HTML5 API refers to the various interfaces and technologies built into the HTML5 specification that allow web developers to interact with web browsers and their underlying technologies, such as the Document Object Model (DOM), Cascading Style Sheets (CSS), and JavaScript.

Types of HTML5 API :

1. Geolocation API :

The *Geolocation API* is a web-based interface provided by modern web browsers that enables websites or web applications to access the location information of a user's device. This API can provide latitude, longitude, altitude, and other related information about the device's location using various sources such as GPS, IP address, and cell tower triangulation. The Geolocation API is useful for location-based services such as maps, directions, and local search, as well as for enhancing the user experience of web applications with location-aware functionality.

Example :

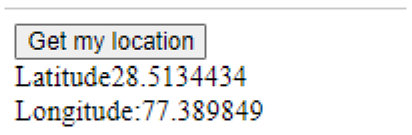
HTML

```html
<!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>
    <button onclick="getUserLocation()">Get my location</button>
    <div id="location"></div>
</body>
<script>
    function getUserLocation(){
        alert("hello")
        if(navigator.geolocation){
navigator.geolocation.getCurrentPosition(showPosition,showError)
        }
        else{
document.getElementById("location").innerHTML="Geolocation is not supported by this browser"
        }
    }
    function showPosition(position){
        document.getElementById("location").innerHTML="Latitude"
+position.coords.latitude+"<br>"+"Longitude:"+position.coords.longitude

    }
    function showError(error){
        console.log(error)
        document.getElementById("location").innerHTML="Error:"+
error.message


    }
</script>
</html>
```

Output :



```
Latitude28.5134434
Longitude:77.389849
```

Geolocation API

Explanation :

The code starts by defining an HTML button with an *onclick* event listener that will execute the *getUserLocation()* function when clicked. The *getUserLocation()* function first checks if the browser supports the Geolocation API using the *navigator.geolocation* object. If the API is supported, it calls the *getCurrentPosition()* method and passes two callback functions as arguments: *showPosition* and *showError*. The *showPosition()* function is called if the location is successfully retrieved, and it updates the *innerHTML* of the *location* element with the latitude and longitude values obtained from the *position* object.If there is an error retrieving the location, the *showError()* function is called, which logs the error to the console and updates the *location* element with the error message.When the user clicks the "Get my location" button, the browser will prompt the user to allow or deny access to their location information. If the user grants permission, the *getCurrentPosition()* method will attempt to retrieve the location information and call the appropriate callback function. If the user denies permission or if there is an error retrieving the location information, the *showError()* function will be called instead.

2. Drag and Drop API

The Drag and Drop API is a set of browser events and methods that enable web developers to create rich and interactive interfaces by allowing users to drag and drop elements within a web page or between different web pages. This API provides several event listeners that enable developers to track and respond to user actions during drag and drop, such as when a user starts dragging an item, moves it, or drops it. Developers can use this API to create features such as file uploads, sortable lists, and interactive games.

Example :

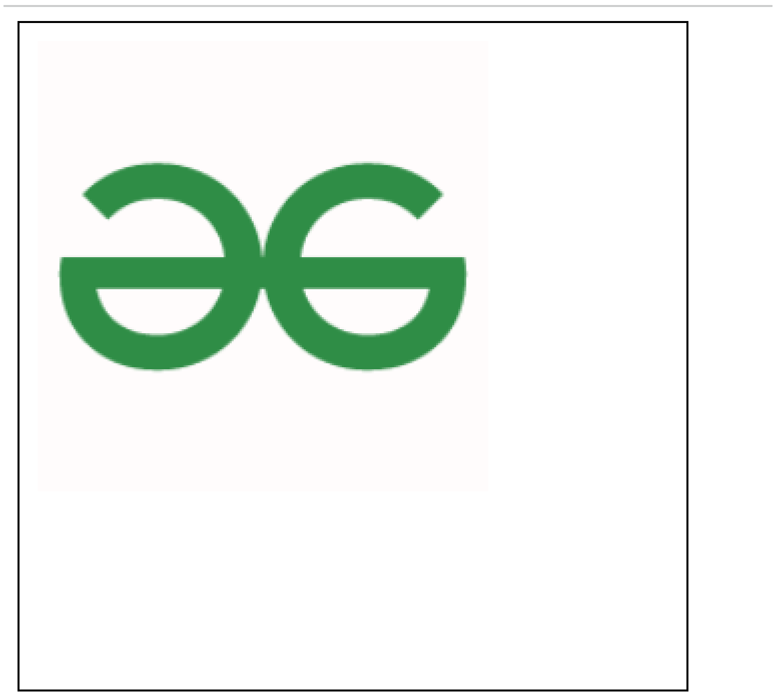
HTML
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
```

