

MÓDULOS EN JAVASCRIPT Y REPASO



Introducción a los Módulos en JavaScript

Contenido:

- Un módulo es un archivo que contiene código JavaScript reutilizable.
- Facilitan la organización y mantenimiento del código.
- Permiten la separación de preocupaciones y el reuso de componentes.



Exportación de Módulos

Exportación Nombrada (**export**)

```
export const nombre = 'Juan';
export function saludar(){
  console.log('Hola!');
}
```

Exportación por Defecto (**export default**):

```
export default function(){
  console.log('Función por defecto');
}
```

Importación de Módulos

Importación Nombrada (import)

```
import { nombre, saludar } from './mimodulo.js';
```

Importación por Defecto (import)

```
import miFuncion from './otroModulo.js';
```

Importación por Defecto (import)

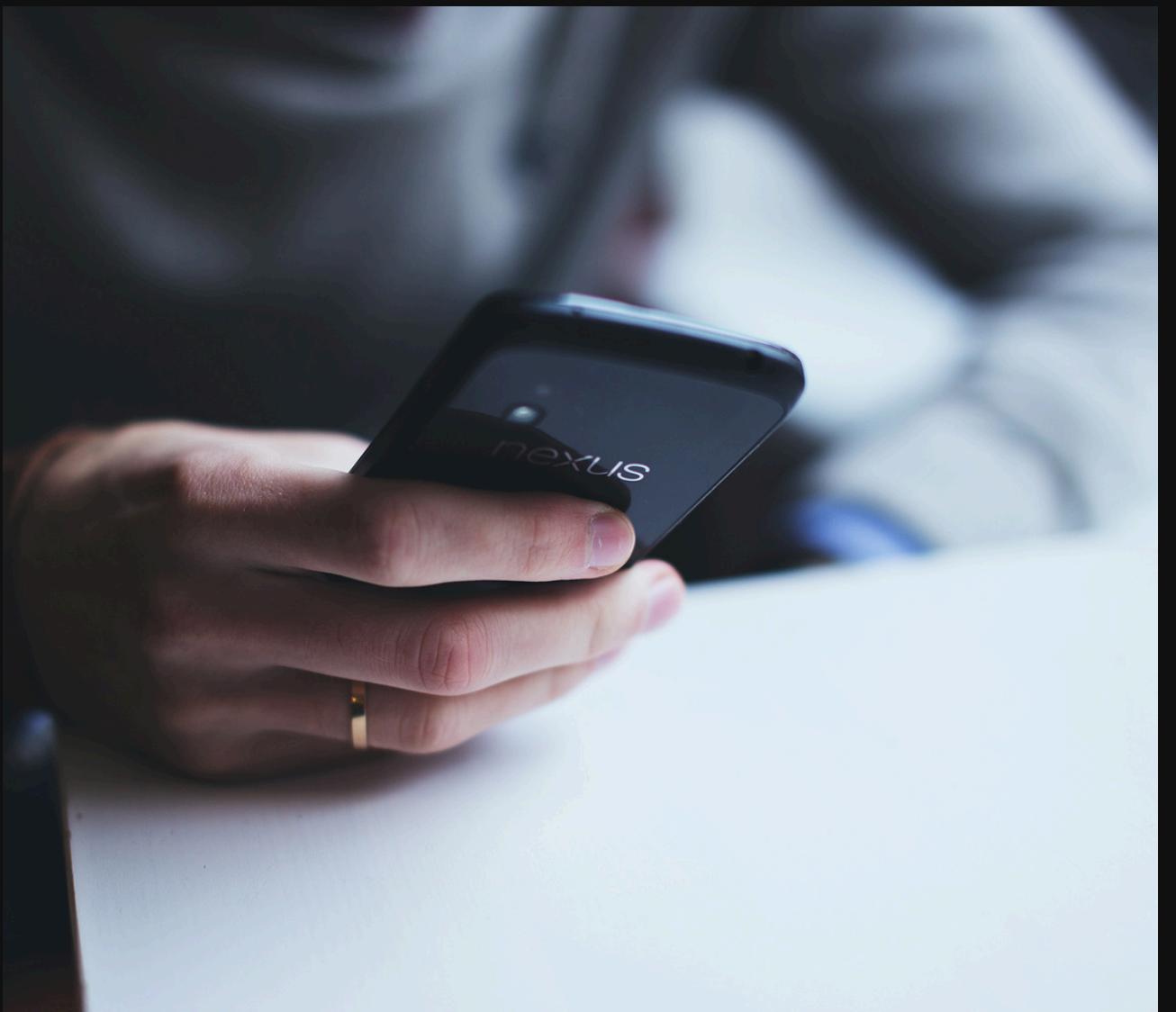
```
import { nombre as nombreUsuario } from './mimodulo.js';
```

Manipulación del Web Storage

localStorage

```
export function guardarEnLocalStorage(clave, valor){  
  localStorage.setItem(clave, JSON.stringify(valor));  
}
```

```
export function obtenerDeLocalStorage(clave){  
  return JSON.parse(localStorage.getItem(clave));  
}
```



Asincronismo en JavaScript

FETCH API

```
export async function obtenerDatos(url){  
    const respuesta = await fetch(url);  
    const datos = await respuesta.json();  
    return datos;  
}
```

PROMESAS Y ASYNC/AWAIT

```
async function mostrarDatos(){  
    const datos = await obtenerDatos('https://api.ejemplo.com/datos');  
    guardarEnLocalStorage('datosApi', datos);  
    actualizarDOM(datos);  
}
```

Preguntas en vivo



Agenda de Contactos Dinámica (hacerlo en clase)

Desarrollar una aplicación web modular que permita a los usuarios gestionar una agenda de contactos. La aplicación debe permitir agregar, editar, eliminar y buscar contactos, utilizando localStorage para persistir los datos, manejar operaciones asincrónicas para obtener información adicional sobre los contactos, y actualizar el DOM de forma dinámica.

Requisitos:

1. Módulos JavaScript:

- Crear un módulo `contactStorage.js` para gestionar la persistencia de contactos en `localStorage`.
- Crear un módulo `api.js` para realizar solicitudes a una API que devuelva información adicional de los contactos (por ejemplo, fotos o detalles de un perfil).
- Crear un módulo `contactDOM.js` para manejar la creación, edición, eliminación y búsqueda de contactos en el DOM.

2. Funcionalidades:

- Agregar Contactos:
 - Los usuarios pueden agregar nuevos contactos con nombre, número de teléfono y correo electrónico.
 - Los contactos se guardan en `localStorage`.
- Mostrar Contactos:
 - Al cargar la página, los contactos guardados en `localStorage` se muestran en una lista.
- Editar Contactos:
 - Los usuarios pueden editar la información de los contactos.
 - Los cambios se reflejan tanto en el DOM como en `localStorage`.
- Eliminar Contactos:
 - Los usuarios pueden eliminar contactos de la lista.
 - Los contactos eliminados también se remueven de `localStorage`.
- Buscar Contactos:
 - Los usuarios pueden buscar contactos por nombre, mostrando solo los que coincidan en la lista.
- Información Adicional:
 - Al agregar o ver un contacto, la aplicación realiza una solicitud asincrónica a una API para obtener información adicional (como una foto de perfil) y la muestra junto con los detalles del contacto.

**Gracias por su
tiempo.... :)**

