HTML5 APIs in Detail

1. Geolocation API

The Geolocation API allows web applications to access the geographical location of a user. This is useful for location-based services, maps, and navigation apps.

Example Usage:

```javascript
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(showPosition, showError);
}

function showPosition(position) {
  console.log("Latitude: " + position.coords.latitude +
  " Longitude: " + position.coords.longitude);
}

function showError(error) {
  switch(error.code) {
    case error.PERMISSION_DENIED:
      console.log("User denied the request for Geolocation.");
      break;
    case error.POSITION_UNAVAILABLE:
      console.log("Location information is unavailable.");
      break;
    case error.TIMEOUT:
      console.log("The request to get user location timed out.");
      break;
    case error.UNKNOWN_ERROR:
      console.log("An unknown error occurred.");
      break;
  }
}
```

2. Web Storage API

The Web Storage API provides mechanisms by which browsers can store key-value pairs, locally within the client's browser. This is more secure and faster than using cookies.

Local Storage Example:

```javascript
// Set item
localStorage.setItem("username", "JohnDoe");

// Get item
var username = localStorage.getItem("username");
console.log(username); // JohnDoe

// Remove item
localStorage.removeItem("username");
```

Session Storage Example:

```javascript
// Set item
sessionStorage.setItem("sessionID", "12345");

// Get item
var sessionID = sessionStorage.getItem("sessionID");
console.log(sessionID); // 12345

// Remove item
sessionStorage.removeItem("sessionID");
```

3. Canvas API

The Canvas API provides a means for drawing graphics via JavaScript and the HTML <canvas> element. It can be used for rendering graphs, game graphics, or other visual images on the fly.

Example Usage:

```html
<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#000000;"></canvas>
<script>
  var canvas = document.getElementById("myCanvas");
  var ctx = canvas.getContext("2d");
  ctx.fillStyle = "#FF0000";
  ctx.fillRect(0, 0, 150, 75);
</script>
```

4. WebSockets API

The WebSockets API provides a way to open a persistent connection between the client and the server. This enables real-time communication, which is essential for applications like chat, online gaming, and live data feeds.

Example Usage:

```javascript
var socket = new WebSocket("ws://www.example.com/socketserver");

socket.onopen = function() {
  console.log("Connection opened");
  socket.send("Hello Server!");
};

socket.onmessage = function(event) {
  console.log("Message from server: " + event.data);
};

socket.onclose = function() {
  console.log("Connection closed");
};

socket.onerror = function(error) {
  console.log("Error: " + error.message);
};
```

5. File API

The File API allows web applications to access files and file systems from the user's device. This is particularly useful for handling file uploads and processing files locally within the browser.

Example Usage:

```html
<input type="file" id="fileInput">
<script>
  document.getElementById('fileInput').addEventListener('change', function(event) {
    var file = event.target.files[0];
    var reader = new FileReader();

    reader.onload = function(e) {
      console.log(e.target.result);
```

```
    };

    reader.readAsText(file);
  });
</script>
```

6. Drag and Drop API

The Drag and Drop API provides a way to implement drag-and-drop functionality in web applications, allowing users to move elements visually across the browser window.

Example Usage:

```html
<div id="drag1" draggable="true" ondragstart="drag(event)" style="width: 100px; height: 100px; background: #f00;"></div>
<div id="dropzone" ondrop="drop(event)" ondragover="allowDrop(event)" style="width: 200px; height: 200px; background: #0f0;"></div>

<script>
  function allowDrop(event) {
    event.preventDefault();
  }

  function drag(event) {
    event.dataTransfer.setData("text", event.target.id);
  }

  function drop(event) {
    event.preventDefault();
    var data = event.dataTransfer.getData("text");
    event.target.appendChild(document.getElementById(data));
  }
</script>
```

7. IndexedDB API

IndexedDB is a low-level API for client-side storage of significant amounts of structured data, including files and blobs. This API uses indexes to enable high-performance searches of this data.

Example Usage:

```javascript
var request = indexedDB.open("myDatabase", 1);

request.onerror = function(event) {
  console.log("Database error: " + event.target.errorCode);
};

request.onsuccess = function(event) {
  var db = event.target.result;
  var transaction = db.transaction(["myObjectStore"], "readwrite");
  var objectStore = transaction.objectStore("myObjectStore");

  var request = objectStore.add({ id: 1, name: "John Doe" });
  request.onsuccess = function(event) {
    console.log("Data added to the object store!");
  };
};

request.onupgradeneeded = function(event) {
  var db = event.target.result;
  var objectStore = db.createObjectStore("myObjectStore", { keyPath: "id" });
};
```