DEVELOPERS
<SOCIETY/>

WEB APIs YOU NEED TO KNOW!

★ *Super Helpful..*

Network Information API

Vibrate API

Bluetooth API

Clipboard Async API

Fullscreen API

Battery Status API

Online Offline API

Geolocation API

Learn all in one place

DEVELOPERS <SOCIETY/>

Tap here to Join our WhatsApp Channel

NETWORK INFORMATION API

The Network Information API provides information about the network types ***(e.g., 'wifi', 'cellular', etc.)***

Observing user's connection

```
navigator
  .connection
  .effectiveType;
```

Returns the ***effective type*** of the connection which could be "4g", "3g", "2g", or "slow-2g".

```
navigator
  .connection
  .downlink;
```

Returns the ***effective bandwidth*** estimated in megabits per second.

@mgechev

JavaScript

```
console.log(navigator.connection);
```

👉 Network Type: 3g
👉 Round Trip Time(rtt): 300
👉 Bandwidth estimate(in MBPS): 1.45
👉 Max Bandwidth estimate(in MBPS): Infinity
👉 Save data enabled: false
👉 Device Connection Type: unknown

DEVELOPERS <SOCIETY/>

Tap here to Join our WhatsApp Channel

FULLSCREEN API

The **Fullscreen API** provides methods to present a specific Element in a **full-screen mode.**

```
<button onclick="activateFullscreen(document.documentElement);">
  Go fullscreen!
</button>

<button onclick="deactivateFullscreen();">
  Leave fullscreen
</button>
```

```
function activateFullscreen(element) {
  if(element.requestFullscreen) {
    element.requestFullscreen();
  }
};

function deactivateFullscreen() {
  if(document.exitFullscreen) {
    document.exitFullscreen();
  }
};
```

DEVELOPERS <SOCIETY/>

Tap here to Join our WhatsApp Channel

VIBRATE API

Use this to make your website cool

```
window.navigator.vibrate(300); // vibrate for 300ms
```

The **Navigator.vibrate()** method pulses the vibration hardware on the device, if such hardware exists.

