Programación IV

Profesora Cynthia Estrada

Fecha: 03/09

"Códigos HTTP"

Integrantes:

- "Díaz Rossini Juan José",
- "Gallo Genaro"
- "Navarro Victor Leandro"

---

1. **¿Qué son los códigos HTTP?**
2. **¿Cuáles son los rangos de los códigos?(Que abarcan cada sección, por ej. Del 100 al 120 son de error)**
3. **¿Qué son los módulos en node?**
4. **¿Cuáles son las características?**
5. **Ventajas de crear módulos**
6. **¿A qué se refiere con importar/exportar un módulo?**
7. **Módulos built-in, módulo consola y console.(Log,warn,warm,error,assert,table)**

¿Qué son los códigos HTTP?

Los códigos HTTP (o códigos de estado) son respuestas que envía un servidor cuando recibe una petición del cliente (por ejemplo, un navegador o una app). Sirven para indicar si la solicitud se procesó correctamente, si hubo un error o si se necesita alguna acción adicional.

¿Cuáles son los rangos de los códigos?

Los códigos HTTP se agrupan en diferentes rangos según el tipo de respuesta:

- **1xx (100–199): Respuestas informativas → Indican que la solicitud fue recibida y el proceso continúa.**
- **2xx (200–299): Respuestas exitosas → La petición se procesó correctamente (ejemplo: 200 OK).**
- **3xx (300–399): Redirecciones → Indican que el cliente debe hacer otra acción para completar la petición (ejemplo: 301 Moved Permanently).**
- **4xx (400–499): Errores del cliente → La petición tiene un error del lado del usuario (ejemplo: 404 Not Found).**
- **5xx (500–599): Errores del servidor → El servidor no pudo completar la solicitud correctamente (ejemplo: 500 Internal Server Error).**

Ejemplos de códigos de estado:

- **200 OK: La solicitud fue procesada con éxito.**
- **301 Moved Permanently: El recurso solicitado fue movido a otra URL de manera permanente.**
- **404 Not Found: El recurso solicitado no existe en el servidor.**

¿Qué son los módulos en Node?

En Node.js, los módulos son bloques de código reutilizables que encapsulan funcionalidades específicas (por ejemplo, manejo de archivos, creación de servidores o conexiones a bases de datos). Sirven para organizar mejor el proyecto y evitar repetir código.

¿Cuáles son las características de los módulos?

- Son reutilizables y permiten dividir el proyecto en partes más manejables.
- Pueden ser internos (propios de Node.js, como fs, http, path), externos (instalados con npm, como express) o creados por el programador.
- Se cargan mediante require o import.
- Facilitan la modularidad y el mantenimiento del código.

Ventajas de crear módulos:

- Mejor organización del código.
- Reutilización en diferentes partes del proyecto.
- Fácil mantenimiento y escalabilidad.
- Permite trabajar en equipo dividiendo responsabilidades.

¿A qué se refiere con importar/exportar un módulo?

- **Exportar un módulo:** Hacer que una función, clase u objeto esté disponible fuera del archivo donde se creó usando module.exports o export.
- **Importar un módulo:** Traer ese módulo en otro archivo para poder usarlo, mediante require o import.

Ejemplo simple en Node.js:

```
// archivo suma.js
function sumar(a, b) {
  return a + b;
}
module.exports = sumar;
// archivo app.js
const sumar = require('./suma');
console.log(sumar(5, 3)); // 8
```

Módulos Built-in en Node.js

Los módulos built-in (o módulos nativos) son aquellos que ya vienen integrados con Node.js, por lo que no necesitan instalarse con npm. Se pueden usar directamente importándolos en el código con require o import.
Ejemplos de módulos built-in:

- **fs → Manejo de archivos.**
- **http → Crear servidores web.**
- **Path → Manejo de rutas de archivos.**

- **Console → Proporciona funciones para imprimir mensajes y depurar programas**

Módulo Console

El módulo console es un módulo built-in que se utiliza para mostrar información en la terminal. Sirve para:

- **Depurar el código mientras se ejecuta.**
- **Mostrar mensajes informativos, advertencias o errores.**
- **Organizar la salida con tablas o verificaciones.**

Ejemplo básico:

```
console.log("Hola, este es un mensaje normal");
console.warn("Advertencia: algo puede estar mal");
console.error("Error: ocurrió un problema");
```

Métodos útiles del módulo console

1. console.log()

Muestra mensajes informativos en la consola.
Es el más usado para la depuración.

```
console.log("Servidor iniciado en el puerto 3000");
```

2. console.warn()

Muestra un mensaje de advertencia.
Se usa para indicar que algo no es crítico, pero debe revisarse.

```
console.warn("El archivo de configuración está vacío, usando valores por defecto");
```

3. console.error()

Muestra un mensaje de error.
Se utiliza cuando ocurre una falla importante en la ejecución.

```
console.error("Error al conectar con la base de datos");
```

4. console.assert(condición, mensaje)

Solo muestra el mensaje si la condición es falsa.
Muy útil para comprobar que algo cumple con lo esperado.

```
let edad = 15;
console.assert(edad >= 18, "Debes ser mayor de edad");
// Muestra el mensaje porque la condición es falsa
```

5. console.table()

Muestra datos en formato de tabla, ideal para arreglos u objetos.

```
const usuarios = [
  { id: 1, nombre: "Juan", rol: "Admin" },
  { id: 2, nombre: "Ana", rol: "User" }
];
console.table(usuarios);
```