**PROJECT REPORT**

**ON**

**“ iDragon Game Using**

**Html Css & Javascript ”**

***Report submitted to***

ITM University, Raipur, Chhattisgarh

***for the partial fulfilment of the award of the degree***

***of***

**Bachelor of Technology**

**in**

**Computer Science Engineering**

Specialisation - Cloud Technology & Information security

***Submitted by***

**Name of the Student : Enrollment No :**

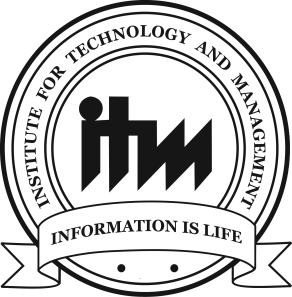
Sohil Kumar Chaturvedi H0131

Under the Supervision of

**Prof. Dr Raktim Sir**

0

ITM UNIVERSITY



**School of Engineering and Research**

**ITM University , Uparwara , Naya Raipur**

**Raipur , Chhattisgarh - 492002**

**DECLARATION**

I am **Sohil Kumar Chaturvedi** the student of Bachelor of Technology in **Computer Science** **Engineering** under the School of Engineering and Research in ITM University, Naya Raipur ,Chhattisgarh hereby declare that the work contained in this Project report is original and has been done by me under the guidance of my supervisor .

The work has not been submitted to any other University/Institute for any degree or diploma. I have followed the guidelines provided by the University in preparing the dissertation report .

The used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references.

**Name of the student Enrollment No. Signature**

Sohil Kumar Chaturvedi H0131

**Date :**

**Signature : Signature :**

Internal Examiner Department Co-ordinator

**Asst. Prof. Sharmistha Puhan**  Department of Computer - Science

& Engineering

**Signature : Forwarded By :**

Internal Examiner Head of Department of

School of Engineering and Research

ITM University, Raipur

**Signature :**

Department of Computer Science Engineering

ITM University, Raipur

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to all those people who have given their heart willing support in making this completion a magnificent experience. Firstly , I would like to thank to my mentor **Dr. Puja Shrivastav Mam** , **Assistant Professor , SER , ITM University** for giving me the opportunity to do an Project within the organization , he was helping me in every problems , learnings , providing necessary notes . I am thankful to **Dr SatyaPrakash Makhija** , Head of Department of School of Engineering and Research , ITM University Raipur , for providing us good and healthy environment for the preparation of this Project . I am also thankful to my Supervisor  **Professor**

**Dr. Raktim Sir ,** who was in the Computer Science Department , for his timely comments and suggestions . He advised on the details of my work and provides valuable discussions . Without the guidance of my supervisor , this Project may not have well materialized . I am overwhelmed by the constant support and needful motivation given by the Computer Science Engineering Department Faculty . I am really grateful to my parents for their support, appreciation and encouragement . I specially acknowledge the authors of different research papers and books for providing necessary contribution to my work .

**ABSTRACT**

**INTRODUCTION :-**

We made this website with Html and Basic CSS .

It is a Full Responsive Wesite .

We create different index file for every single navigation bar to make option working for the website and this could help us to move another page by Clicking the link index from Home Page .

**OBJECTIVES :-**

* Our Front design skills will be developed .
* We can make any types of Webpages .
* Our Html knowledge will be increases .
* Our CSS skills will be more effective .

**PROGRAMMER AND OPPORTUNITIES :**

* Full stack web Developer
* Website Designer
* Website Maintanence
* Software Engineer

**METHODOLOGY :**

1. HTML
2. CSS
3. PYTHON
4. FRONT-END TOOLS
5. WEB TOOLS
6. BACK-END TOOLS

**JOB SCOPE :-**

* Web Developer
* Junior Web Developer
* Senior Web Developer
* Graphic Designers
* Computer Programmer
* Web Desingner
* Web Architect and Webmaster

**TABLE OF CONTENTS**

Certificate declaration 1

Acknowledgement 2

Abstract- Introduction , Objective , Programmers & Opportunities , Methodology , Job Scope

PROJECT INTRODUCTION 3

* + 1. HTML CODE – html
    2. CSS CODE - style.css
    3. JAVASCRIPT CODE - script.js

**Let’s Started Coding ……**

**Chapter 1 Introduction to web 4**

* + 1. Introduction 5
    2. History of web 6
       1. Web Browser 7
       2. Website 8
       3. Webpages 9
    3. Client Server Architecture 10

**Chapter 2 HTML 11**

1. Introduction 5
2. History of html 6
3. Html Tags 7
   * + 1. Image 8
       2. Button
       3. Heading
       4. Links
       5. Documents
4. Html Tables & Forms 10

**Chapter 3 CSS 11**

i. Introduction

ii. History of CSS 11

iii. Syntax 11

iv. Types of CSS 11

Internal

External

Inline

**MOTIVATION 11**

**FUTURE SCOPE 11**

CONCLUSION 11

REFERENCES 11 PROJECT INTRODUCTION :-

“Dragon’s World” is a game in which one dragon tries to save itself from the other dragon by jumping over the dragon which comes in its way. The score is updated when one dragon saves himself from the other dragon.

The project will contain HTML, CSS and JavaScript files. The HTML file adds structure to the game followed by styling using CSS. JavaScript adds functionality to the game .

**File Structure :-**

* Index.html
* Style.css
* Script.js

**HTML CODE**

* **Heading portion:** It will show the name of the game.
* **Game over portion:** It will be shown when you lose the game.
* **Obstacle portion:** It will contain the obstacle from which the dragon has to save itself.
* **Dragon portion:** It will contain the dragon which has to be saved from the obstacle i.e. the other dragon.
* **Score portion:** It will show the current score of the game.

**HTML**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content=

"width=device-width, initial-scale=1.0">

<link rel="stylesheet" href=

"style.css?\_cacheOverride=1606401798626">

<link href=

"https://fonts.googleapis.com/css2?family=Ubuntu:ital,

wght@0, 300;1, 700&display=swap"

rel="stylesheet">

</head>

<body>

<h1 id="gameName">Welcome to Dragon's world</h1>

<div class="container">

<div class="gameover">Game Over</div>

<div id="scorecount">Your score : 0</div>

<div class="obstacle animateobstacle"></div>

<div class="dragon" style="left: 426px;"></div>

</div>

</body>

</html>

**CSS CODE**

* **Positioning the game’s name:** The name of the game is positioned by the *absolute* property of CSS.
* **Background image styling:** In the *container* class, we have put the background image of the game with the *background-size* set to *cover*.
* **Score Card styling:** We have positioned the scorecard to top right of the page and also provided it with a suitable background color to make it attractive. The text in it would be shown in *white*.
* **Obstacle image styling:** We have positioned the obstacle to bottom left of the page and provided animation to it so that it could move towards left.
* **Dragon’s styling:** We have positioned the dragon to bottom left of the page and provide animation to it so that it could jump up and save himself.
* **Game Over Styling:** We have positioned the game over portion to the center of the page and it will appear when the dragon is hit by the obstacle.

**Style.css**

/\* CSS Reset \*/

\*{

margin:0px;

padding:0px;

}

body {

/\* Hides the bottom scrollbar \*/

overflow: hidden;

}

/\* Styling of the Game's Name \*/

#gameName {

position: absolute;

top:30vh;

left:38vw;

}

/\* Background image styling \*/

.container {

background-image: url(cover.png);

background-size: cover;

width:100vw;

height:100vh;

}

/\* ScoreCard Styling \*/

#scorecount {

position: absolute;

top:20px;

right:20px;

background-color: black;

padding: 28px;

border-radius: 20px;

color: white;

}

/\* Obstacle image styling and positioning \*/

.obstacle {

background-image: url(obstacle.png);

background-size: cover;

width:154px;

height: 126px;

position: absolute;

bottom:0px;

right:120px;

}

/\* Applying animation to the obstacle

so that it can move towards left \*/

.animateobstacle {

animation: aniob 5s linear infinite;

}

@keyframes aniob {

0% {

left:100vw;

}

100% {

left:-10vw;

}

}

/\* Dragon's styling \*/

.dragon {

background-image: url(dragon.png);

background-size: cover;

width: 194px;

height: 126px;

position: absolute;

bottom:0px;

left:90px;

}

/\* Applying animation to the dragon so

that it can save himself by jumping up \*/

.animatedragon {

animation: ani 1s linear;

}

@keyframes ani {

0% {

bottom:0px;

}

25% {

bottom:150px;

}

50% {

bottom:300px;

}

75% {

bottom:211px;

}

100% {

bottom:0px;

}

}

/\* gameover styling and positioning \*/

.gameover {

visibility: hidden;

font-family: 'Ubuntu', sans-serif;

position: absolute;

top: 50vh;

left: 35vw;

color: red;

font-weight: bold;

font-size: 6rem;

background-color: firebrick;

border-radius: 20px;