DEVELOPERS <SOCIETY/>

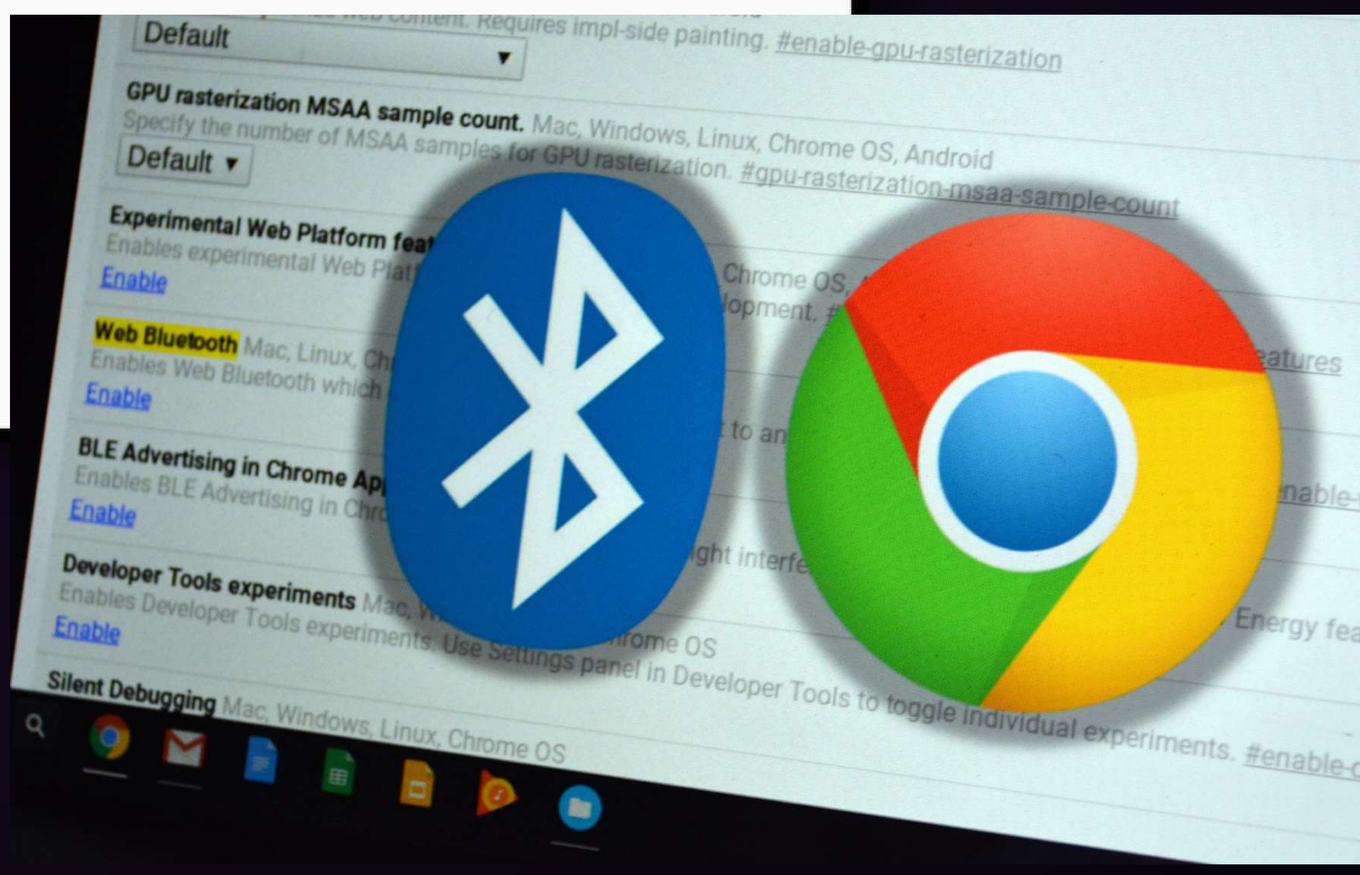
Tap here to Join our WhatsApp Channel

BLUETOOTH API

This **web API** allows you to connect to the **bluetooth** devices.

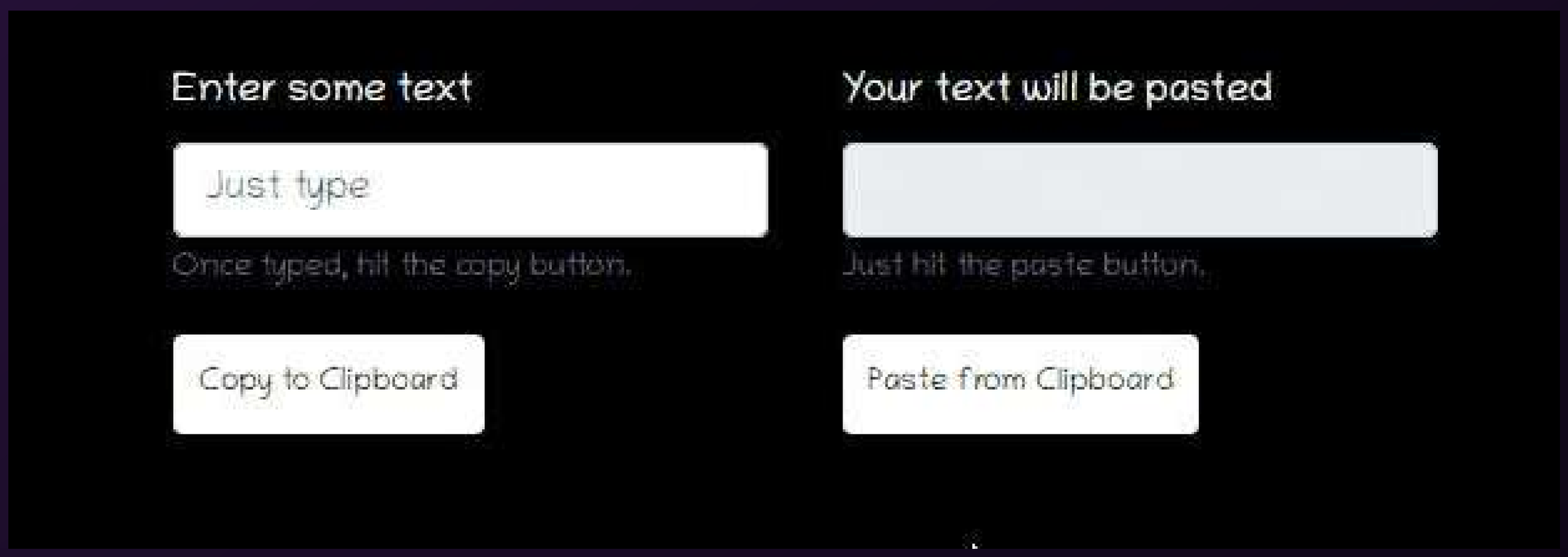
JavaScript

```
navigator.bluetooth.requestDevice({
    acceptAllDevices: true
}).then(device => {
    setDeviceName(device.name);
    setDeviceId(device.id)
    setDeviceConnected(device.connected);
}).catch(err => {
    console.log(err);
    setError(true);
})
```



