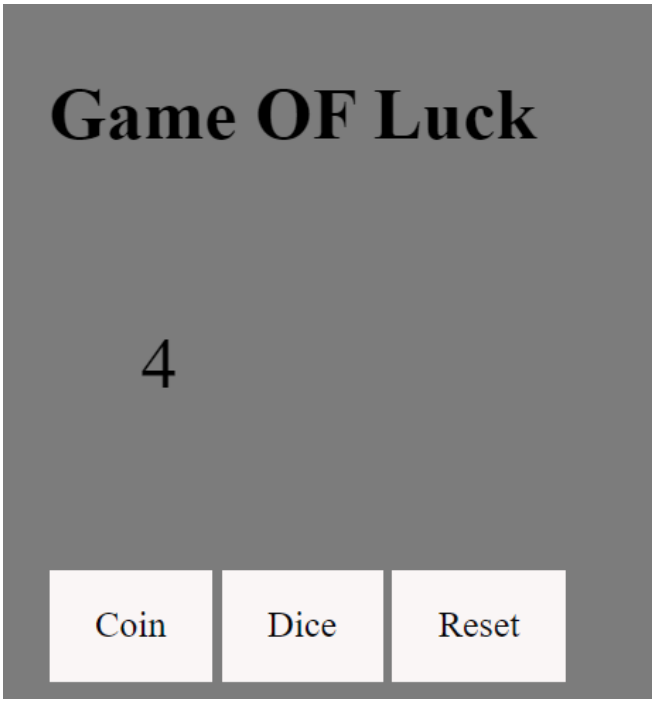
Sol.
Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        *{
            font-family:serif;
        }
        .no1 {
            background-color: rgb(124, 124, 124);
            width: 25%;
            padding: 10px 20px;
            zoom: 130%;
        }
        .no1 p {
            zoom: 200%;
            padding: 20px 20px;
        }
        .no1 button {
            background-color: rgb(250, 246, 246);
            border: none;
            color: rgb(8, 8, 11);
            padding: 15px 20px;
            text-align: center;
            text-decoration: none;
```

```
            display: inline-block;
            font-size: 16px;
        }
        .no1 button:hover {
            cursor: pointer;
        }
    </style>
</head>
<body>
    <script>
        function up(){
            var a = Math.floor(Math.random() * (100 - 1) ) + 1;
            if(a%2==0){
                document.getElementById("val").innerHTML= "H";
            }
            else{
                document.getElementById("val").innerHTML= "T";
            }
        }
        function down(){
            var a = Math.floor(Math.random() * (7 - 1) ) + 1;
            document.getElementById("val").innerHTML= a;
        }
    </script>
    <div class="no1">
        <h1>Game OF Luck</h1>
        <p id="val">0</p>
        <div class="ba">
            <button class="b" type="button" onclick="up()">Coin</button>
            <button class="b" type="button" onclick="down()">Dice</button>
                                            <button      class="b"      type="button"
onclick="document.getElementById('val').innerHTML= 0;">Reset</button>
        </div>
    </div>
```

```
</body>
</html>
```

OutPUt:-



Q2. Write a code that asks the user to enter a string expression with numbers and addition operators. Also, write a javascript function “solve” that takes that expression and returns the result as a number. For example, 5+4+3 must return 12.

Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>
<script>
var a=0;
function up(){
```

```
var inp = prompt("Input string");
document.getElementById("val").innerHTML= inp;
}

function down(val){
const myArray = val.split("+");
var sum=0;
for( i = 0 ; i<myArray.length;i++){
sum+=parseInt(myArray[i]);
}
document.getElementById("val2").innerHTML= sum;

}

</script>
<p id="val">0</p>
<p id="val2">Solution</p>

<button class="b" type="button" onclick="up()">Input</button>
<button class="b" type="button" onclick="down(val.innerHTML)">Solve</button>
</body>
</html>
```

OutPut:-

Q3. Write code for a convertor. It will have two div sections, one for Degree Celsius to Fahrenheit and the other for Fahrenheit to Degree Celsius. Each section will have an input field for the user to enter the number and the other where the result is displayed as soon as the user enters the data in the first field.