}

**JAVASCRIPT CODE**

**1. Movement of the dragon :** This is provided by the *onkeydown* event .

* **Up arrow key:** On pressing it, the dragon will jump upwards (animation provide by CSS).
* **Left arrow key:** On pressing it, the dragon will move to the left (animation provide by CSS).
* **Right arrow key:** On pressing it, the dragon will move to the left (animation provide by CSS).

document.onkeydown = function(e) {

console.log(e.keyCode);

if (e.keyCode == 38) {

dragon = document.querySelector('.dragon');

dragon.classList.add('animatedragon');

setTimeout(() => {

dragon.classList.remove('animatedragon');

}, 700);

}

if (e.keyCode == 37) {

dragon = document.querySelector('.dragon');

dragonx = parseInt(window.getComputedStyle(dragon, null)

.getPropertyValue('left'));

dragon.style.left = dragonx - 112 + "px";

}

if (e.keyCode == 39) {

dragon = document.querySelector('.dragon');

dragonx = parseInt(window.getComputedStyle(

dragon, null).getPropertyValue('left'));

dragon.style.left = dragonx + 112 + "px";

}

}

**2. Updating the score:** The score is satisfied only when a given condition is satisfied. We will compute the left and bottom values of both the obstacle and the dragon and then increase the score based on a proper value which shows that the dragon has saved himself from the obstacle. For this, we have taken “cross” variable and assigned “true” to it. When the dragon crosses the obstacle safely, we set the value to “false”. After approximately 1s we change the value of cross to “true”. We have also made the obstacle run faster after each cross and thus increasing the difficulty level.

setInterval(() => {

dragon = document.querySelector('.dragon');

gameover = document.querySelector('.gameover');

obstacle = document.querySelector('.obstacle');

dx = parseInt(window.getComputedStyle(

dragon, null).getPropertyValue('left'));

dy = parseInt(window.getComputedStyle(

dragon, null).getPropertyValue('bottom'));

ox = parseInt(window.getComputedStyle(

obstacle, null).getPropertyValue('left'));

oy = parseInt(window.getComputedStyle(

obstacle, null).getPropertyValue('bottom'));

offsetx = Math.abs(dx - ox);

offsety = Math.abs(dy - oy);

console.log(offsetx, offsety);

if (offsetx < 120 && offsety <= 144) {

gameover.style.visibility = 'visible';

obstacle.classList.remove('animateobstacle');

} else if (offsetx < 125 && cross) {

score += 1;

updateScore(score);

cross = false;

setTimeout(() => {

cross = true;

}, 1000);

setInterval(() => {

obsanidur = parseFloat(window

.getComputedStyle(obstacle, null)

.getPropertyValue('animation-duration'));

obstacle.style.animationDuration

= obsanidur - 0.01 + 's';

}, 500);

}

}, 10);

function updateScore(score) {

scorecount.innerHTML = "Your score : " + score;

}

**Script.js**

<script>

cross = true;

score = 0;

document.onkeydown = function(e) {

console.log(e.keyCode);

if (e.keyCode == 38) {

dragon = document.querySelector('.dragon');

dragon.classList.add('animatedragon');

setTimeout(() => {

dragon.classList.remove('animatedragon');

}, 700);

}

if (e.keyCode == 37) {

dragon = document.querySelector('.dragon');

dragonx = parseInt(window.getComputedStyle(

dragon, null).getPropertyValue('left'));

dragon.style.left = dragonx - 112 + "px";

}

if (e.keyCode == 39) {

dragon = document.querySelector('.dragon');

dragonx = parseInt(window.getComputedStyle(

dragon, null).getPropertyValue('left'));

dragon.style.left = dragonx + 112 + "px";

}

}

setInterval(() => {

dragon = document.querySelector('.dragon');

gameover = document.querySelector('.gameover');

obstacle = document.querySelector('.obstacle');

dx = parseInt(window.getComputedStyle(

dragon, null).getPropertyValue('left'));

dy = parseInt(window.getComputedStyle(

dragon, null).getPropertyValue('bottom'));

ox = parseInt(window.getComputedStyle(

obstacle, null).getPropertyValue('left'));

oy = parseInt(window.getComputedStyle(

obstacle, null).getPropertyValue('bottom'));

offsetx = Math.abs(dx - ox);

offsety = Math.abs(dy - oy);

console.log(offsetx, offsety);

if (offsetx < 120 && offsety <= 144) {

if (score != 0)

scorecount.innerHTML = "Your score : " + score;

gameover.style.visibility = 'visible';

obstacle.classList.remove('animateobstacle');

} else if (offsetx < 125 && cross) {

score += 1;

updateScore(score);

cross = false;

setTimeout(() => {

cross = true;

}, 1000);

setInterval(() => {

obsanidur = parseFloat(window

.getComputedStyle(obstacle, null)

.getPropertyValue('animation-duration'));

obstacle.style.animationDuration

= obsanidur - 0.01 + 's';

}, 500);

}

}, 10);

function updateScore(score) {

scorecount.innerHTML = "Your score : " + score;

}

</script>

**Let’s get started Coding…….**

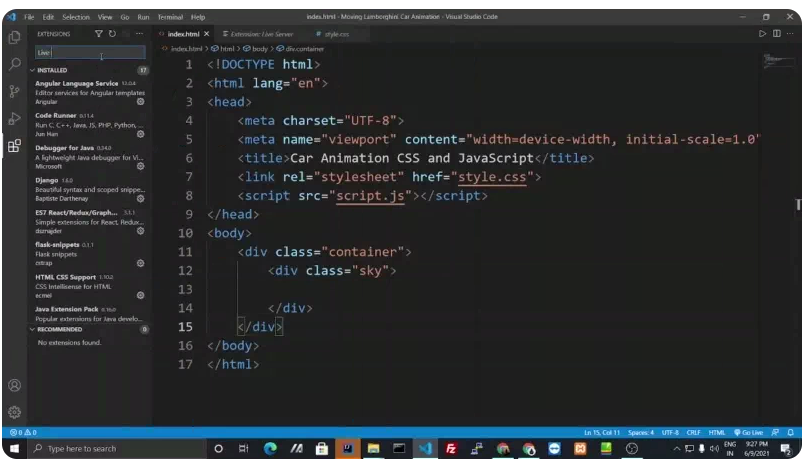
we will make an interesting game named iDragon. We will make this possible with the help of HTML, CSS, and pure Javascript without any external library.

We will add music, animation, score features, etc in this game and code this game in a way where you can challenge your friends to play and break your score.

Let's open VS code in the folder. We will make 3 files, named index.html(HTML file,

to write HTML), style.css(CSS file, where we will do the styling), and

script.js(JavaScript file, in this file we will keep scripting of frontend).



Let's start by creating the structure of the web page using HTML. So in the index.html file, we will construct one main div block with a class named “**gameContainer”** inside the <body>  tag.

<body>

<div class="gameContainer">

</div>

</body>

Now inside the div block with **class “gameContainer”,** We will create three more div blocks with class names as “gameOver”(for displaying when the game is over), **“dino**”(for dinosaur), and “**obstacle**”(for dragon). Also, create one div block with an id named “**scorecount”(**for displaying score).

<div class="gameContainer">

<div class="gameOver">Welcome to the idragon game!!</div>

<div class="dino"></div>

<div id="scoreCount"></div>

<div class="obstacle"></div>

</div>

* **Styling background and game area in a web page**

We will start styling in style.css file by adding a universal selector to reset CSS and target the body to set as overflow is hidden. Then we will target the element by class and will add some CSS properties to set the background image and the size of the gameContainer as shown below :-

\*{

margin: 0;

Padding: 0;

}

body{

overflow: hidden;

}

.gameContainer{

background-image: url(bg.png);

background-repeat: no-repeat;