CLIPBOARD ASYNC API

The **Clipboard API** provides the ability to respond to ***copy, cut and paste.***




```
async function performPaste(event) {
    event.preventDefault();
    try {
        const text = await navigator.clipboard.readText();
        setPastetext(text);
        console.log('Pasted content: ', text);
    } catch (err) {
       console.error('Failed to read clipboard contents: ', err);
    }
}
```

DEVELOPERS <SOCIETY/>

Tap here to Join our WhatsApp Channel

ONLINE OFFLINE API

Checks if the user is online or offline

```
if (navigator.onLine) {
  console.log('online');
} else {
  console.log('offline');
}

//To check if you are
online or offline
```

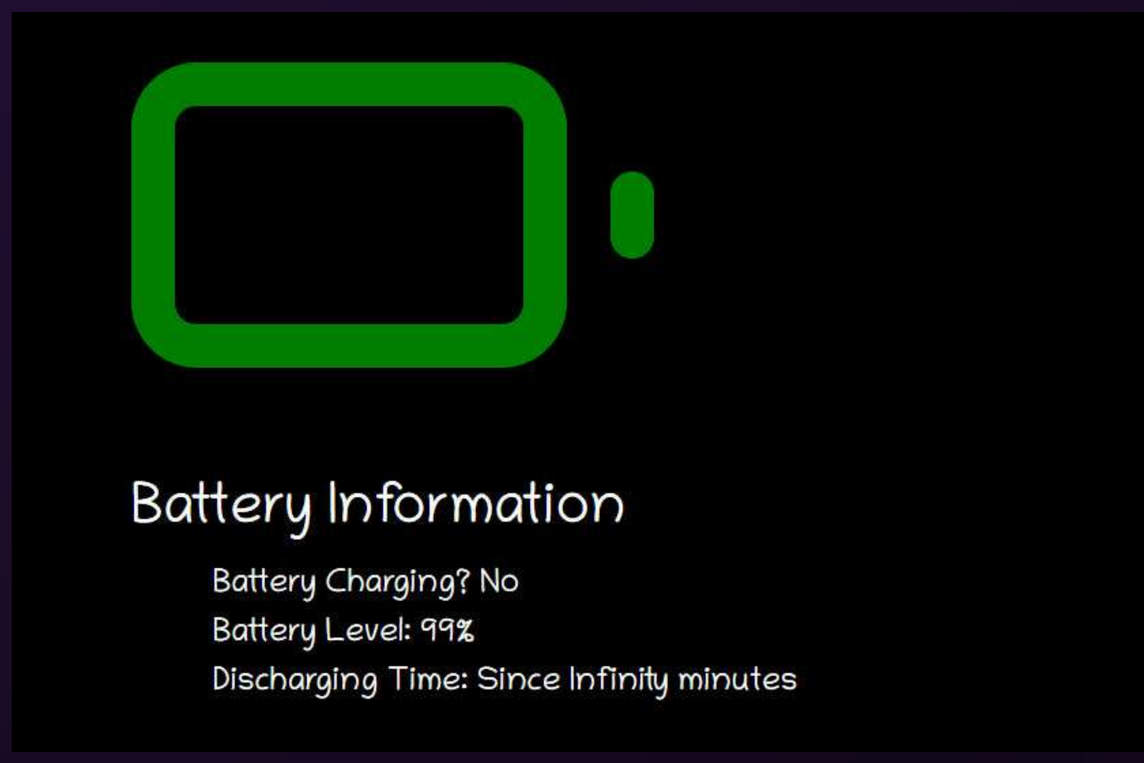
If the browser doesn't support **navigator.onLine** the above example will always come out as ***false/undefined.***