```
</head>
<style>
#container{
    width: 350px;
    height: 350px;
    padding: 10px;
    border: 1px solid black;
}
</style>
<body>
    <div id="container" ondragover="allowDrop(event)"
ondrop="drop(event)"></div>
    <br>
    <img id="myimage" src="./image.png" alt="image" draggable="true" width="250"
height="250" ondragstart="drag(event)">
</body>
<script>
    function allowDrop(e){
    e.preventDefault()
    }
    function drag(e){

        e.dataTransfer.setData("myimagetotransfer",e.target.id)
    }
    function drop(e){
        e.preventDefault();
        var data=e.dataTransfer.getData("myimagetotransfer")
        e.target.appendChild(document.getElementById(data))
    }
</script>
</html>
```

Output :

Drag And Drop API



Drag And Drop API

Explanation :

There is an image with the draggable attribute set to "true", which means it can be dragged by the user. When the user starts dragging the image, the drag event is fired, and the setData method is used to store the ID of the image being dragged. When the user drops the image onto the container div, the drop event is fired, and the getData method is used to retrieve the ID of the image. Finally, the appendChild method is used to add the image to the container div. The allowDrop function is used to allow the image to be dropped onto the container div by preventing the default behavior of the browser.

3. File API

The *File API* is a web-based JavaScript API that provides a way to access and manipulate files on a user's local computer. It allows web applications to read and write files, access metadata about the file, and even generate URLs that can be used to download files. The

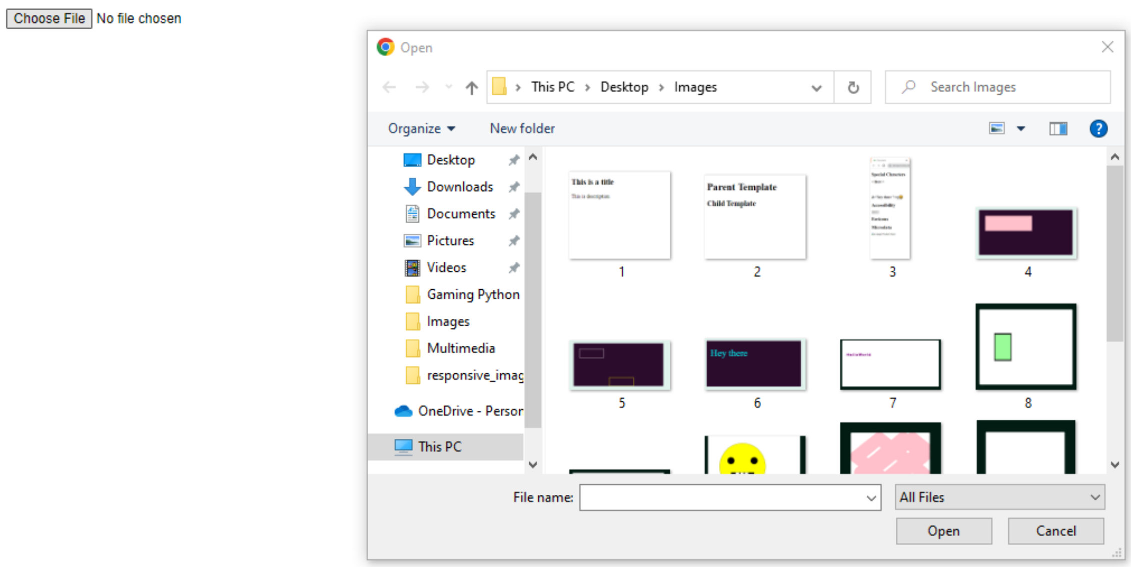
File API is commonly used in web-based file managers, photo galleries, and other web applications that require access to the user's local file system.

Example :

HTML

```
<!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>
    <input type="file" id="file" id="fileInput" onchange="displayFileData()"/>
    <div id="output"></div>
</body>
<script>
    function displayFileData(){
        var file=document.getElementById("file input")
        console.log(fileInput.files[0])
        var filedata=fileInput.files[0]
        var output=document.getElementById("output")
        output.innerHTML=
        "File Name : " +filedata.name+"<br>"+
        "File Type : " +filedata.type+"<br>"+
        "File Size : " +filedata.size
    }
</script>
</html>
```

Output :



File API

Explanation :

In the code, there is an HTML input element of type "file" that allows the user to select a file from their computer. When the user selects a file, the "onchange" event is triggered, which calls the "displayFileData()" function. This function retrieves the selected file using the "getElementById()" method and the "files" property. It then accesses the file properties such as "name", "type" and "size" using the "File" object.Finally, the file properties are displayed in a "div" element with the id "output". The "innerHTML" property of the "div" element is set to a string that includes the file name, type, and size information.

4. Web Storage API

Allows web applications to store data locally on a user's device.

5. Web Workers API

Allows web applications to run scripts in the background without interrupting the user interface.

6. WebSockets API

Allows real-time communication between web browsers and servers.

7. Canvas API

Allows for dynamic, scriptable rendering of 2D shapes and bitmap images.

8. Audio and Video API

Allows web applications to play and manipulate audio and video files.

9. WebRTC API

Enables real-time communication between web browsers, including video chat and file sharing.