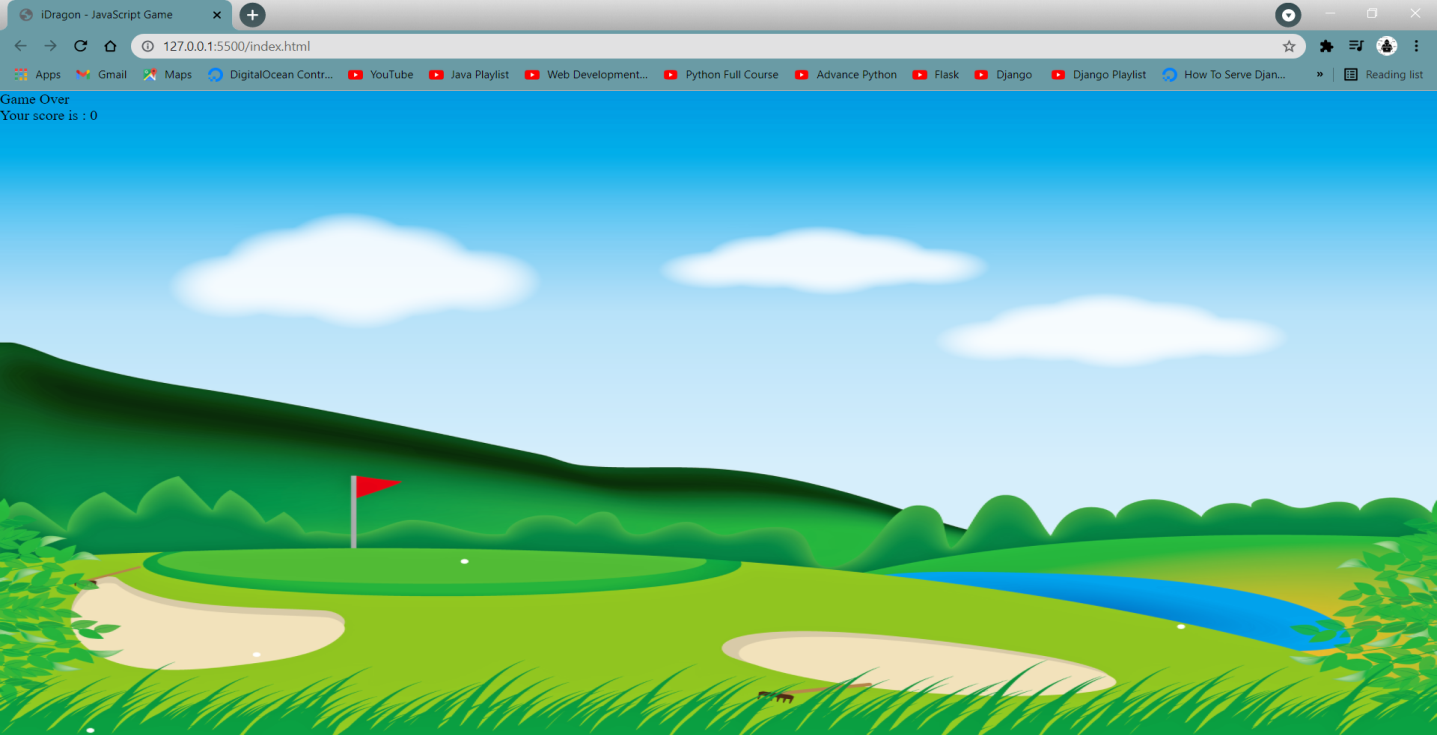
background-size: 100vw 100vh;

width: 100%;

height: 100vh;

}

**Now our webpage will look like this:**



* **Creating 1st Character Dinosaur (dino)**

We will create our first character dino by targeting the div block with class name “dino” and will add some CSS properties to it like background-image, background-repeat, background-size, width, height, position, bottom, and left. For creating, and locating this character.

.dino{

background-image: url(dino.png);

background-repeat: no-repeat;

background-size: cover;

width: 233px;

height: 114px;

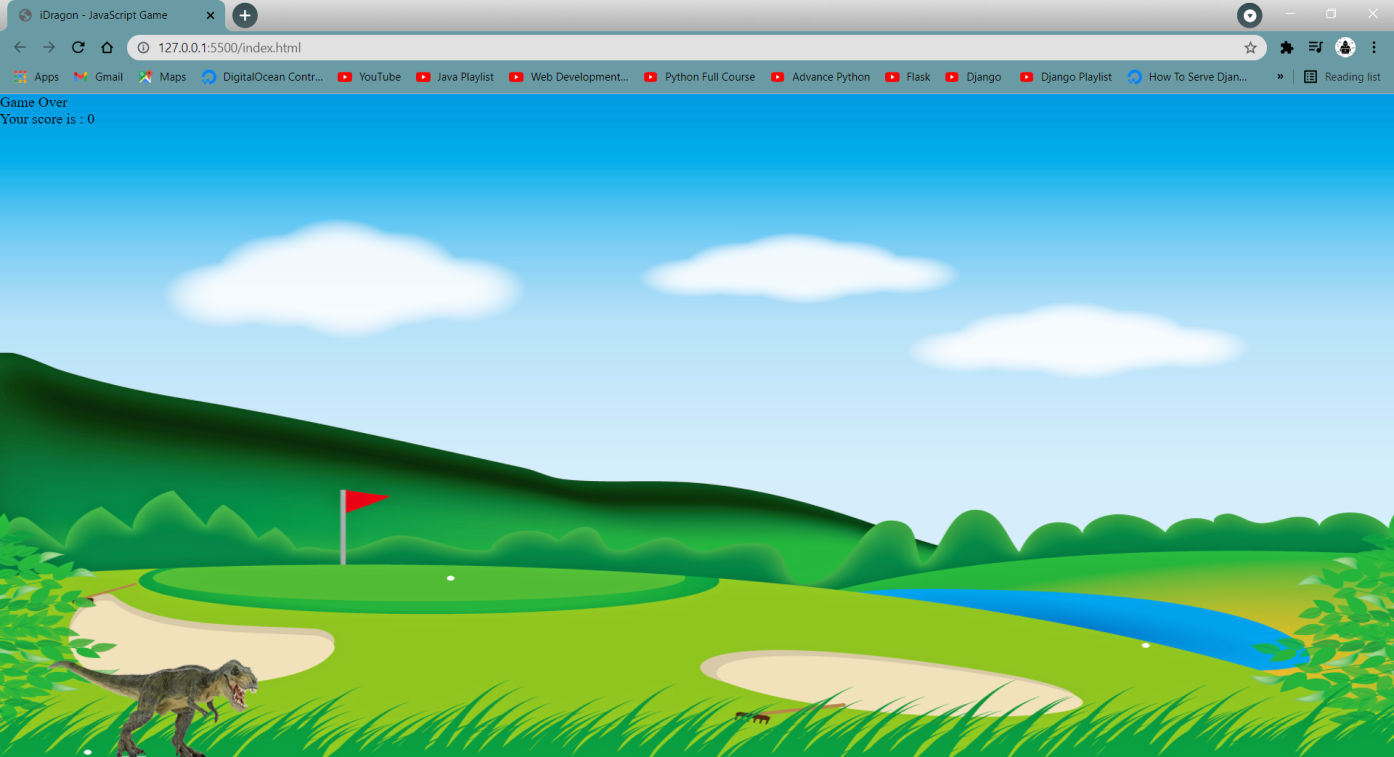
position: absolute;

bottom:0;

left: 52px;

}

**Now the webpage will look like this:**

****

* **Creating 2nd Character Dragon (obstacle)**

Now we will create an obstacle of our game by targeting the div block with class name that is “obstacle” and will add some CSS properties to it like background-image, background-repeat, background-size, width, height, position, bottom, and left. For creating, and locating for this character.

.obstacle{

background-image: url(dragon.png);

background-repeat: no-repeat;

background-size: cover;

width: 166px;

height: 113px;

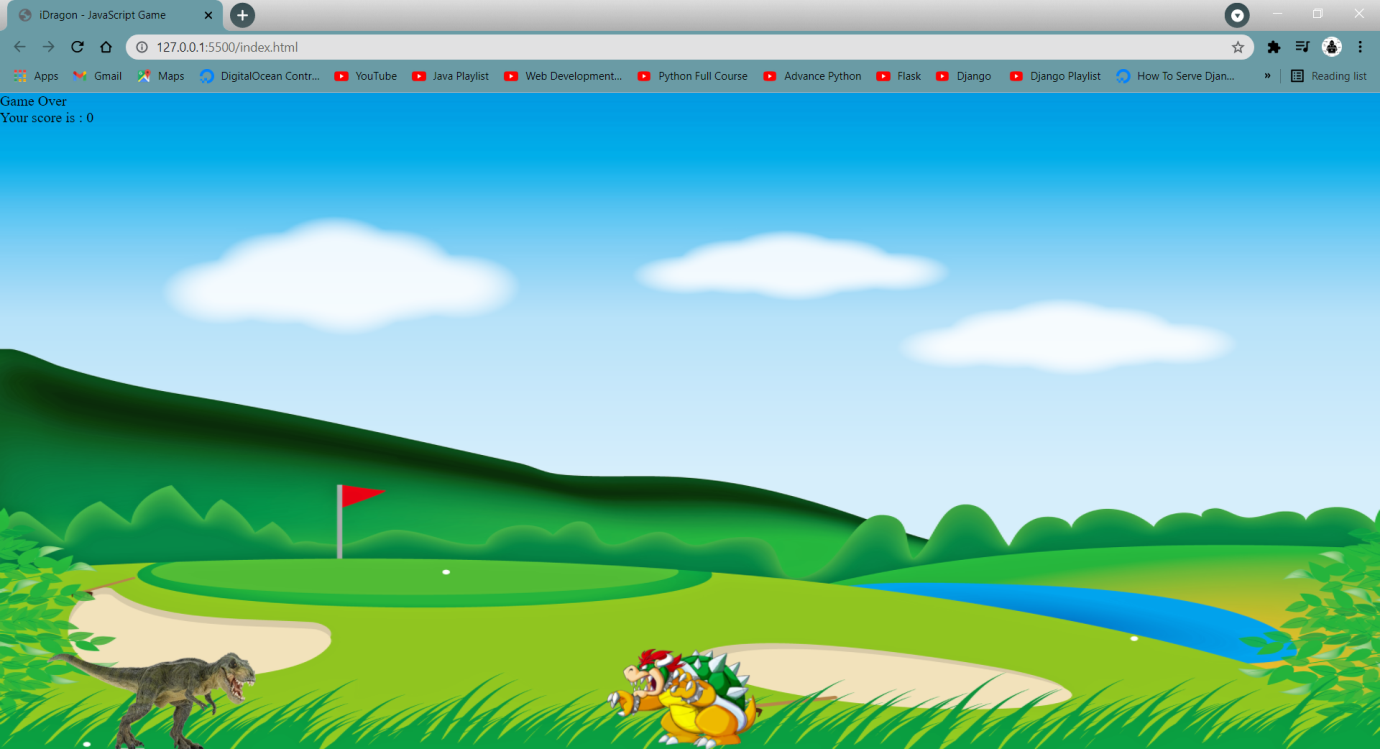
position: absolute;

