Modules in JavaScript

Definition: Modules in JavaScript are a way to encapsulate and organize code into reusable, self-contained units. They allow you to break down your code into smaller, more manageable pieces, each with its own scope. Modules can export functions, objects, or values from one module so that they can be imported and used in another module.

Key Features:

- **Encapsulation:** Keeps code organized and reduces global namespace pollution by encapsulating code within its own scope.
- **Reusability:** Allows code to be reused across different parts of an application or even different applications.
- **Maintainability:** Makes it easier to manage and maintain code by breaking it down into smaller, more manageable pieces.

ES Modules (ES6+):

- **Syntax:**
  - **Exporting:** export, export default
  - **Importing:** import

Usage in HTML:

To use ES modules in a browser, you need to include the type="module" attribute in the <script> tag.

```
<!-- index.html -->
<script type="module" src="index.js"></script>
```

Exporting and Importing in ES Modules:

Named Exports:

Export multiple values from a module. These values must be imported using the same names.

```
export let name = "Hema";

export let greet =()=>{
   console.log(`Hi, welcome to module`)
}
 let accountNumber = 123545685;
// export let accountHolderName = "Mahesh";
 let name = "Mahesh";


 let balanceCheck =()=>{
   console.log("Balance Checking...")
```

```
}

export {accountNumber, name, balanceCheck}
```

```
// import {name} from './src/App.js';
// import {greet} from './src/App.js'

import {name,greet} from './src/App.js';

// import{name as call,accountNumber,balanceCheck} from './src/Account.js'
import * as account from './src/Account.js'


console.log(name);
greet()

// console.log(call)
console.log(account.name)

console.log(account.accountNumber);
account.balanceCheck()
```

Default Exports:

Export a single value from a module. This value can be imported using any name.

Exporting:

```
export default function multiply(a, b) {
   return a * b;
}
```

Importing:

```
import multiply from './math.js';
console.log(multiply(2, 3)); // Output: 6
```

Cookies in JavaScript

Definition: Cookies are small pieces of data stored on the user's browser by websites. They are used to remember information about the user, such as login status, preferences, and tracking information, across different sessions or page visits.

Key Features:

- **Storage:** Cookies store data as name-value pairs.
- **Expiration:** Cookies can have an expiration date, after which they are automatically deleted.
- **Scope:** Cookies can be scoped to a specific domain and path.
- **Accessibility:** Cookies can be accessed by both the client-side (JavaScript) and the server-side.

Creating and Setting Cookies:

You can create and set cookies in JavaScript using the document.cookie property. Each cookie is a string in the format name=value, followed by optional attributes such as expires, path, domain, secure, and SameSite.

```
document.cookie = "username=Mahesh; expires=Fri, 31 Dec 2021 23:59:59 GMT; path=/";
```

 username=JohnDoe: The name-value pair of the cookie.
 expires: The expiration date of the cookie. If not specified, the cookie is a session cookie and will be deleted when the browser is closed.
 path=/: The path attribute specifies the URL path the cookie belongs to. If not specified, it defaults to the current path of the current document location.

Reading Cookies:

You can read cookies in JavaScript using the document.cookie property. It returns all cookies in a single string, with each cookie separated by a semicolon and space.

Example: Reading Cookies:

```
console.log(document.cookie);
```

Limitations and Considerations:

- **Size Limit:** Cookies have a size limit of about 4KB each.
- **Number of Cookies:** Browsers typically limit the number of cookies to around 20-50 per domain.
- **Security:** Cookies can be a security risk if not handled properly. Sensitive data should not be stored in cookies, and the secure and SameSite attributes should be used to enhance security.

Local Storage in JavaScript

Definition:

Local storage is a type of web storage that allows you to store data in the browser persistently. Data stored in local storage remains until explicitly deleted and is not sent to the server with each HTTP request.

Key Features:

- **Persistence:** Data persists even after the browser is closed and reopened.
- **Scope:** Data is scoped to the domain.
- **Capacity:** Typically around 5MB per domain.

