Promise

```
function getData() {
    return new Promis e(( res olve, reject) => {
        set Tim eout(() => {
            res olv e("O btained data");
        }, 1000);
    });
}
getData()
    .th en(data => consol e.l og( data)) // Output:
Obtained data (after 1 second)
    .ca tch (error => consol e.e rro r(e rror));
```

Objects that represent the future result of an asynchronous operation.

module

```
// file: module.js
export function greet( name) {
          con sol He.llloog( ${name}!);
}
// file: main.js
import { greet } from './mod ule.js';
greet( 'Bob'); // Output: Hello, Bob!
```

import and export to share code between files.
They allow you to organize your code in a modular and reusable way.

SessionStorage

```
//SAVING
const sessio nToken = 'abcde f12 345';
sessio nSt ora ge.s et Ite m(' ses sio nTo ken',
sessio nTo ken);
consol e.l og( 'Se ssion token saved to sessio -
nSt ora ge.');
//RETR IEVING
const stored Token = sessio nSt ora ge.g et Ite -
m(' ses sio nTo ken');
if (store dToken) {
    con sol e.l og( 'Se ssion token retrieved from
sessio nSt ora ge:', stored Token);
```

SessionStorage (cont)

```
> } else {
  console.log('No session token found.');
}
//SAVING and RETRIEVING a temporary object
const currentOrderStatus = {
  orderId: 123,
  status: 'pending'
};
sessionStorage.setItem('orderStatus', JSON.stringify(currentOrder-
Status));
console.log('Order status saved to sessionStorage.');
const storedOrderStatus = sessionStorage.getItem('orderStatus');
if (storedOrderStatus) {
  const parsedStatus = JSON.parse(storedOrderStatus);
  console.log('Order status retrieved from sessionStorage:', parsed-
Status);
  console.log('Current status:', parsedStatus.status);
}
//Removing a specific session item
sessionStorage.removeItem('sessionToken');
console.log('Session token removed from sessionStorage.');
//Clearing all session items
sessionStorage.clear();
console.log('All items cleared from sessionStorage.');
```

1. Allows you to store data in the browser.
2. Data persists only for the current browser session.
3. Exists only within the current browser tab.
4. The data is stored as key-value pairs, and both the key and the value must be strings.

async/await

```
async function getDataAsync() {
   try {
        const data = await getData(); //getD ata()
is a Promise
        con sol e.l og( data); // Output: Obtained
data (after 1 second)
   } catch (error) {
        con sol e.e rro r(e rror);
   }
}
getDat aAs ync();
```

More readable syntax for working with Promises.
async indicates that a function returns a Promise.
await suspends execution until a Promise is completed.

LocalStorage

```
const username = "JohnDoe";
localS tor age.se tIt em( 'us ern ame', username);
consol e.l og( 'Us ername saved to localS tor -
age.');
const stored Use rname = localS tor age.ge tIt -
em( 'us ern ame');
if (store dUs ername) {
    con sol e.l og( 'Us ername retrieved from
localS torage: ', stored Use rname);
} else {
    con sol e.l og('No username found in localS -
tor age.');
}
//SAVING OBJECT requires stringify
const userSe ttings = {
    theme: 'dark',
    fon tSize: '16px',
    not ifi cat ion sEn abled: true
};
localS tor age.se tIt em( 'us erS ett ings',
JSON.s tri ngi fy( use rSe tti ngs));
consol e.l og( 'User settings saved to localS tor -
age.');
//RETR IEVING OBJECT needs to be parsed
```

LocalStorage (cont)

```
> const storedSettings = localStorage.getItem('userSettings');
if (storedSettings) {
  const parsedSettings = JSON.parse(storedSettings);
  console.log('User settings retrieved from localStorage: ', parsedSet-
tings);
  console.log('Current theme: ', parsedSettings.theme);
} else {
  console.log('No user settings found in localStorage.');
}
//Removing a specific local item
localStorage.removeItem('username');
console.log('User after removal:', localStorage.getItem('theme')); //
Output: User after removal: null
//Clearing all local items
localStorage.clear();
console.log('Items in localStorage after clear:', localStorage); //
Output: Storage {length: 0}
```

Allows you to store data in the browser.
Data persists even after closing the browser.
Shared between all windows (or tabs) with the same origin
The data is stored as key-value pairs, and both the key and the value must be strings.