bottom: 0;

left: 44vw;

}

**Now the webpage will look like this:**

****

**CSS animations** make it possible to animate transitions from one CSS style configuration to another. Animations consist of two components, a style describing the CSS animation and a set of keyframes that indicate the start and end states of the animation’s style, as well as possible intermediate waypoints.

**Creating an animation to make the dinosaur jump**

***CSS Syntax of @keyframes :***

@keyframes animationname {keyframes-selector {css-styles;}}

We will create the animation named **dino**and set three keyframes for the bottom level at **0% = 0px**, **50% = 422px** and again at **100% = 0px.** So that it will look like an animation of a jumping dinosaur.

@keyframes dino{

0%{

bottom: 0;

}

50%{

bottom: 422px;

}

100%{

bottom: 0;

}

}

***CSS Syntax for applying animation :***

animation: name duration timing-function delay iteration-count direction fill-mode play-state;

Then we will create a class named **animateDino**and apply this animation named dino into it as shown below :

.animateDino{

animation: dino 0.6s linear;

}

CSS file will look as follows :

.animateDino{

animation: dino 0.6s linear;

}

@keyframes dino{

0%{

bottom: 0;

}

50%{

bottom: 422px;

}

100%{

bottom: 0;

}

}

* **Scripting for dinosaur in JavasScript**

We will write the script for our character(dinosaur).  Firstly, we will use an onkeydown event of javascript to provide a code indicating which key is pressed, while keypress indicates which character is entering. For example, a lowercase "a" will be reported as 65 by key down. We can see the keycode of any key(on the keyboard) by pressing it using this code as shown below or you can find them easily on the internet.

document.onkeydown = function (e) {

console.log("Key code is: ", e.keyCode)

}

So the keycode of the up arrow key is 38 by keydown.

The document method **querySelector()** returns the first element within the document that matches the specified selector, or group of selectors. If no matches are found, null is returned.

**Syntax :**

element = document.querySelector(selectors);

Here we have to query the class ‘.dino’, then we will add a class named animateDino (for jump) which we have created in style.css (css file).

dino = document.querySelector('.dino');

dino.classList.add('animateDino');

We also have to remove a class named animateDino(for jump) after some time, otherwise, our character will jump continuously so for that, we will use setTimeout().

The **setTimeout()** method sets a timer which executes a function or specified piece of code once the timer expires. Here we will remove a class named animateDino(for jump) after 700 milliseconds from dino using setTimeout() method.

setTimeout(() => {

dino.classList.remove('animateDino')

}, 700);

We have to write the code in a way when the up arrow key is pressed, a class named animateDino(for jump) should apply and after 700 milliseconds the class should be removed as shown below :

document.onkeydown = function (e) {

console.log("Key code is: ", e.keyCode)

if (e.keyCode == 38) {

dino = document.querySelector('.dino');

dino.classList.add('animateDino');

setTimeout(() => {

dino.classList.remove('animateDino')

}, 700);

}

}

* **Creating an animation to run the obstacle**

We will create an animation named obstacleAni and set two keyframes for left direction at 0% = 100vw(viewport width), and at 100% =  -10vw(viewport width). So that it will look like an animation of a dragon running.

@keyframes obstacleAni{

0%{

left: 100vw;

}

100%{

left: -10vw;

}

}

Now it's time to create a class **obstacleAni**where we will apply this animation named as obstacleAnias shown below :

.obstacleAni{

animation: obstacleAni 3s linear infinite;

}

* **Logic of game over**

To over the game we have to detect the collision between our dino and obstacle, for that we will use the **setInterval()** method. The **setInterval()** method calls a function or evaluates an expression at specified intervals(in milliseconds).

**Syntax :**

setInterval(function, milliseconds, param1, param2, ...)

Here our strategy will be to check the distance continuously in an interval of 10 milliseconds by the setInterval() method.

We will select these three classes “.dino”, “.gameOver”, and “.obstacle” by using **querySelector().**

dino = document.querySelector('.dino');

gameOver = document.querySelector('.gameOver');

obstacle = document.querySelector('.obstacle');

Now we will use the **window.getComputedStyle()**method which returns an object containing the values of all CSS properties of an element, after applying active stylesheets and resolving any basic computation those values may contain.

***Syntax :***

window.getComputedStyle(element);

window.getComputedStyle(element, pseudoElt);

We will get the left and top property values of dino and obstacle by using **window.getComputedStyle()**method and also parsing into integer,then saving in variables named such that **dx**(dino x-axis), **dy**(dino y-axis), **ox**(obstacle x-axis), and **oy(**obstacle y-axis) as shown below :

dx = parseInt(window.getComputedStyle(dino, null).getPropertyValue('left'));

dy = parseInt(window.getComputedStyle(dino, null).getPropertyValue('top'));

ox = parseInt(window.getComputedStyle(obstacle, null).getPropertyValue('left'));

oy = parseInt(window.getComputedStyle(obstacle, null).getPropertyValue('top'));

Now we will calculate the values of dx - ox and dy - oy using the **Math.abs()**function which returns the absolute value of a number. That is, it returns x  if x is positive or zero, and the negation of x if x is negative. And store them into variables named **offsetX**(dx - ox) for x-axis and **offsetY**(dy - oy) for y-axis.

We will create an if statement such as if the value of **offsetX**is less than 73 and **offsetY**is less than 52 then we will set the innerHTML of class **gameover** to "Game Over - Reload to Play Again" and will also remove the class **“obstacleAni”**from class **obstacle**to stop the animation of obstacle to over the game as shown below :

offsetX = Math.abs(dx - ox);

offsetY = Math.abs(dy - oy);

console.log(offsetX, offsetY)

if (offsetX < 73 && offsetY < 52) {

gameOver.innerHTML = "Game Over - Reload to Play Again"

obstacle.classList.remove('obstacleAni')

}

The entire code will  written inside the **setInterval()** method as shown below :

setInterval(() => {

dino = document.querySelector('.dino');

gameOver = document.querySelector('.gameOver');

obstacle = document.querySelector('.obstacle');

dx = parseInt(window.getComputedStyle(dino, null).getPropertyValue('left'));

dy = parseInt(window.getComputedStyle(dino, null).getPropertyValue('top'));

ox = parseInt(window.getComputedStyle(obstacle, null).getPropertyValue('left'));

oy = parseInt(window.getComputedStyle(obstacle, null).getPropertyValue('top'));

offsetX = Math.abs(dx - ox);

offsetY = Math.abs(dy - oy);

console.log(offsetX, offsetY)

if (offsetX < 73 && offsetY < 52) {

gameOver.innerHTML = "Game Over - Reload to Play Again"

obstacle.classList.remove('obstacleAni')

}

}, 10);

**MOTION OF DINO**

As earlier we have used the onkeydownevent of javascript to provide a code indicating which key is pressed. Here we will use the left and right arrow key to move the dino in the left and right direction. So the keycode of the right key is 39 and the keycode of the left key is 37.

