FCyT
Sede Concepción del Uruguay

**Ingeniería en Telecomunicaciones**

Desarrollo de Aplicaciones Distribuidas - 2025

**Ejercicios JS**

# 01 Práctica de JavaScript: Redes de Datos

---

## Ejercicio 1: Crear un objeto literal para un dispositivo de red

Crea un objeto literal que represente un router con las siguientes propiedades: modelo, marca, puertos, velocidad (en Mbps), y soportaWifi.

## Ejercicio 2: Array de dispositivos de red

Crea un array con 5 dispositivos de red (routers, switches, firewalls, etc.) donde cada uno sea un objeto literal con propiedades como tipo, marca, modelo y precio.

## Ejercicio 3: Filtrar dispositivos por marca

Dado un array de dispositivos de red, utiliza el método filter() para obtener solo los dispositivos de una marca específica.

```
const dispositivosRed = [
{ tipo: "Router", marca: "Cisco", modelo: "1941", precio: 1200 },
{ tipo: "Switch", marca: "TP-Link", modelo: "TL-SG108", precio: 150 },
{ tipo: "Firewall", marca: "Cisco", modelo: "ASA 5506-X", precio: 2500 },
{ tipo: "Access Point", marca: "Ubiquiti", modelo: "UniFi AP AC Pro", precio: 320 },
{ tipo: "Router", marca: "TP-Link", modelo: "Archer C7", precio: 180 }
];
```

## Ejercicio 4: Mapear direcciones IP

Dado un array de servidores con direcciones IP, utiliza el método map() para crear un nuevo array que contenga solo las direcciones IP.

```
const servidores = [
 { nombre: "Servidor Web", ip: "192.168.1.10", sistema: "Linux" },
 { nombre: "Servidor de Base de Datos", ip: "192.168.1.11", sistema: "Windows" },
 { nombre: "Servidor de Correo", ip: "192.168.1.12", sistema: "Linux" },
 { nombre: "Servidor DNS", ip: "192.168.1.13", sistema: "Linux" },
 { nombre: "Servidor de Archivos", ip: "192.168.1.14", sistema: "Windows" }
];
```

## Ejercicio 5: Filtrar y ordenar paquetes de datos

Dado un array de paquetes de datos, filtra aquellos que tengan un tamaño mayor a 1000 bytes y ordénalos de mayor a menor según su prioridad.

```
const paquetesDatos = [
 { id: 1, origen: "192.168.1.5", destino: "192.168.1.10", tamaño: 1200, prioridad: 3 },
 { id: 2, origen: "192.168.1.7", destino: "192.168.1.12", tamaño: 800, prioridad: 1 },
 { id: 3, origen: "192.168.1.3", destino: "192.168.1.11", tamaño: 1500, prioridad: 5 },
 { id: 4, origen: "192.168.1.8", destino: "192.168.1.14", tamaño: 950, prioridad: 2 },
 { id: 5, origen: "192.168.1.2", destino: "192.168.1.13", tamaño: 2000, prioridad: 4 }
];
```

## Ejercicio 6: Calcular estadísticas de red

Dado un objeto con estadísticas de tráfico de red por hora, calcula el total de datos transferidos y la hora con mayor tráfico.

```
const traficoRed = {
 "08:00": 1250,  // MB transferidos
 "09:00": 1870,
 "10:00": 2100,
 "11:00": 1950,
 "12:00": 1600,
 "13:00": 1300,
 "14:00": 1700,
```

```
 "15:00": 2200,
 "16:00": 1800,
 "17:00": 1500
};
// Calcula el total de datos transferidos
// Encuentra la hora con mayor tráfico
```

## Ejercicio 7: Analizar conexiones de red

Dado un array de conexiones de red, agrupa las conexiones por protocolo y cuenta cuántas hay de cada tipo.

```
const conexiones = [
 { id: 1, origen: "192.168.1.5", destino: "192.168.1.10", protocolo: "HTTP" },
 { id: 2, origen: "192.168.1.7", destino: "192.168.1.12", protocolo: "FTP" },
 { id: 3, origen: "192.168.1.3", destino: "192.168.1.11", protocolo: "SSH" },
 { id: 4, origen: "192.168.1.8", destino: "192.168.1.14", protocolo: "HTTP" },
 { id: 5, origen: "192.168.1.2", destino: "192.168.1.13", protocolo: "HTTPS" },
 { id: 6, origen: "192.168.1.6", destino: "192.168.1.10", protocolo: "FTP" },
 { id: 7, origen: "192.168.1.9", destino: "192.168.1.15", protocolo: "SSH" },
 { id: 8, origen: "192.168.1.4", destino: "192.168.1.11", protocolo: "HTTP" }
];

// Crea un objeto para contar las conexiones por protocolo

// Tu código aquí (recorre el array y cuenta las conexiones)

console.log("Conexiones por protocolo:", conexionesPorProtocolo);
```