Code:-

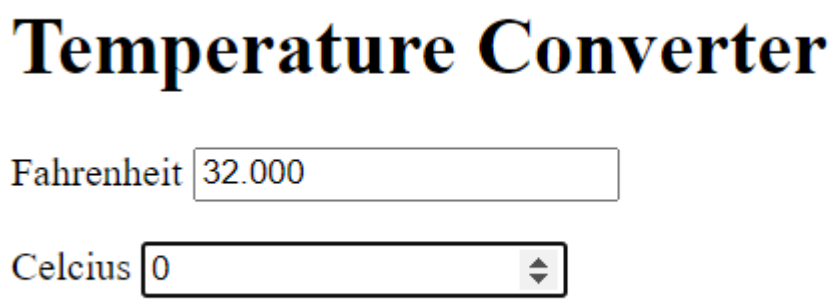
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
  function Converter() {
    var inp = document.getElementById("Fahr").value;
        inp = parseFloat(inp);
    celcius = ((inp-32)/1.8).toFixed(3);
    document.getElementById("Cel").value=celcius;
  }
  function Converter2(){
    var inp = document.getElementById("Cel").value;
      inp =parseFloat(inp);
      fahr=((inp*1.8)+32).toFixed(3);
      document.getElementById("Fahr").value=fahr;
  }
  </script>

  <h1>Temperature Converter</h1>

  <p>
    <label>Fahrenheit</label>
```

```
    <input id="Fahr" type="number" placeholder="Fahrenheit" oninput="Converter()"
onchange="Converter()">
  </p>
  <p>
    <label>Celcius</label>
    <input id="Cel" type="number" placeholder="Celcius" oninput="Converter2()"
onchange="Converter2()">
  </p>
</body>
</html>
```

OutPut:-



Q4. Write code to create and style a responsive grid architecture with 5 columns.

Code:-

```
<!DOCTYPE html>
<html>
<head>
<style>

.grid-container {
  display: grid;
  grid-template-columns: 15% 35% 20% 15% 15%;
  padding: 10px;
}
.grid-item {
```

```
  background-color: rgb(83, 202, 220);
  border: 3px solid rgb(0, 0, 0);
  color: black;
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
</head>
<body>

<h1>Grid Architecture with 5 columns.</h1>
<div class="grid-container">
  <div class="grid-item">A</div>
  <div class="grid-item">B</div>
  <div class="grid-item">C</div>
  <div class="grid-item">D</div>
  <div class="grid-item">E</div>
</div>

</body>
</html>
```

OutPut:-

Grid Architecture with 5 columns.

Q5. Explain the following with examples:

- Multi-step animation

  That's the concept of multi-step animations in a nutshell: more than one change taking place in the animation from start to finish.

  CSS animations are rad and the concept is fairly simple. Name the animation, define the movement in @keyframes and then call that animation on an element. If you haven't worked with them, you can level up on the syntax While the concept is simple, there are little tricks to make the animations seem complex and one of those is multi-step transitions. That's what we're going to look at in this post.

  1 keyframe can be a "step"

  If we set up a keyframe animation to change the background color of an element to change from orange to black (because orange is the new black, after all) on hover over five seconds, it will do exactly that. It will divide that change up over time and make the transition.

  We can add as many steps as we like in a keyframe animation. For example, here is blue being added right in the middle of the transition.

  That's the concept of multi-step animations in a nutshell: more than one change taking place in the animation from start to finish.

- Matrix property in Transform

  matrix()

  The matrix() CSS function defines a homogeneous 2D transformation matrix. Its result is a <transform-function> data type.

  Syntax

  The matrix() function is specified with six values. The constant values are implied and not passed as parameters; the other parameters are described in the column-major order.

  matrix(a, b, c, d, tx, ty)

  Matrix Transform

  CSS Transform property allows to scale, rotate, skew and move HTML elements.

  1) Scale - resize elements(small or bigger)
  2) Rotate - by angle about the origin
  3) Skew - transformation along the X or Y axis
  4) Translate - move element in XY direction

  linear transformations also can be represented by Matrix function. It combine multiple transform properties into single matrix function. Thanks to this wikipedia image which makes clear everything about matrix transformation.

- Responsive media queries

  Media query is a CSS technique introduced in CSS3.

  It uses the @media rule to include a block of CSS properties only if a certain condition is true.

  Example

If the browser window is 600px or smaller, the background color will be lightblue:

```
@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

Add a Breakpoint

Earlier in this tutorial we made a web page with rows and columns, and it was responsive, but it did not look good on a small screen.

Media queries can help with that. We can add a breakpoint where certain parts of the design will behave differently on each side of the breakpoint.

Always Design for Mobile First

Mobile First means designing for mobile before designing for desktop or any other device (This will make the page display faster on smaller devices).

This means that we must make some changes in our CSS.

Instead of changing styles when the width gets smaller than 768px, we should change the design when the width gets larger than 768px. This will make our design Mobile First:

Another Breakpoint

You can add as many breakpoints as you like.

We will also insert a breakpoint between tablets and mobile phones.