* **To move right**

We will create an if statement such as if the keycode is equal to 39 it means if the right arrow key is press we will select the **“.dino”** class in variable named **dino**using querySelector() method and get the values of dino in direction left by using **window.getComputedStyle()**method and also parsing them into integer and saving them in variable named **dinoX.**Now it's time to add left property of css into dino class using JavaScript, For moving right we will add value of (**dinox**+ 112)px as shown below :

if (e.keyCode == 39) {

dino = document.querySelector('.dino');

dinoX = parseInt(window.getComputedStyle(dino, null).getPropertyValue('left'));

dino.style.left = dinoX + 112 + "px";

}

* **To move left**

We will create an if statement such as if the keycode is equal to 37 it means if the left arrow key is press we will select the **“.dino”** class in variable named **dino**using **querySelector()** method, and get the value of left property by using **window.getComputedStyle()**method and also parsing them into integer and saving them in variable named **dinoX**as we have done above**.**Now it's time to add left property of css into dino class using JavaScript but for moving left, here we will add value of (**dinox**- 112)px as shown below :

if (e.keyCode == 37) {

dino = document.querySelector('.dino');

dinoX = parseInt(window.getComputedStyle(dino, null).getPropertyValue('left'));

dino.style.left = (dinoX - 112) + "px";

}

This both code will be written inside the function which we have created for up arrow key as shown below :

document.onkeydown = function (e) {

console.log("Key code is: ", e.keyCode)

if (e.keyCode == 38) {

dino = document.querySelector('.dino');

dino.classList.add('animateDino');

setTimeout(() => {

dino.classList.remove('animateDino')

}, 700);

}

if (e.keyCode == 39) {

dino = document.querySelector('.dino');

dinoX = parseInt(window.getComputedStyle(dino, null).getPropertyValue('left'));

dino.style.left = dinoX + 112 + "px";

}

if (e.keyCode == 37) {

dino = document.querySelector('.dino');

dinoX = parseInt(window.getComputedStyle(dino, null).getPropertyValue('left'));

dino.style.left = (dinoX - 112) + "px";

}

}

* **Logic of score and increasing speed of obstacle**

First, we will create two variables named **score**and **cross**. The value of the score is 0 and the value of the cross will be True(boolean).

We will create an **else if** statement below the if statement which we had created for gameover in **setInterval()**method at an interval of 10 milliseconds.

The condition of the else if statement will be such as if offsetX is less then 145 and cross is true. we will use assignment operator in **score**for adding 1 and **cross**will be false every time when **else if** is executed. Also, we will use the **setTimeout()**method for the cross variable to change its value to true after 1000 milliseconds and we will also call a function named **updateScore** with a parameter score. This function **updateScore**will be created further.

else if (offsetX < 145 && cross) {

score += 1;

updateScore(score);

cross = false;

setTimeout(() => {

cross = true;

}, 1000);

}

We will create a function named **updateScore**which will take parameter **score.**The work of this function will be to update scores such as to set the innerHTML of an id named **scoreCount**using parameter **score.**

function updateScore(score) {

scoreCount.innerHTML = "Your Score: " + score

}

Now we will write the code to increase the obstacle speed with the score. The

**setTimeout()** method will be used with the timer of 500 milliseconds so that no lag will be seen while the dino is crossing the obstacle. Inside the setTimeout() method first we will get the value of property **‘animation-duration’**and parse the value in float and store in the variable named **aniDur**, then create a new variable named **newDur**for storing value of (**aniDur** - 0.1) it means animation of obstacle will be decrease by 0.1s. Adding style by JavaScript in class obstacle, set property animation duration to newDur and also we have to add **s**(seconds)behind it as shown below :

setTimeout(() => {

aniDur = parseFloat(window.getComputedStyle(obstacle, null).getPropertyValue('animation-duration'));

newDur = aniDur - 0.1;

obstacle.style.animationDuration = newDur + 's';

console.log('New animation duration: ', newDur)

}, 500);

This setTimeout() method will be inside the else if block as shown below :

else if (offsetX < 145 && cross) {

score += 1;

updateScore(score);

cross = false;

setTimeout(() => {

cross = true;

}, 1000);

setTimeout(() => {

aniDur = parseFloat(window.getComputedStyle(obstacle, null).getPropertyValue('animation-duration'));

newDur = aniDur - 0.1;

obstacle.style.animationDuration = newDur + 's';

console.log('New animation duration: ', newDur)

}, 500);

}

}, 10);

* **Styling gameOver AND scoreCount**

We will target the class name **gameOver**to add some CSS propertiessuch as position, top, font-size, text-align, and font-family as shown below :

.gameOver{

position: relative;

top:63px;

font-size: 53px;

text-align: center;

font-family: 'Ubuntu', sans-serif;

}

Now for score we will target the id named **scoreCount**and will add some CSS properties like font-size, border, padding as shown below :

#scoreCount{

font-size: 25px;

color: #54212f;

font-weight: bold;

position: absolute;

right: 45px;

top: 31px;

border: 2px solid black;

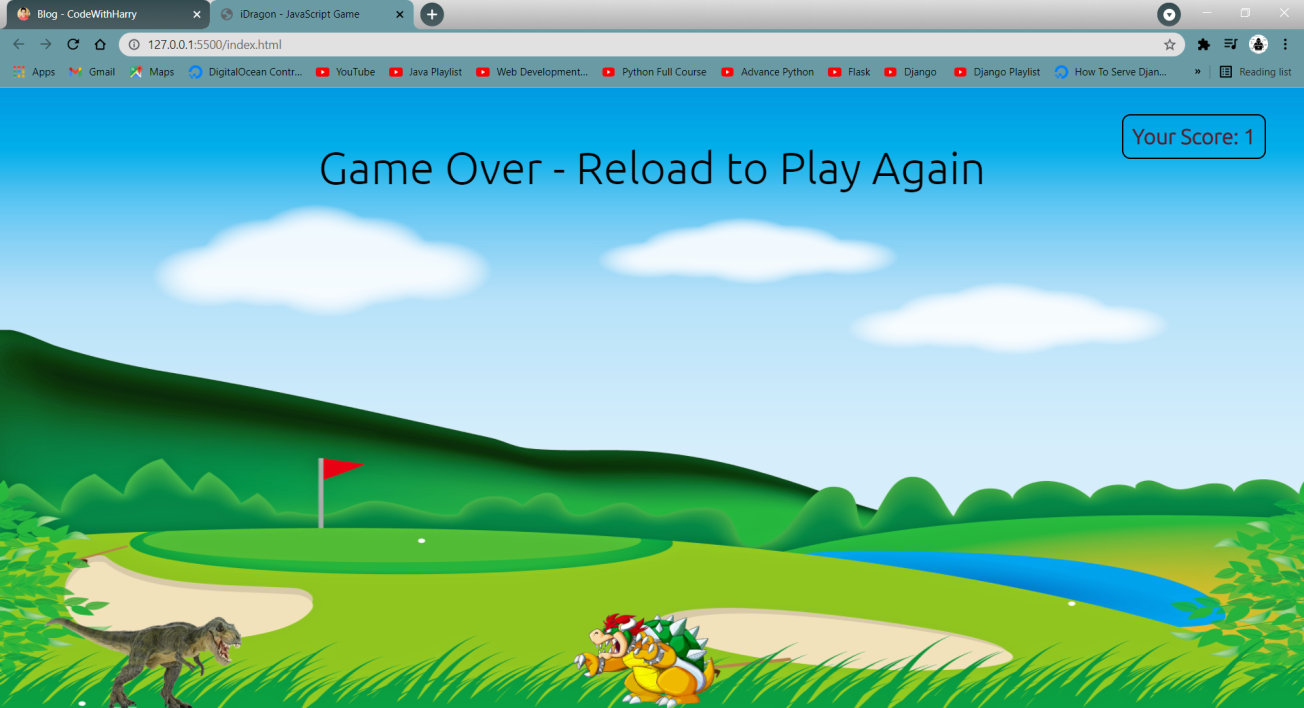
padding: 10px;

font-family: 'Ubuntu', sans-serif;

border-radius: 10px;

}

**Webpage will look as shown below :**

****

* **Adding sounds in game**

We will create an object of audio by giving parameter of file name as string and will store them in variables named such as **audio**(music) and **audiogo**(gameover sound) as shown below :

audio = new Audio('music.mp3');

audiogo = new Audio('gameover.mp3');

We use the **play()** method to start playing the current audio.