## Ejercicio 8: Filtrar y transformar alertas de seguridad

Dado un array de alertas de seguridad de red, filtra las que sean de nivel "alto" y transfórmalas en mensajes para el administrador.

```
const dispositivos = [
```

```
{ id: 1, nombre: "PC-Desarrollo", ip: "192.168.1.5", tipo: "Estación de trabajo" },
{ id: 2, nombre: "PC-Marketing", ip: "192.168.1.7", tipo: "Estación de trabajo" },
{ id: 3, nombre: "Servidor-Web", ip: "192.168.1.10", tipo: "Servidor" },
{ id: 4, nombre: "Servidor-BD", ip: "192.168.1.11", tipo: "Servidor" }
];

const conexionesActivas = [
{ origen: "192.168.1.5", destino: "192.168.1.10", protocolo: "HTTP", bytes: 8500 },
{ origen: "192.168.1.7", destino: "192.168.1.11", protocolo: "MySQL", bytes: 12000 },
{ origen: "192.168.1.5", destino: "192.168.1.11", protocolo: "MySQL", bytes: 9200 }
];

// Crea un informe que combine la información de dispositivos y conexiones
const informeActividad = conexionesActivas.map(conexion => {
// Encuentra los dispositivos de origen y destino
// Tu código aquí
 // Retorna un objeto con la información combinada
return {
  // Tu código aquí
};
});

console.log("Informe de actividad de red:", informeActividad);
```

## Ejercicio 9: Combinar datos de dispositivos y conexiones

Combina información de dispositivos y conexiones para crear un informe detallado de la actividad de red.

```
const topologiaRed = {
nodos: [
  { id: "A", tipo: "Router", ubicacion: "Planta 1" },
  { id: "B", tipo: "Switch", ubicacion: "Planta 1" },
  { id: "C", tipo: "Switch", ubicacion: "Planta 2" },
  { id: "D", tipo: "Switch", ubicacion: "Planta 3" },
```

```
  { id: "E", tipo: "Router", ubicacion: "Planta 3" }
 ],
 conexiones: [
  { origen: "A", destino: "B", ancho_banda: 1000 },
  { origen: "A", destino: "C", ancho_banda: 1000 },
  { origen: "B", destino: "C", ancho_banda: 100 },
  { origen: "B", destino: "D", ancho_banda: 100 },
  { origen: "C", destino: "D", ancho_banda: 100 },
  { origen: "C", destino: "E", ancho_banda: 1000 },
  { origen: "D", destino: "E", ancho_banda: 1000 }
 ]
};

// Cuenta el número de conexiones por nodo
const conexionesPorNodo = {};
topologiaRed.nodos.forEach(nodo => {
 conexionesPorNodo[nodo.id] = 0;
});

// Tu código aquí para contar las conexiones

// Encuentra los nodos con más conexiones
const nodosOrdenados = Object.entries(conexionesPorNodo)
 .sort(/* Tu código aquí para ordenar de mayor a menor */);

// Sugiere optimizaciones (por ejemplo, los nodos con más de 2 conexiones podrían necesitar más ancho de banda)
const sugerencias = [];
// Tu código aquí

console.log("Conexiones por nodo:", conexionesPorNodo);
console.log("Nodos ordenados por número de conexiones:", nodosOrdenados);
console.log("Sugerencias de optimización:", sugerencias);
```

## Ejercicio 10: Analizar y optimizar topología de red

Dado un objeto que representa una topología de red, encuentra los nodos con más conexiones y sugiere optimizaciones.

```
const topologiaRed = {
 nodos: [
  { id: "A", tipo: "Router", ubicacion: "Planta 1" },
  { id: "B", tipo: "Switch", ubicacion: "Planta 1" },
  { id: "C", tipo: "Switch", ubicacion: "Planta 2" },
  { id: "D", tipo: "Switch", ubicacion: "Planta 3" },
  { id: "E", tipo: "Router", ubicacion: "Planta 3" }
 ],
 conexiones: [
  { origen: "A", destino: "B", ancho_banda: 1000 },
  { origen: "A", destino: "C", ancho_banda: 1000 },
  { origen: "B", destino: "C", ancho_banda: 100 },
  { origen: "B", destino: "D", ancho_banda: 100 },
  { origen: "C", destino: "D", ancho_banda: 100 },
  { origen: "C", destino: "E", ancho_banda: 1000 },
  { origen: "D", destino: "E", ancho_banda: 1000 }
 ]
};

// Cuenta el número de conexiones por nodo
const conexionesPorNodo = {};
topologiaRed.nodos.forEach(nodo => {
 conexionesPorNodo[nodo.id] = 0;
});

// Tu código aquí para contar las conexiones

// Encuentra los nodos con más conexiones
```

```
const nodosOrdenados = Object.entries(conexionesPorNodo)
 .sort(/* Tu código aquí para ordenar de mayor a menor */);
```