

JS



PROFE VIDERMID

Programación en Java Script

(Parte 8)

Facilitador: Ing. Esp. Vidermid Sánchez



@vidermid



+584147106623



@ingenieriadigitalsc

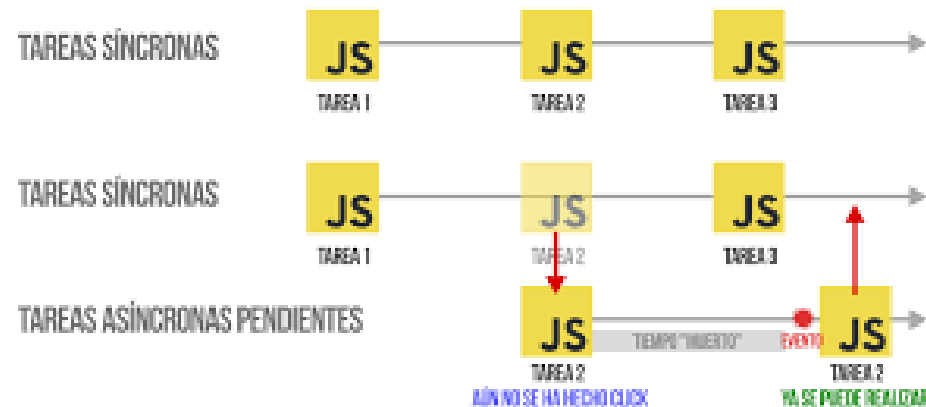


+584147464801



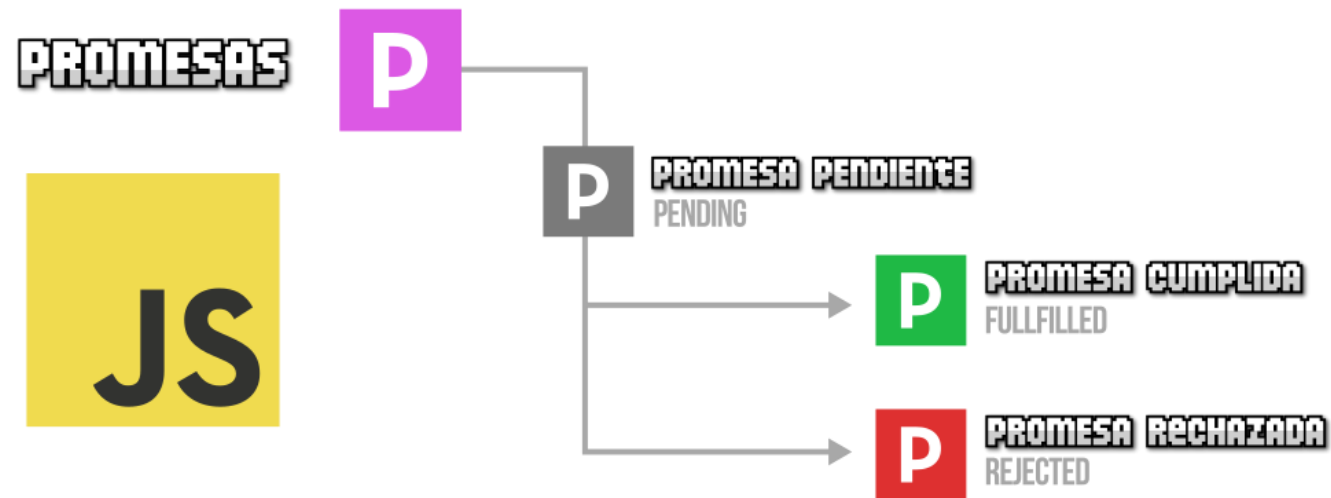
Programación Asíncrona

Es una técnica que permite a tu programa iniciar una tarea de larga duración y seguir respondiendo a otros eventos mientras esa tarea se ejecuta, en lugar de tener que esperar hasta que esa tarea haya terminado. Una vez que dicha tarea ha finalizado, tu programa presenta el resultado.



Promesas en JS

Las promesas en JavaScript representan procesos que ya están sucediendo, que se pueden encadenar con funciones de devolución de llamada. Si está buscando evaluar lentamente una expresión, considere usar una función sin argumentos,



Async/await

Existe una sintaxis especial para trabajar con promesas de una forma más confortable, llamada “async/await”. Es sorprendentemente fácil de entender y usar.

Funciones async

La palabra “async” ante una función significa solamente una cosa: que la función siempre devolverá una promesa. Otros valores serán envueltos y resueltos en una promesa automáticamente. Ejemplo:

```
async function f() {  
  return Promise.resolve(1);  
}
```

```
f().then(alert); // 1
```

Entonces, `async` se asegura de que la función devuelva una promesa, o envuelve las no promesas y las transforma en una. Bastante simple, ¿correcto? Pero no solo eso. Hay otra palabra, `await`, que solo trabaja dentro de funciones `async` y es muy interesante, `await` hace que JavaScript espere hasta que la promesa responda y devuelve su resultado. Ejemplo:

```
async function f() {  
    let promise = new Promise((resolve, reject) => {  
        setTimeout(() => resolve("¡Hecho!"), 1000)  
    });  
    let result = await promise; // espera hasta que la promesa se resuelva (*)  
    alert(result); // "¡Hecho!"  
}  
f();
```

Nota: La ejecución de la función es pausada en la línea (*) y se reanuda cuando la promesa responde, con `result` volviéndose su resultado. Entonces el código arriba muestra “¡Hecho!” en un segundo.

Método fetch()

El método fetch de la nueva API fetch busca y trae recursos como datos, imágenes, objetos, JSON de otros archivos (fetch significa ir a buscar algo, traer algo). Ejemplo: utilizar la función fetch para traer una imagen .jpg y mostrarla en pantalla.