***Syntax***:

audioObject.play()

We will play audio inside the **setTimeout()** method to play the sound after 1000 milliseconds by using **play()** method.

setTimeout(() => {

audio.play()

}, 1000);

Previously we have made the if statement for game over, In that block we will play **audiogo**using **play()** method. Then we will use the pause() method to pause the audiogo and audio after 1000 milliseconds.

if (offsetX < 73 && offsetY < 52) {

gameOver.innerHTML = "Game Over - Reload to Play Again"

obstacle.classList.remove('obstacleAni')

audiogo.play();

setTimeout(() => {

audiogo.pause();

audio.pause();

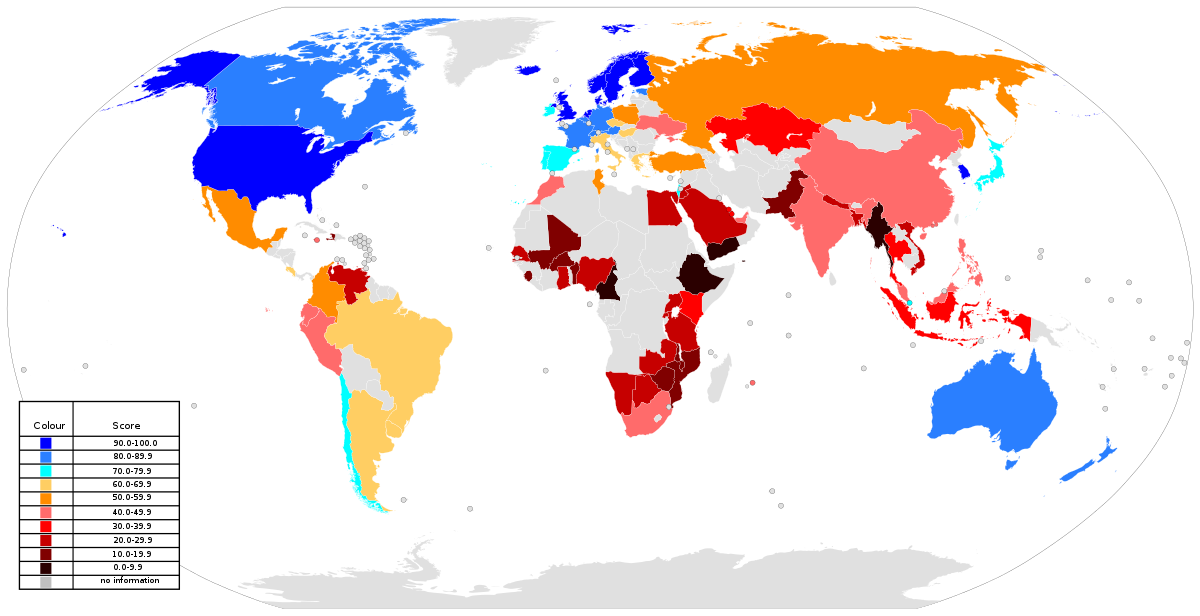
}, 1000);

}

CHAPTER 1

**INTRODUCTION TO WEB**

**1.1 INTRODUCTION**



A global map of the Web index for countries in 2014

The World Wide Web (WWW), also called the Web, is an information space where documents and other web resources are identified by Uniform Resource Locators (URLs), interlinked by hypertext links, and accessible via the Internet. English scientist Tim Berners-Lee invented the World Wide Web in 1989. He wrote the first web browser in 1990 while employed at CERN in Switzerland.The browser was released outside CERN in 1991, first to other research institutions starting in January 1991 and to the general public on the Internet in August 1991.

**1.2 HISTORY OF WEB**

Sir Tim Berners-Lee is a British computer scientist. He was born in London, and his parents were early computer scientists, working on one of the earliest computers. Growing up, Sir Tim was interested in trains and had a model railway in his bedroom .

He recalls : “I made some electronic gadgets to control the trains. Then I ended up getting more interested in electronics than trains . Later , on when I was in college I made a computer out of an old television set " .

After graduating from Oxford University, Berners-Lee became a software engineer at CERN, the large particle physics laboratory near Geneva, Switzerland. Scientists come from all over the world to use its accelerators, but Sir Tim noticed that they were having difficulty sharing information.

“In those days, there was different information on different computers, but you had to log on to different computers to get at it. Also, sometimes you had to learn a different program on each computer. Often it was just easier to go and ask people when they were having coffee…”, Tim says.

Tim thought he saw a way to solve this problem – one that he could see could also have much broader applications. Already, millions of computers were being connected together through the fast-developing internet and Berners-Lee realised they could share information by exploiting an emerging technology called hypertext.

In March 1989, Tim laid out his vision for what would become the web in a document called “Information Management: A Proposal”. Believe it or not, Tim’s initial proposal was not immediately accepted. In fact, his boss at the time, Mike Sendall, noted the words “Vague but exciting” on the cover.

The web was never an official CERN project, but Mike managed to give Tim time to work on it in September 1990. He began work using a NeXT computer, one of Steve Jobs’ early products. By October of 1990, Tim had written the three fundamental technologies that remain the foundation of today’s web (and which you may have seen appear on parts of your web browser ,

**• HTML :** HyperText Markup Language. The markup (formatting) language for the web.

**• URI :** Uniform Resource Identifier. A kind of “address” that is unique and used to identify to each resource on the web. It is also commonly called a URL.

**• HTTP** : Hypertext Transfer Protocol. Allows for the retrieval of linked resources from across the web. Tim also wrote the first web page editor/browser (“WorldWideWeb.app”) and the first web server (“httpd“). By the end of 1990, the first web page was served on the open internet, and in 1991, people outside of CERN were invited to join this new web community.

**1.2.1 Web browser**

A web browser (commonly referred to as a browser) is a software application for accessing information on the World Wide Web. Each individual web page, image, and video is identified by a distinct URL, enabling browsers to retrieve and display them on the user's device.

A web browser is not the same thing as a search engine, though the two are often confused. For a user, a search engine is just a website, such as google.com, that stores searchable data about other websites. But in order to connect to and display websites on their device, a user needs to have a web browser installed. The most popular web browsers are Chrome, Firefox, Safari, Internet Explorer, and Edge.

**1.2.2 Website**

A website is a collection of related web pages, including multimedia content, typically identified with a common domain name, and published on at least one web server. Notable examples are wikipedia.org, google.com, and amazon.com. Today roughly 380 new websites are created every minute across the World. A website may be accessible via public InternetProtocol (IP) network, such as the Internet, or a private local area network (LAN), by referencing a uniform resource locator (URL) that identifies the site.

**1.2.3 Webpage**

A web page or webpage is a document commonly written in HyperText Markup Language (HTML) that is accessible through the Internet or other network using an Internet browser. A web page is accessed by entering a URL address and may contain text, graphics, and hyperlinks to other web pages and files.

**1.3 CLIENT SERVER ARCHITECTURE**

Client-server architecture, architecture of a computer network in which many clients (remote processors) request and receive service from a centralized server (host computer).

Client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns. Servers wait for requests to arrive from clients and then respond to them.

Ideally, a server provides a standardized transparent interface to clients so that clients need not be aware of the specifics of the system (i.e., the hardware and software) that is providing the service.

Clients are often situated at workstations or on personal computers, while servers are located elsewhere on the network , usually on more powerful machines.

This computing model is especially effective when clients and the server each have distinct tasks that they routinely perform.

In hospital data processing, for example, a client computer can be running an application program for entering patient information while the server computer is running another program that manages the database in which the information is permanently stored.

Many clients can access the server’s information simultaneously, and, at the same time, a client computer can perform other tasks, such as sending e-mail. Because both client and server computers are considered intelligent devices, the client-server model is completely different from the old “mainframe” model, in which a centralized mainframe computer performed all the tasks for its associated "dumb" terminals .

**CHAPTER 2**

**HTML**

**2.1 INTRODUCTION TO HTML**

HTML is the standard markup language for creating Web pages.

• HTML stands for Hyper Text Markup Language

• HTML describes the structure of Web pages using markup

• HTML elements are the building blocks of HTML pages

• HTML elements are represented by tags

• HTML tags label pieces of content such as "heading", "paragraph", "table", and so on

• Browsers do not display the HTML tags, but use them to render the content of the page

**2.2 HISTORY OF HTML**

The history of hypertext markup language is a strange and interesting tale. From its simple start as an online subset of SGML through political maneuverings of the huge browser companies to its current piecemeal – but growing – compatibility, the language has weathered a storm of growth, abuse, and innovation. The idea behind HTML was a modest one. When Tim Berners-Lee was putting together his first elementary browsing and authoring system for the Web, he created a quick little hypertext language that would serve his purposes. He imagined dozens, or even hundreds, of hypertext formats in the future, and smart clients that could easily negotiate and translate documents from servers across the Net.

In 1993, a debate was exploding on the fledgling HTML mailing list, and finally a college student named Marc Andreessen added <img> to his Mosaic browser. People objected, saying it was too limited. They wanted <include>or <embed>, which would allow you to add any sort of media to a Web page with the much-touted content negotiation used on the client.

That was too big a project, according to Marc, and he need to ship ASAP. Mosaic went with <img>, and it would be years before including media in a page using <embed>, or <applet>, or <object> would come to the surface again .Something needed to give. If things kept up the way they were going, Netscape and Microsoft would eventually have two completely proprietary versions of HTML, but with no way of supporting the utopian vision of content negotiation.