DEVELOPERS <SOCIETY/>

Tap here to Join our WhatsApp Channel

BATTERY STATUS API

This API helps in knowing all the information like, the ***battery is charging or not,*** how much charge is left




```
navigator.getBattery().then(function (battery) {

   // handle the charging change event
   battery.addEventListener("chargingchange", function () {
      console.log("Battery charging? " + (battery.charging ? "Yes" : "No"));
   });

   // handle charge level change
   battery.addEventListener("levelchange", function () {
      console.log("Battery level: " + battery.level * 100 + "%");
   });

   // handle charging time change
   battery.addEventListener("chargingtimechange", function () {
      console.log( "Battery charging time: " + battery.chargingTime + " seconds");
   });

   // handle discharging time change
   battery.addEventListener("dischargingtimechange", function () {
      console.log("Battery discharging time: " + battery.dischargingTime + " seconds");
   });

});
```

DEVELOPERS <SOCIETY/>

Tap here to Join our WhatsApp Channel

GEOLOCATION API

Use this to get the users location

```
navigator.geolocation.getCurrentPosition(callbackfunction)
```

Checks if geolocation is available

```
if (navigator.geolocation) {
   navigator.geolocation.getCurrentPosition(function(position){
                  console.log(position);
      });
}
```

The **geolocation API** allows javascript or web content to access the user's location or device's location.

DID YOU LEARN SOMETHING NEW?

FOLLOW US ON INSTAGRAM

Tap Here

@Developers_Society

Job Updates | Interview Preperations

JOIN OUR WhatsAPP right now, you won't find such post *anywhere!*

Tap here to Join our WhatsApp Channel

GIT CHEAT SHEET

CREATE

Clone an existing repository
```
$ git done ssh://user@domain.com/repo.git
```

Create a new local repository
```
$ git init
```

LOCAL CHANGES

Changed files in your working directory
```
$ git status
```

Changes to tracked files
```
$ git diff
```

Add all current changes to the next commit
```
$ git add.
```

Add some changes in <file> to the next commit
```
$ git add -p <file>
```

Commit all local changes in tracked files
```
$ git commit -a
```

Commit previously staged changes
```
$ git commit
```

Change the last commit
Don't amend published commits!
```
$ git commit --amend
```

COMMIT HISTORY

Show all commits, starting with newest
```
$ git log
```

Show changes over time for a specific file
```
$ git log -p <file>
```

Who changed what and when in <file>
```
$ git blame <file>
```

BRANCHES & TAGS

List all existing branches
```
$ git branch -av
```

Switch HEAD branch
```
$ git checkout <branch>
```

Create a new branch based on your current HEAD
```
$ git branch <new-branch>
```

Create a new tracking branch based on a remote branch
```
$ git checkout --track <remote/branch>
```

Delete a local branch
```
$ git branch -d <branch>
```

Mark the current commit with a tag
```
$ git tag <tag-name>
```

UPDATE & PUBLISH

List all currently configured remotes
```
$ git remote -v
```

Show information about a remote
```
$ git remote show <remote>
```

Add new remote repository, named <remote>
```
$ git remote add <shortname> <url>
```

Download all changes from <remote>, but don't integrate into HEAD
```
$ git fetch <remote>
```

Download changes and directly merge/integrate into HEAD
```
$ git pull <remote> <branch>
```

Publish local changes on a remote
Delete a branch on the remote
```
$ git branch -dr <remote/branch>
```

Publish your tags
```
$ git push --tags
```

MERGE & REBASE

Merge <branch> into your current HEAD
```
$ git merge <branch>
```

Rebase your current HEAD onto <branch>
Don't rebase published commits!
```
$ git rebase <branch>
```

Abort a rebase
```
$ git rebase --abort
```

Continue a rebase after resolving conflicts
```
$ git rebase --continue
```

Use your configured merge tool to solve conflicts
```
$ git mergetool
```

Use your editor to manually solve conflicts and (after resolving) mark file as resolved
```
$ git add <resolved-file>
```
```
$ git rm <resolved-file>
```

UNDO

Discard all local changes in your working directory
```
$ git reset -hard HEAD
```

Discard local changes in a specific file
```
$ git checkout HEAD <file>
```

Revert a commit (by producing a new commit with contrary changes)
```
$ git revert <commit>
```

Reset your HEAD pointer to a previous commit ... and discard all changes since then
```
$ git reset -hard <commit>
```

... and preserve all changes as unstaged changes
```
$ git reset <commit>
```

.... and preserve uncommitted local changes
```
$ git reset -keep <commit>
```