JAVASCRIPT

WEB API'S

JAVASCRIPT

Web **Speech API**

```
const textToSpeak = "Hello There";
const utterance =
    new SpeechSynthesisUtterance(textToSpeak);
window.speechSynthesis.speak(utterance);
```


“Hello There” Spoken out

JAVASCRIPT

Web **Storage API**

(localStorage)

```
// Store data
localStorage.setItem('key', 'value');
// Retrieve data
const storedValue = localStorage.getItem('key');
// Remove data
localStorage.removeItem('key');
```


Persistent storage in browser memory

JAVASCRIPT

Web **Storage API**

(sessionStorage)

```
// Store data
sessionStorage.setItem('key', 'value');
// Retrieve data
const storedValue = sessionStorage.getItem('key');
// Remove data
sessionStorage.removeItem('key');
```


Session-specific storage

JAVASCRIPT

Web **Fetch API**

```
fetch('http://example.com/movies.json')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('error:', error));
```


Fetch API to make a GET call

JAVASCRIPT

Web **Geolocation API**

```
navigator.geolocation.getCurrentPosition(
    position => {
      console.log(`Latitude: ${position.coords.latitude},
                   Longitude: ${position.coords.longitude}`
                  );
    },
    error => {
      console.error("Error getting location:", error.message);
    }
);
```

Get user's location.

JAVASCRIPT

Web **Canvas API**

```
const canvas = document.createElement('canvas');
canvas.width = 200;
canvas.height = 100;
document.body.appendChild(canvas);
const ctx = canvas.getContext('2d');
ctx.fillStyle = 'blue';
ctx.fillRect(10, 10, 180, 80);
```


To draw a filled rectangle in blue

JAVASCRIPT

Web **Audio API**

```
const audioContext = new (window.AudioContext ||
                                  window.webkitAudioContext)();
const oscillator = audioContext.createOscillator();
oscillator.frequency.setValueAtTime(440, audioContext.currentTime);
oscillator.connect(audioContext.destination);
oscillator.start();
oscillator.stop(audioContext.currentTime + 1);
```


An oscillator and play a tone

JAVASCRIPT

Web **Sockets API**

```
const socket = new WebSocket('wss://example.com/socket');
socket.addEventListener('open',
          () => socket.send('Hello, server!'));
socket.addEventListener('message',
          event => console.log('Received:', event.data));
socket.addEventListener('close',
          () => console.log('Connection closed.'));
```


Real-time, bidirectional communication protocol.

JAVASCRIPT

web **IndexedDB API**

```js
// Open (or create) the database
const dbName = "InstagramPostsDB";
const dbVersion = 1;

const request = indexedDB.open(dbName, dbVersion);
//handles errors that may occur during the database opening process.
request.onerror(() => {});
//specifies the actions to be taken when the database structure is being upgraded.
request.onupgradeneeded(() => {});
//defines the actions to be taken upon successful opening of the database.
request.onsuccess(() => {});
```


Structured client-side data storage.

JAVASCRIPT

web **File API**

```html
<input type="file" id="imageInput" accept="image/*">
<button onclick="uploadPost()">Upload Post</button>
<script>
  function uploadPost() {
    const file = document.getElementById('imageInput').files[0];
    console.log('Selected file:', file);
  }
</script>
```


Manipulate files, access metadata.

JAVASCRIPT

web **Notification API**

```
Notification.requestPermission()
    .then( permission => {
      new Notification('Hello, World!');
    });
```


Display system notifications.

JAVASCRIPT

web **Workers API**

```
const worker = new Worker('worker.js');
worker.postMessage('Hello from main script!');
```


Execute background scripts.

JAVASCRIPT

web **Intersection Observer API**

```
const observer = new IntersectionObserver(entries =>
    entries.forEach(
        entry => entry.isIntersecting &&
        console.log('Element is in the viewport!')
        )
    );
observer.observe(document.getElementById('yourElementId'));
```


Efficiently observes element visibility changes.

JAVASCRIPT

web **Mutation Observer API**

```
const observer = new MutationObserver(mutations =>
    mutations.forEach(mutation =>
        console.log('DOM change detected:', mutation)
        )
    );
const targetNode = document.getElementById('yourElementId');
const config = { attributes: true, childList: true, subtree: true };
observer.observe(targetNode, config); // Start observing DOM changes.
```


Observes DOM changes asynchronously.

JAVASCRIPT

web **Pointer Lock API**

```
const element = document.getElementById('yourElementId');
element.requestPointerLock();
```


Captures mouse movements precisely in-browser.

JAVASCRIPT

web **Battery Status API**

```
navigator.getBattery().then(battery => {
  console.log('Battery Level:', battery.level * 100 + '%');
  console.log('Charging:', battery.charging ? 'Yes' : 'No');
});
```


Monitors device battery information asynchronously.

JAVASCRIPT

web **Gamepad API**

```
window.addEventListener("gamepadconnected", (event) =>
    console.log("Gamepad connected:", event.gamepad.id)
);
window.addEventListener("gamepaddisconnected", (event) =>
    console.log("Gamepad disconnected:", event.gamepad.id)
);
```


Interacts with game controller devices.

JAVASCRIPT

web DeviceOrientation and Motion API

```
window.addEventListener("deviceorientation", (event) => {
  console.log("Device Orientation:", event.alpha,
      event.beta, event.gamma);
});

window.addEventListener("devicemotion", (event) => {
  console.log("Device Motion:", event.acceleration.x,
        event.acceleration.y, event.acceleration.z);
});
```

Tracks device orientation and motion data.

JAVASCRIPT

web **Push API**

```
// Check for Push API support
if ('PushManager' in window) {
  // Request notification permission
  Notification.requestPermission().then(permission => {
    if (permission === 'granted') {
      // Subscription logic goes here
    }
  });
}
```

Enables push notifications in browsers.

JAVASCRIPT

web **Payment Request API**

```
const supportedInstruments = [{ supportedMethods: 'basic-card' }];
const paymentDetails = { total:
    { label: 'Total', amount: { currency: 'USD', value: '10.00' } } };
const paymentPromise = new PaymentRequest(supportedInstruments, paymentDetails);
paymentPromise.show().then(paymentResponse =>
      paymentResponse.complete('success')
  );
```


Facilitates streamlined online payment processing in browsers.