Practical 9

Aim: to display hello world with alert box
Objective:

- To create a simple Cordova mobile application that displays “Hello World” on the screen.
- To demonstrate the working of the deviceready event in Cordova, ensuring JavaScript functions run only after the device APIs are fully loaded.

Code:
html

```
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="Content-Security-Policy" content="default-src 'self' data: https://ssl.gstatic.com
'unsafe-eval'; style-src 'self' 'unsafe-inline'; media-src *; img-src 'self' data: content:;">
<meta name="format-detection" content="telephone=no">
<meta name="msapplication-tap-highlight" content="no">
<meta name="viewport" content="initial-scale=1, width=device-width, viewport-fit=cover">
<meta name="color-scheme" content="light dark">
<link rel="stylesheet" href="css/index.css">
<title>Hello World</title>
</head>
<body>
<div class="app">
<p> Hello World </p>
<p> Shiva Gupta  15 </p>
</div>
<script src="cordova.js"></script>
<script src="js/index.js"></script>
</body>
</html>
```

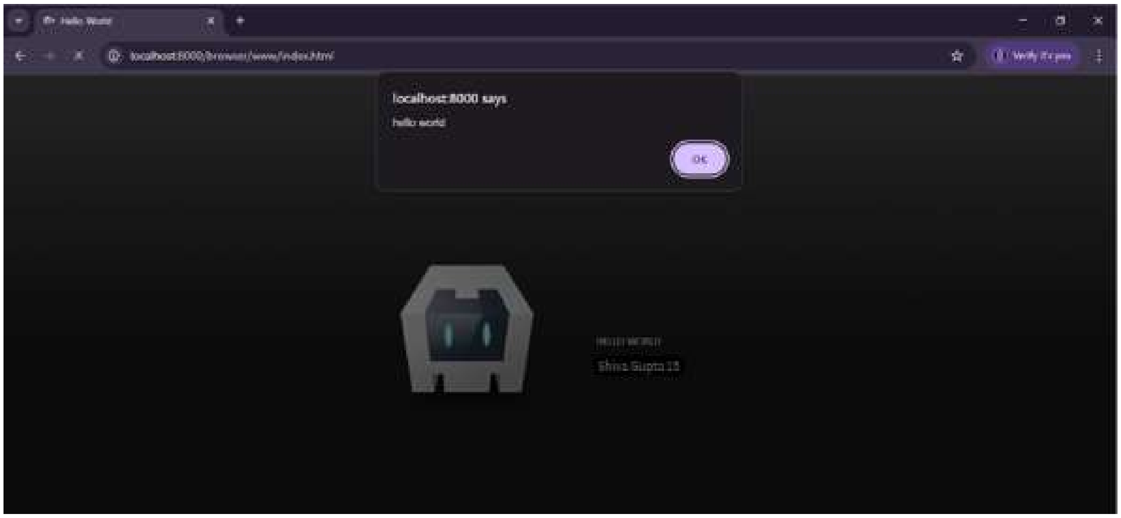
```
var app= {
initilize:function() {
document.addEventListener('deviceready', this.OnDeviceReady.bind(this),false);
},
OnDeviceReady:function(){
console.log('cordova running');
alert('hello world');
}
};
app.initilize();
```

Method:

- initialize() → Attaches the deviceready event listener.

- OnDeviceReady() → Executes when the device is ready, logs "cordova running" to console, and shows an alert message.

Conclusion:
The program confirms Cordova is running by logging to console and triggering an alert.
Output:



localhost:8000 says
hello world

HELLO WORLD
Shiva Gupta 15

Practical 10

Aim: to use json with css styles

Objective:

The objective of this project is to develop a simple Cordova application that demonstrates dynamic data handling. The app displays basic information on the screen and allows the user to load additional data from a local JSON file with the click of a button. It aims to show how Cordova integrates HTML, CSS, and JavaScript to build mobile applications, and how XMLHttpRequest can be used for fetching and displaying data dynamically inside the app interface.Code:

Index.js

```
function loadData()
{
var xhr=new XMLHttpRequest();
xhr.open("GET","data.json", true);
xhr.onreadystatechange=function(){
if(xhr.readyState===4 && xhr.status===200){
var res= JSON.parse(xhr.responseText);
document.getElementById("output".innerhtml=res.message);
}
};
xhr.send();
}
```