**2.3 HTML TAGS**

HTML tags are element names surrounded by angle brackets :

<tagname>content goes here...</tagname>

• HTML tags normally come in pairs like <p> and </p>

• The first tag in a pair is the start tag, the second tag is the end tag

• The end tag is written like the start tag, but with a forward slash inserted before the tag name

**2.3.1 Web Browsers**

The purpose of a web browser (Chrome, IE, Firefox, Safari) is to read HTML documents and display them.

HTML Page Structure

Below is a visualization of an HTML page structure :

<html>

<head>

<title>Page title</title>

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph</p>

<p>This is another paragraph</p>

</body>

</html>

**Example.2.3.1**

<html>

<head>

<title>Page Title</title>

</head>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>

**Output :**

**My First Heading**

My first paragraph.

**Example Explained**

• The <html> element is the root element of an HTML page

• The <head> element contains meta information about the document

• The <title> element specifies a title for the document

• The <body> element contains the visible page content

• The <h1> element defines a large heading

• The <p> element defines a paragraph

**2.3.2 HTML Editors**

Write HTML Using Notepad or TextEdit

Web pages can be created and modified by using professional HTML editors.

However, for learning HTML we recommend a simple text editor like Notepad (PC) or TextEdit (Mac).

We believe using a simple text editor is a good way to learn HTML .

Follow the four steps below to create your first web page with Notepad or TextEdit.

**Step 1:** Open Notepad (PC)

**Windows 8 or later :**

Open the **Start Screen** (the window symbol at the bottom left on your screen)

Type **Notepad**

**Windows 7 or earlier :**

Open **Start** > **Programs** > **Accessories** > **Notepad**

**Step 2 :** Write Some HTML

Write or copy some HTML into Notepad .

<html>

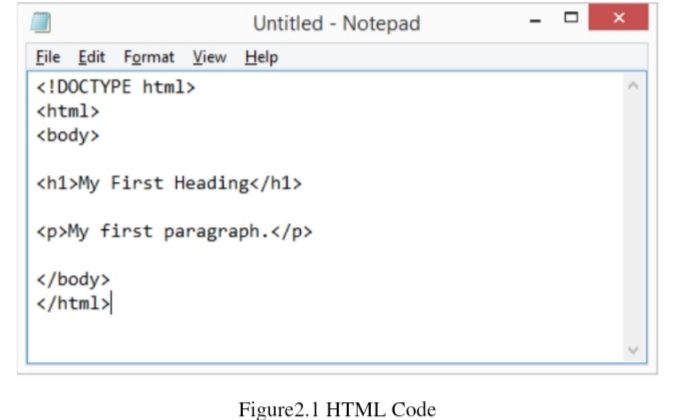
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

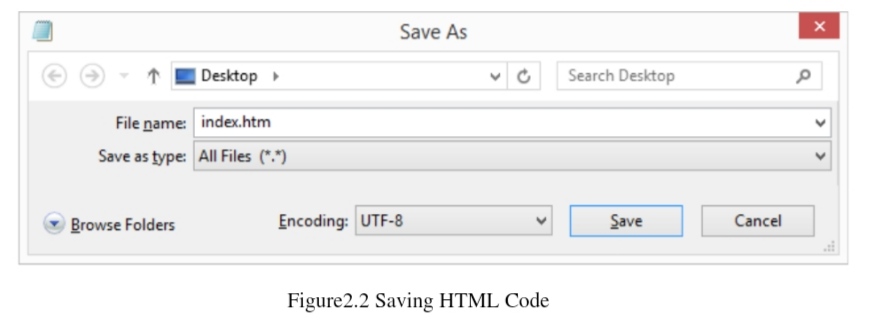
</html>



**Step 3 :** Save the HTML Page Save the file on your computer.

Select **File** > **Save as** in the Notepad menu

Name the file **"index.htm"** and set the encoding to UTF-8 ( which is the preferred encoding for HTML files )



Step 4 : View the HTML page in your browser

Open the saved HTML file in your favourite browser ( Double click on the file or right-click and choose "open with" ) .

The result will look much like this :



**2.3.3 HTML DOCUMENT**

The html document itself begins with <html> and ends with </html>

The visible part of the HTML document is between <body> and </body>

**Example : 2.3.2**

<html>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>

**Output :**

**My First Heading**

My first paragraph.

**2.3.4 HTML Headings**

HTML headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading.

<h6> defines the least important heading

**Example:2.3.3**

<h1>This is heading 1</h1>

<h2>This is heading 2</h2>

<h3>This is heading 3</h3>

**Output :**

This is heading 1

This is heading 2

This is heading 3

**2.3.5 HTML Paragraphs**

HTML paragraphs are defined with the <p> tag

**Example: 2.3.4**

<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

**2.3.6 HTML Links**

HTML links are defined with the <a> tag:

**Example:2.3.5**

<a href="https://www.w3schools.com">This is a link</a>

**2.3.7 HTML Images**

HTML images are defined with the <img> tag.

The source file (src), alternative text (alt), width, and height are provided as attributes:

**Example: 2.3.6**

<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">

**2.3.8 HTML Buttons**

HTML buttons are defined with the <button> tag:

**Example:2.3.7**

<button>Click me</button>

**2.3.9 HTML Lists**

HTML lists are defined with the <ul> (unordered/bullet list) or the <ol> (ordered/numbered list) tag, followed by <li> tags (list items)

**Example:2.3.8**

<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

**2.3.10 HTML Elements**

An HTML element usually consists of a start tag and end tag, with the content inserted in between:

<tagname>Content goes here...</tagname>

The **HTML** element is everything from the start tag to the end tag :

<p>My first paragraph.</p>

**2.4 HTML TABLES**

Defining an HTML Table

An HTML table is defined with the <table> tag.

Each table row is defined with the <tr> tag. A table header is defined with the <th> tag. By default, table headings are bold and centered. A table data/cell is defined with the <td> tag.

**Example:2.4.1**

<html>

<body>

<h2>Basic HTML Table</h2>

<table style="width:100%">

<tr>

<th>Firstname</th>

<th>Lastname</th>

<th>Age</th>

</tr>

<tr>

<td>Anu</td>

<td>Smith</td>

<td>23</td>

</tr>

<tr>

<td>John</td>

<td>Nick</td>

<td>34</td>

</tr>

<tr>

<td>John</td>

<td>Dov</td>

<td>40</td>

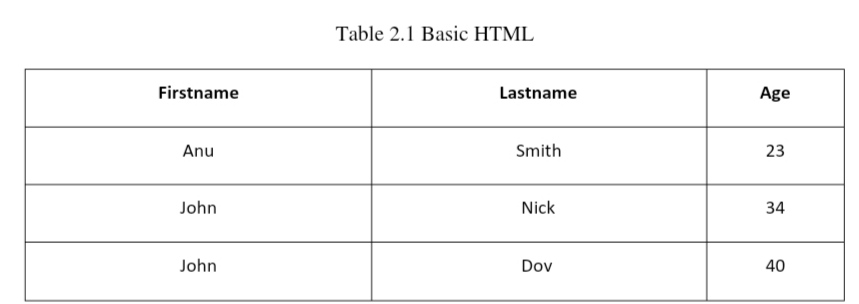
</tr>

</table>

</body>

</html>

Table 2.1 Basic HTML



**2.4.1 HTML Table - Adding a Border**

If you do not specify a border for the table, it will be displayed without borders.

A border is set using the CSS border property:

**Example:2.4.2**

table, th, td

{

border:

1px solid black;

}

**2.4.2 HTML Table - Collapsed Borders**

If you want the borders to collapse into one border, add the CSS border-collapse property:

**Example:2.4.3**

table, th, td

{

border:

1px solid black;

border-collapse: collapse;

}

**2.4.3 HTML Table - Adding Cell Padding**

Cell padding specifies the space between the cell content and its borders. If you do not specify a padding, the table cells will be displayed without padding. To set the padding, use the CSS padding property: Example:2.4.4

th, td {

padding: 15px;

}

**2.4.4 HTML Table - Left-align Headings**

By default, table headings are bold and centered. To left-align the table headings, use the CSS text-align property:

**Example:2.4.5**

th {

text-align: left;

}

**2.4.5 HTML Table - Adding Border Spacing**

Border spacing specifies the space between the cells. To set the border spacing for a table, use the CSS border-spacing property:

**Example:2.4.6**

table

{a

border-spacing:

}

**2.5 HTML FORMS**

The <form> Element

The HTML <form> element defines a form that is used to collect user input :

<form>

.

form elements

.

