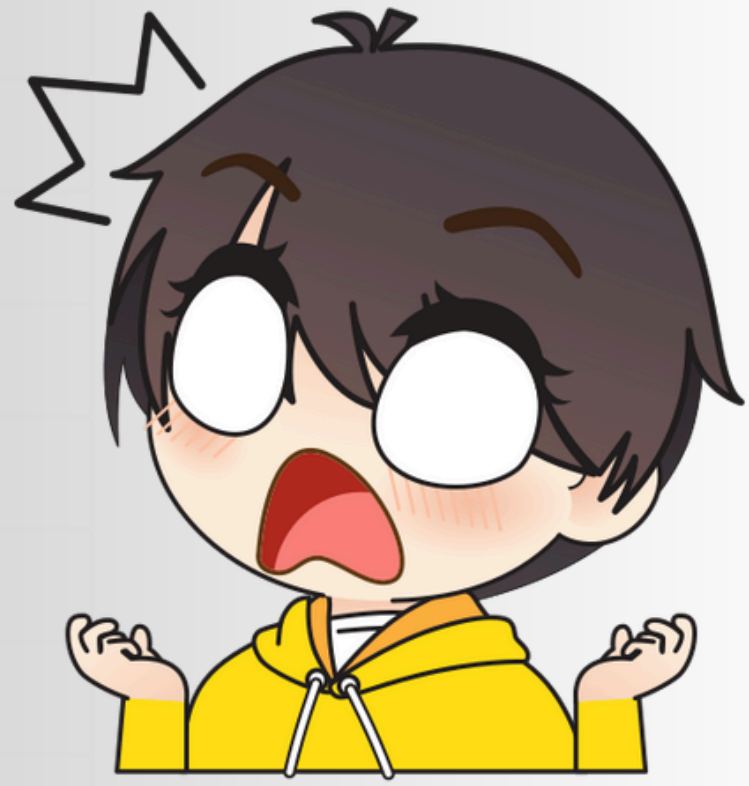
1000 APIs

FOR FRONT-END DEVELOPMENT

PART - 1



@frontend_in_depth

WEB-SPEECH API

Speech recognition is the automatic process of converting audio of human speech into text.

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

WEB STORAGE API

The Web Storage API provides mechanisms for webpages to store string-only key/value pairs.

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

WEB STORAGE API

The Web Storage API provides mechanisms for webpages to store string-only key/value pairs

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

Web Fetch API

The Fetch API provides an interface for fetching resources (including across the network).

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

WEB AUDIO API

The Web Audio API provides a powerful and versatile system for controlling audio on the Web, allowing developers to choose audio sources, add effects to audio, create audio visualizations, apply spatial effects (such as panning) and much more.

```
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);
```

WEB SOCKETS API

The WebSocket API makes it possible to open a two-way interactive communication session between the user's browser and a server. With this API, you can send messages to a server and receive responses without having to poll the server for a reply.

```
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.'));
```

WEB INDEXDB API

IndexedDB is a low-level API for client-side storage of significant amounts of structured data, including files/blobs. This API uses indexes to enable high-performance searches of this data. While Web Storage is useful for storing smaller amounts of data, it is less useful for storing larger amounts of structured data.

```
// 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(() => {});
```

WEB FILE API

The File API enables web applications to access files and their contents.

```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>
```

WEB NOTIFICATION API

The Notification interface of the Notifications API is used to configure and display desktop notifications to the user.

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

WEB WORKERS API

Web Workers makes it possible to run a script operation in a background thread separate from the main execution thread of a web application.

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

WEB INTERSECTION OBSERVER API

The Intersection Observer API provides a way to asynchronously observe changes in the intersection of a target element with an ancestor element or with a top-level document's viewport.

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

WEB MUTATION OBSERVER API

The MutationObserver interface provides the ability to watch for changes being made to the DOM tree. It is designed as a replacement for the older Mutation Events feature, which was part of the DOM3 Events specification.

```
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.
```

WEB POINTER LOCK API

The Pointer Lock API (formerly called Mouse Lock API) provides input methods based on the movement of the mouse over time (i.e., deltas), not just the absolute position of the mouse cursor in the viewport. It gives you access to raw mouse movement, locks the target of mouse events to a single element

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

WEB BATTERY STATUS API

The Battery Status API, more often referred to as the Battery API, provides information about the system's battery charge level and lets you be notified

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

GAMEPAD API

The Gamepad API is a way for developers to access and respond to signals from gamepads and other game controllers in a simple, consistent way.

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

ORIENTATION AND MOTION API

Mobile devices commonly have sensors such as gyroscopes, compasses, and accelerometers that can enable applications running on the device to detect the device's orientation and motion.

The device orientation events enable you to write web applications that can change their behavior based on the orientation of the user's device, and that can react when the user moves their device.

```
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);
});
```

PUSH API

The Push API gives web applications the ability to receive messages pushed to them from a server, whether or not the web app is in the foreground, or even currently loaded, on a user agent

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

WEB PAYMENT REQUEST API

The Payment Request API provides a consistent user experience for merchants and users. It is not a new way of paying for things; instead, it's a way for users to select their preferred way of paying for things and make that information available to a merchant.

```
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')
  );
```