Style.css

```
body{
font-family: 'Courier New', Courier, monospace;
background: #f4f4f9;
text-align: center;
margin: 0;
padding: 0;
}
h1{
color: #333;
}
button{
background-color: #007bff;
border: none;
color: white;
padding: 10px 20px;
cursor: pointer;
font-size: 16px;
}
button:hover{
background-color: #0056b3;
}
#output{
```

```
margin-top: 20px;
font-size: 18px;
color: #444;
}
```

Data.json

```
{
"message":"I love KC College Data Loaded Successfully."
}
```

Index.html

```
<html>
<head>
<meta charset="utf-8">
<meta name="format-detection" content="telephone=no">
<meta name="msapplication-tap-highlight" content="no">
<meta name="viewport" content="initial-scale=1, width=device-width, viewport-fit=cover">
<meta name="color-scheme" content="light dark">
<link rel="stylesheet" href="css/index.css">
<link rel="stylesheet" href="css/style.css">
<title>Hello World</title>
</head>
<body>
<div class="app">
<p> Hello World </p>
<p> Anam Kazi 27 </p>
<button onclick="loadData()"> load data </button>
<div id="output">
<p> Fetch the data </p>
</div>
</div>
<script src="cordova.js"></script>
<script src="js/index.js"></script>
</body>
</html>
```

Config.xml

```
<?xml version='1.0' encoding='utf-8'?>
<widget id="com.example.helloworld1" version="1.0.0" xmlns="http://www.w3.org/ns/widgets"
xmlns:cdv="http://cordova.apache.org/ns/1.0">
<name>HelloWorldApp1</name>
<description>Sample Apache Cordova App</description>
<author email="dev@cordova.apache.org" href="https://cordova.apache.org">
Apache Cordova Team
</author>
<content src="index.html" />
```

```
<allow-intent href="http://*/*" />
<allow-intent href="https://*/*" />
<access origin="https://trusted.cdn.com"/>
</widget>
```

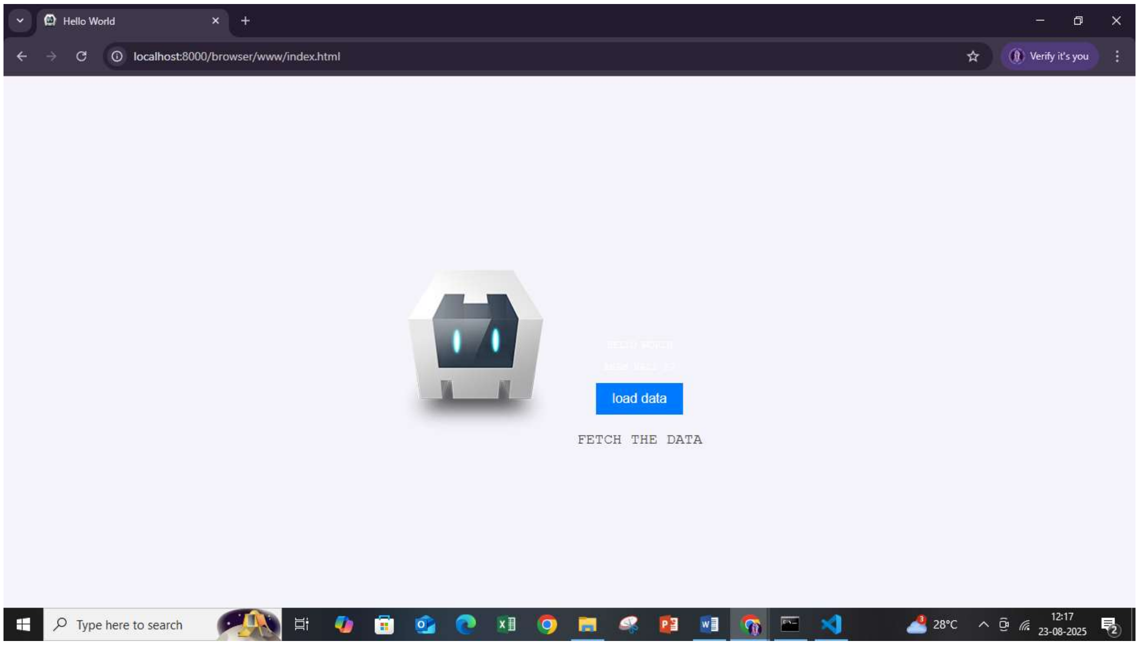
Method:

loadData() → Creates an XMLHttpRequest, fetches data.json, parses it as JSON, and updates the content of the output div with the message.