Setting Data:

```
localStorage.setItem('username', 'Mahesh');
```

Reading Data:

```
const username = localStorage.getItem('username');
console.log(username);
```

Removing Data:

```
localStorage.removeItem('username');
```

Clearing All Data:

```
localStorage.clear();
```

Session Storage in JavaScript

Definition:

Session storage is a type of web storage that allows you to store data in the browser for the duration of the page session. Data stored in session storage is cleared when the page session ends (i.e., when the browser tab or window is closed).

Key Features:

- **Persistence:** Data persists only for the duration of the page session.
- **Scope:** Data is scoped to the domain and the window/tab.
- **Capacity:** Typically around 5MB per domain.

Setting Data:

```
sessionStorage.setItem('username', 'Mahesh');
```

Reading Data:

```
const username = sessionStorage.getItem('username');
console.log(username);
```

Removing Data:

```
sessionStorage.removeItem('username');
```

Clearing All Data:

```
sessionStorage.clear();
```

Interview Questions:

Modules

Question 1: What is a module in JavaScript?

Answer: A module in JavaScript is a self-contained unit of code that encapsulates functionality and can be imported and used in other parts of an application. It helps organize code into reusable, maintainable pieces.

Question 2: How do you export and import functions using ES6 modules?

```
// Exporting
// math.js
export function add(a, b) {
    return a + b;
}

export function subtract(a, b) {
    return a - b;
}

// Importing
// app.js
import { add, subtract } from './math.js';
console.log(add(2, 3)); // Output: 5
console.log(subtract(5, 3)); // Output: 2
```

Question 3: What is the difference between named exports and default exports in ES6 modules?

- **Answer:** Named exports allow you to export multiple values from a module, which must be imported using the same names. Default exports allow you to export a single value from a module, which can be imported using any name.

```
// Named Export
export const PI = 3.14;

// Default Export
export default function multiply(a, b) {
   return a * b;
}
```

Question 4: How do you use ES modules in an HTML file?

```
<script type="module" src="index.js"></script>
```

Cookies

Question 1: What is a cookie in JavaScript?

- **Answer:** A cookie is a small piece of data stored on the user's browser by websites to remember information about the user, such as login status, preferences, and tracking information, across different sessions or page visits.

Question 2: How do you set a cookie in JavaScript?

```
document.cookie = "username=Mahesh; expires=Fri, 31 Dec 2021 23:59:59 GMT; path=/";
```

Question 3: How do you read a cookie in JavaScript?

```
console.log(document.cookie);
```

Question 4: How do you delete a cookie in JavaScript?

```
document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 GMT; path=/";
```

Question 5: What are some common attributes used with cookies?

- **Answer:** Common attributes include:
    - expires: Specifies when the cookie should expire.

- path: Specifies the URL path that must exist in the requested URL for the browser to send the cookie header.
- domain: Specifies the domain that must exist in the requested URL for the browser to send the cookie header.
- secure: A flag that indicates the cookie should only be sent over HTTPS.
- SameSite: Controls whether the browser sends the cookie along with cross-site requests (Strict, Lax, None).

Local Storage

Question 1: What is local storage in JavaScript?

- **Answer:** Local storage is a type of web storage that allows you to store data in the browser persistently. Data stored in local storage remains until explicitly deleted and is not sent to the server with each HTTP request.

Question 2: How do you set an item in local storage?

```
localStorage.setItem('username', ‘Mahesh’);
```

Question 3: How do you retrieve an item from local storage?

```
const username = localStorage.getItem('username');
console.log(username);
```

Question 4: How do you remove an item from local storage?

```
localStorage.removeItem('username');
```

Question 5: How do you clear all items from local storage?

```
localStorage.clear();
```

Session Storage

Question 1: What is session storage in JavaScript?

- **Answer:** Session storage is a type of web storage that allows you to store data in the browser for the duration of the page session. Data stored in session storage is cleared when the page session ends (i.e., when the browser tab or window is closed).

Question 2: How do you set an item in session storage?

```
sessionStorage.setItem('username', 'Mahesh');
```

Question 3: How do you retrieve an item from session storage?

```
const username = sessionStorage.getItem('username');
console.log(username);
```

Question 4: How do you remove an item from session storage?

```
sessionStorage.removeItem('username');
```

Question 5: How do you clear all items from session storage?

```
sessionStorage.clear();
```