

Promise

```
function getData() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve("Obtained data");
    }, 1000);
  });
}

getData()
  .then(data => console.log(data)) // Output:
  .catch(error => console.error(error));
```

Objects that represent the future result of an asynchronous operation.

module

```
// file: module.js
export function greet(name) {
  console.log(`Hello, ${name}!`);
}

// file: main.js
import { greet } from './module.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 sessionToken = 'abcdef12345';
sessionStorage.setItem('sessionToken', sessionToken);
console.log('Session token saved to sessionStorage.');
```

```
//RETRIEVING
const storedToken = sessionStorage.getItem('sessionToken');
if (storedToken) {
  console.log('Session token retrieved from sessionStorage:', storedToken);
}
```

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(currentOrderStatus));
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:", parsedStatus);
  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.



By Techelopment
(techelopment)

Published 13th April, 2025.
Last updated 13th April, 2025.
Page 1 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

async/await

```
async function getDataAsync() {
  try {
    const data = await getData(); //getData()
    //getData() is a Promise
    console.log(data); // Output: Obtained
    //data (after 1 second)
  } catch (error) {
    console.error(error);
  }
}

getDataAsync();
```

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";
localStorage.setItem('username', username);
console.log('Username saved to localStorage');

const storedUsername = localStorage.getItem('username');
if (storedUsername) {
  console.log('Username retrieved from localStorage: ', storedUsername);
} else {
  console.log('No username found in localStorage');
}

//SAVING OBJECT requires stringify
const userSettings = {
  theme: 'dark',
  fontSize: '16px',
  notificationEnabled: true
};

localStorage.setItem('userSettings', JSON.stringify(userSettings));
console.log('User settings saved to localStorage');

//RETRIEVING 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: ', parsedSettings);
  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.



By **Techelopment**
 (techelopment)

Published 13th April, 2025.
 Last updated 13th April, 2025.
 Page 2 of 2.

Sponsored by **Readable.com**
 Measure your website readability!
<https://readable.com>