Conclusion:

The Cordova app successfully fetches and displays the JSON message "I love KC College Data Loaded Successfully." when the button is clicked, confirming proper integration of HTML, CSS, JavaScript, and Cordova.

Output:



FETCH THE DATA

Practical 11

Aim: to accept two numbers and add them up
Objective:
The objective of this project is to design and implement a simple web-based calculator that performs the addition of two numbers. The application provides an interactive interface where users can input numeric values, process them through JavaScript, and instantly view the result. This project demonstrates the integration of HTML for structure, CSS for styling, and JavaScript for functionality in building basic web applications.

Code:
Html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Simple Calculator</title>
<style>
body {
font-family: Arial, sans-serif;
background-color: #f0f2f5;
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
margin: 0;
flex-direction: column;
}
.container {
background-color: #fff;
padding: 2rem;
border-radius: 8px;
box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
text-align: center;
width: 300px;
}
h2 {
color: #333;
margin-bottom: 1.5rem;
}
form {
display: flex;
flex-direction: column;
align-items: center;
}
```

```
label {
margin-bottom: 0.5rem;
color: #555;
font-weight: bold;
}
input[type="number"] {
width: 100%;
padding: 0.75rem;
margin-bottom: 1rem;
border: 1px solid #ccc;
border-radius: 4px;
box-sizing: border-box;
}
button {
background-color: #007bff;
color: #fff;
border: none;
padding: 0.75rem 1.5rem;
border-radius: 4px;
cursor: pointer;
font-size: 1rem;
transition: background-color 0.3s ease;
}
button:hover {
background-color: #0056b3;
}
h3 {
margin-top: 2rem;
color: #333;
}
#result {
font-weight: bold;
color: #007bff;
font-size: 1.25rem;
}
</style>
</head>
<body>
<div class="container">
<h2>Add Two Numbers</h2>
<form id="calculatorForm">
<label for="num1">First Number:</label>
<input type="number" id="num1" required>
<br><br>
<label for="num2">Second Number:</label>
<input type="number" id="num2" required>
```

```
<br><br>
<button type="button" onclick="addNumbers()">Add</button>
</form>
<h3>Result: <span id="result"></span></h3>
</div>
<script>
function addNumbers() {
// Get the input values and convert them to numbers
const num1 = parseFloat(document.getElementById('num1').value);
const num2 = parseFloat(document.getElementById('num2').value);
// Check if the inputs are valid numbers
if (isNaN(num1) || isNaN(num2)) {
document.getElementById('result').textContent = 'Please enter valid numbers.';
return;
}
// Perform the addition
const sum = num1 + num2;
// Display the result in the 'result' span
document.getElementById('result').textContent = sum;
}
</script>
</body>
</html>
```

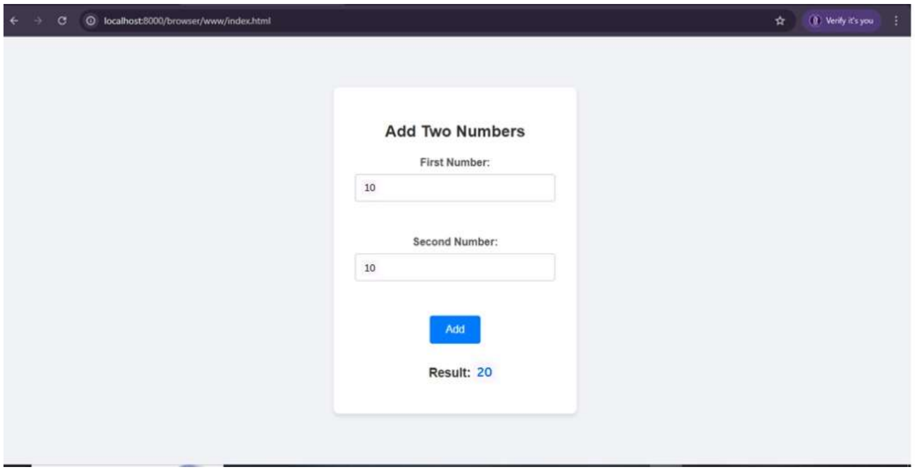
Method:

addNumbers() → Fetches two input values, validates them, converts them into numbers, calculates their sum, and displays the result dynamically on the webpage.

Conclusion:

The simple calculator successfully accepts two inputs, performs addition, and displays the result instantly, demonstrating the use of HTML, CSS, and JavaScript for creating interactive web applications.

Output:



Add Two Numbers

First Number:

10

Second Number:

10

Result: 20