Project that shows live time and keeps on updating.

Description: The website will simply have a digital clock that shows the current time and date.

This can be implemented using basic html, css and javascript.

Rough Diagram of how the website will look:



Elements:

Navbar:

Home:

Time Zones: Study about how the time are calculated for various places using GMT.

Change to analog: Changes the content to analog i.e., to an analog clock.

Contacts will simply contain a form to give feedback. Home will contain the time of India.

The digital clock can be implemented using the timeInterval functions.

For analog clock:

Elements and Techniques used.

1. Create a nav bar: Create using simple CSS.

ul {

list-style-type: none;

margin: 0;

padding: 0;

background-color: #333;

overflow: hidden;

// The **overflow** property specifies what should happen if content **overflows** an element's box. This property specifies whether to clip content or to add scrollbars when an element's content is too big to fit in a specified area.

position: relative;

top: 0;

width: 100%;

}

li {

float: left;

//The **float CSS** property places an element on the left or right side of its container, allowing text and inline elements to wrap around it. The element **is** removed from the normal flow of the page, though still remaining a part of the flow

border-right: 1px solid white;

}

li a {

display: block;

padding: 14px 16px;

text-decoration: none;

color: white;

text-align: center;

}

li a:hover:not(.active) {

background-color: #111;

}

.active {

background-color: #4caf50;

}

1. Creating a dropdown list at the navbar links:

This can be used for Zones element of the navbar. Here we can have different time zones name and depending on the time zone that the user clicks, the time at that location can be changed.

**Creating a dropdown menu:**

HTML for creating the dropdown menu:

<li class="DropDown">

<a href="javascript:void(0)" class="Button">Dropdown</a>

<div class="DropContent">

<a href="#">Link 1</a>

<a href="#">Link 2</a>

<a href="#">Link 3</a>

</div>

</li>

Now for the CSS: Note everything remains the same but here we need to add the design for class DropDown, Button and DropContent classes.

.DropDown {

display: inline-block;

}

.DropContents {

display: none;

position: absolute;

min-width: 160px;

max-height: 200px;

background-color: #f9f9ff;

box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);

z-index: 1;

overflow: scroll;

}

.DropContents a {

display: block;

text-align: left;

padding: 12px 16px;

color: black;

}

.DropContents a:hover {

background-color: aquamarine;

}

.DropDown:hover .DropContents {

display: block;

}

1. Now we are creating a jumbotron like box at the middle where we are going to display the Date and Time.

Here we a using a simple div with width and height constraints.

After this, we are going to create a div inside the box that we have just created to hold the time and date attributes:

<div class="Clock">

<h2>Indian Time:</h2>

<div class="TimeConstraints">

<h3>Time: </h3>

<h3>Date: </h3>

</div>

</div>

For the CSS:

.Clock {

background-color: rgb(41, 40, 40);

width: 600px;

height: 300px;

margin: auto;

margin-top: 50px;

border-radius: 5px;

padding: 10px;

}

.Clock h2 {

color: white;

text-align: center;

}

.TimeConstraints h3 {

color: white;

}

1. Now for the date and time, we will use javascript.

Here we will use setTimeout and setInterval functions to ensure that the screen gets continuously update and

And we are able to see the change in seconds.

date = () => {

let t = document.getElementById("Time")

let d = document.getElementById("Date")

let myDate = new Date()

let a;

if(myDate.getHours() < 12 && myDate.getMinutes()<60)

a = "am"

else

a = "pm"

let date = myDate.getDate() +"/"+ myDate.getMonth() + "/" + myDate.getFullYear()

let time = myDate.getHours() + ":" + myDate.getMinutes() + ":" + myDate.getSeconds() + " " + a

t.innerText = "Time: " + time

d.innerText = "Date: " + date

console.log("Hello")

}

clr = setInterval(date, 1000)

StopInterval = () =>{

clearInterval(clr)

alert("Timer stopped")

}

setTimeout(StopInterval, 20000)