</form>

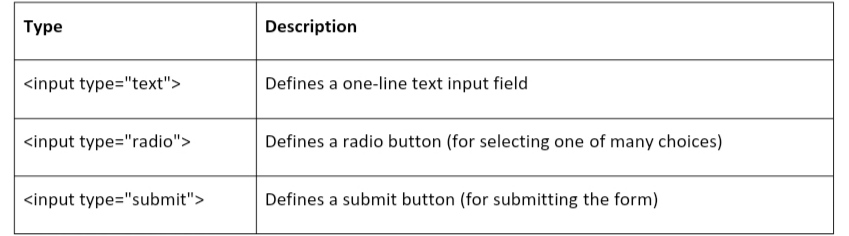
The <input> Element

The <input> element is the most important form element.

The <input> element can be displayed in several ways, depending on the **type** attribute.

Here are some examples

Table 2.2 HTML forms



**2.5.1 Text Input**

<input type="text"> defines a one-line input field for text input: Example:2.5.1

<form>

First name:<br>

<input type="text" name="firstname"><br>

Last name:<br>

<input type="text" name="lastname">

</form>

**Text Input**

First name:

Last name:

**2.5.2 Radio Button Input**

<input type="radio"> defines **a radio button.**

Radio buttons let a user select ONE of a limited number of choices:

**Example: 2.5.2**

<form>

<input type="radio" name="gender" value="male" checked> Male<br> <input type="radio" name="gender" value="female"> Female<br>

<input type="radio" name="gender" value="other"> Other

</form>

This is how the HTML code above will be displayed in a browser

* Male
* Female
* Other

**2.5.3 The Submit Button**

<input type="submit"> defines a button for **submitting** the form data to a **form-handler.**

The form-handler is specified in the form's **action** attribute: **Example:2.5.3**

<form action="/action\_page.php">

First name:<br>

<input type="text" name="firstname" value="Mickey"><br>

Last name:<br>

<input type="text" name="lastname" value="Mouse"><br><br>

<input type="submit" value="Submit">

</form>

**HTML FORMS**

First name :

Last name :

Submit

**CHAPTER 3**

**CSS**

**3.1 INTRODUCTION**

• CSS stands for Cascading Style Sheets

• CSS describes how HTML elements are to be displayed on screen, paper, or in other media

• CSS saves a lot of work. It can control the layout of multiple web pages all at once

• External stylesheets are stored in CSS files

**3.2 HISTORY OF CSS**

Before Cascading Style Sheets (CSS) there was very little that could be done to change the design of a web page. While Hyper Text Markup Language (HTML) creates documents for the World Wide Web, it was specifically designed to hold the content of a web page. Housed in a separate file, CSS adds the style and design to a web page. The term cascading comes from the ability to combine multiple CSS files to determine the style for one page.

As more people started using HTML, the demand grew for more design capabilities, which would allow developers to control how web documents looked. But browsers offered limited capabilities for styling.

In 1993 NCSA Mosaic was released, making the web more popular than ever, but it only offered limited capability to change fonts and colors.

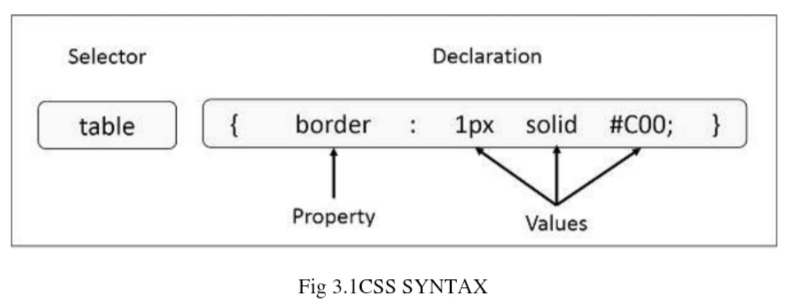
In October 1994, Tim Berners-Lee formed the World Wide Web Consortium, (W3C) at the Massachusetts Institute of Technology Laboratory for Computer Science , W3C has members that are government entities, businesses, educational institutions and individuals.

Although it took 3 years for any browser to come close to fully implementing CSS, August 1996 Microsoft Internet Explorer became the first browser to support CSS. Netscape followed suit in supporting CSS, but also implemented an alternative Javascript Style Sheets, which were never fully completed, and are now deprecated. To this day, there are differences in the way CSS is implemented in different browsers, leading developers to use hacks to make web pages look consistent in different browsers .

**Why Use CSS ?**

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

**3.3 CSS SYNTAX**



**Selector** − A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.

**Property** − A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.

**Value** − Values are assigned to properties. For example, color property can have value either red or #F1F1F1 etc.

**Example : 3.1**

You can define a table border as follows −

table{ border :1px solid #C00; }

**3.4 TYPES OF CSS**

There are three types of css-

* Internal style sheet
* External style sheet
* Inline style

**3.4.1 Internal Style Sheet**

An internal style sheet may be used if one single page has a unique style. Internal styles are defined within the <style> element, inside the <head> section of an HTML page

**Example :-**

<head>

<style>

body {

background-color: linen;

}

h1 {

color: maroon;

margin-left: 40px;

}

</style>

</head>

**3.4.2 External Style Sheet**

With an external style sheet, you can change the look of an entire website by changing just one file!

Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section <head>

<link rel="stylesheet" type="text/css" href="mystyle.css">

</head>

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

Here is how the "mystyle.css" looks

body {

background-color: lightblue;

}

h1 {

color: navy;

margin-left: 20px;

}

**3.4.3 Inline Styles**

An inline style may be used to apply a unique style for a single element. To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

The example below shows how to change the color and the left margin of a <h1> element:

<h1 style="color:blue;margin-left:30px;">This is a heading</h1>

**3.5 CSS COLORS**

CSS uses color values to specify a color. Typically, these are used to set a color either for the foreground of an element (i.e., its text) or else for the background of the element. They can also be used to affect the color of borders and other decorative effects.

**3.5.1 CSS colors-hex codes**

A hexadecimal is a 6 digit representation of a color.

The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB).

Each hexadecimal code will be preceded by a pound or hash sign '#'. Following are the examples to use Hexadecimal notation.

**Color Color HEX**

black #000000

red #FF0000

green #00FF00

blue #0000FF

yellow #FFFF00

white #FFFFFF

**3.5.2 CSS Background**

The CSS background properties are used to define the background effects for elements.

CSS background properties:

• background-color

• background-image

• background-repeat

• background-attachment

• background-position

**EXAMPLE**

body {

background-color: lightblue;

}

**3.5.3 CSS Borders**

css border properties:

The CSS border properties allow you to specify the style, width, and color of an element's border.

The border-style property specifies what kind of border to display.

The following values are allowed :-

* 1. **dotted -** Defines a dotted border
  2. **dashed -** Defines a dashed border
  3. **solid -** Defines a solid border
  4. double - Defines a double border
  5. **groove -** Defines a 3D grooved border. The effect depends on the border-color value
  6. **ridge -** Defines a 3D ridged border. The effect depends on the border-color value
  7. **inset -** Defines a 3D inset border. The effect depends on the border-colour value
  8. **outset** - Defines a 3D outset border. The effect depends on the border-color value
  9. **none -** Defines no border
  10. **hidden -** Defines a hidden border

**The border-style** property can have from one to four values (for the top border, right border, bottom border, and the left border)

**3.5.4 CSS TEXT**

• The color property is used to set the color of a text

• The direction property is used to set the text direction

• The text-align property is used to align the text of a document.

The following example demonstrates how to set the text color.

Possible value could be any color name in any valid format

**Example 3.5**

<html>

<head>

</head>

<body>

<p style = "color:red;">

This text will be written in red.

</p>

</body>

</html>

**3.5.5 Font Family**

The font family of a text is set with the font-family property

**EXAMPLE 3.6**

p {

font-family: "Times New Roman", Times,serif;

} sss

**3.5.6 CSS Tables**

**TABLE BORDER** To specify table borders in CSS, use the border property. SYNTAX

table, th, td {

border: 1px solid black;

}

**MOTIVATION :-**

Our motive of this web programming project is to help the beginners of HTML & CSS as well as help everyone to make their first website with only html and css , no need of any external libraries .

Our second motive is to help people who are interested in coding like ours website development

**FUTURE - SCOPE :-**

HTML itself worth very little but if we pair it with CSS , then they can be a Front-End Developer .

I hope that you use twitter , instagram , facebook , google all this site is made upon CSS & HTML in the Front-End .

**CONCLUSION**

In a nutshell, this Project has been an excellent and rewarding experience. I can conclude that there has been a lot I’ve learned from my mentor .

The technical aspects of my work are not flawless and could be improved provided enough time.

As someone with no prior experience with Node.js whatsoever, I believe my time spent in research and discovering it was well worth it and contributed to finding an acceptable solution to build a fully functional web service. Two main things that I’ve learned the importance of are time-management skills and self-motivation.

**REFERENCES**

1. https://en.wikipedia.org/wiki/Web\_development

2. https://www.udemy.com/courses/development/web-development/

3. https://www.edx.org/learn/web-development

4. https://www.codeconquest.com/what-is-coding/web-programming/