La palabra clave await puede ser utilizada solamente en funciones asíncronas, Empleamos la palabra clave async para especificar que una función se ejecuta de manera asíncrona. Ejemplo:

Primero necesitamos crear una función asíncrona

```
async function funcionAsincrona(url){
```

```
//espera (await) una respuesta de fetch()
```

```
const respuesta = await fetch(url);
```

```
//espera (await) el blob, Los Blobs representan datos que no necesariamente se encuentran en formato nativo de JS.
```

```
const respuesta2= await response.blob();
```

```
//y cuando lo tenga crea un nuevo objeto url y utilízalo como valor del atributo src de la imagen
```

```
fetchTest.src = URL.createObjectURL(blob)
```

```
}
```


Método fetch()

HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-
width, initial-scale=1.0">
  <link rel="shortcut icon" href="#" type="image/x-
icon">
  <title>Prueba JS</title>
</head>
<body>
  <h1>Prueba JS Asíncrono con Promesas</h1>
  <img src="" id="imagen" />
  <p id="output"></p>
  <script src="lib.js"></script>
</body>
</html>
```

JS

```
fetch("http://127.0.0.1:5500//hola.jpg")
  //Crear el objeto response
  .then(respuesta=>{
    console.log(respuesta+'Soy el response');
    return respuesta.blob();
  })

  // crea el objeto de lo que viene de la URL
  .then(respuesta2=>{
    console.log(respuesta2);
    imagen.src = URL.createObjectURL(respuesta2)
  })

  //exception si hay error
  .catch(error =>{console.error(error)})
```

¿Qué es JSON?

JSON significa **J**ava **S**cript **O**bject **N**otation, es un formato para almacenar, y transportar datos, se usa a menudo cuando los datos se envían desde un servidor a una página web y/o aplicaciones, sin importar el lenguaje de programación o plataforma en el que están realizados.



La sintaxis JSON se deriva de la sintaxis de notación de objetos de JavaScript, pero el formato JSON es solo texto. El código para leer y generar datos JSON se puede escribir en cualquier lenguaje de programación.

JSON es de fácil lectura y escritura para los usuarios. Es fácil de analizar y generar por parte de las máquinas, se basa en un subconjunto del lenguaje de programación JavaScript, Estándar ECMA-262 3ra Edición - Diciembre de 1999.

La siguiente sintaxis JSON define un objeto de docentes: una matriz de 3 registros de docentes (objetos):

```
{  
  "docentes": [  
    {"nombre": "Pedro", "apellido": "Parra"},  
    {"nombre": "Gabriel", "apellido": "Paredes"},  
    {"nombre": "Maria", "apellido": "Cabeza"}  
  ]  
}
```

El formato JSON es sintácticamente idéntico al código para crear objetos JavaScript, debido a esta similitud, un programa JavaScript puede convertir fácilmente datos JSON en objetos JavaScript nativos.

Reglas de sintaxis JSON

- Los datos están en pares de nombre/valor.
- Los datos están separados por comas.
- Las llaves sostienen objetos.
- Los corchetes contienen matrices.

Los datos JSON se escriben como pares de nombre/valor, al igual que las propiedades de objeto de JavaScript. Un par de nombre/valor consta de un nombre de campo (entre comillas dobles), seguido de dos puntos, seguido de un valor:

```
"curso":"python"
```

Los objetos JSON se escriben entre llaves, al igual que en JavaScript, los objetos pueden contener varios pares de nombre/valor:

```
"diplomado":[  
    {"modulo":"python", "docente":"pedro"},  
    {"modulo":"react", "docente":"gabriel"},  
    {"modulo":"jsp", "docente":"maria"}  
]
```

Crear un script .JSON

```
{  
  "cedula": "30159753",  
  "apellido": "Urdaneta",  
  "nombre": "Francisco",  
  "hobbies": ["comer", "bailar", "dormir"],  
  "edad": 18,  
  "notas": [  
    {"asignatura": "matematica",  
      "nota": 18  
    },  
    {"asignatura": "fisica",  
      "nota": 15  
    },  
    {"asignatura": "quimica",  
      "nota": 16  
    },  
  ],  
}
```

```
    {"asignatura": "programacion",  
      "nota": 20  
    },  
  ],  
  "direccion": {  
    "pais": "venezuela",  
    "estado": "zulia",  
    "ciudad": "machiques",  
    "municipio": "san felix",  
    "parroquia": "las adjuntas",  
    "calle": "Av, Urdaneta",  
    "casa": "# 4-88"  
  }  
}
```

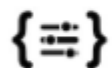
JS

Extensión JSON Viewer



chrome web store

[Inicio](#) > [Extensiones](#) > [JSON Viewer](#)



JSON Viewer

Destacados

★★★★★ 1.008

← → ↻ ⓘ 127.0.0.1:5500/prueba1.json

```
1 // 20230606202839
2 // http://127.0.0.1:5500/prueba1.json
3
4 {
5   "cedula": "30159753",
6   "apellido": "Urdaneta",
7   "nombre": "Francisco",
8   "hobbies": [↔],
13  "edad": 18,
14  "notas": [↔],
32  "direccion": {↔}
41 }
```

Un uso común de JSON es leer datos de un servidor web y mostrar los datos en una página web, para simplificar, esto se puede demostrar usando una cadena como entrada. Primero, cree una cadena JavaScript que contenga la sintaxis JSON.

Posteriormente se utiliza la función incorporada de JavaScript `JSON.parse()`, con ella se puede convertir la cadena en un objeto de JavaScript,

Ejemplo:

```
const objeto = JSON.parse(texto);
```

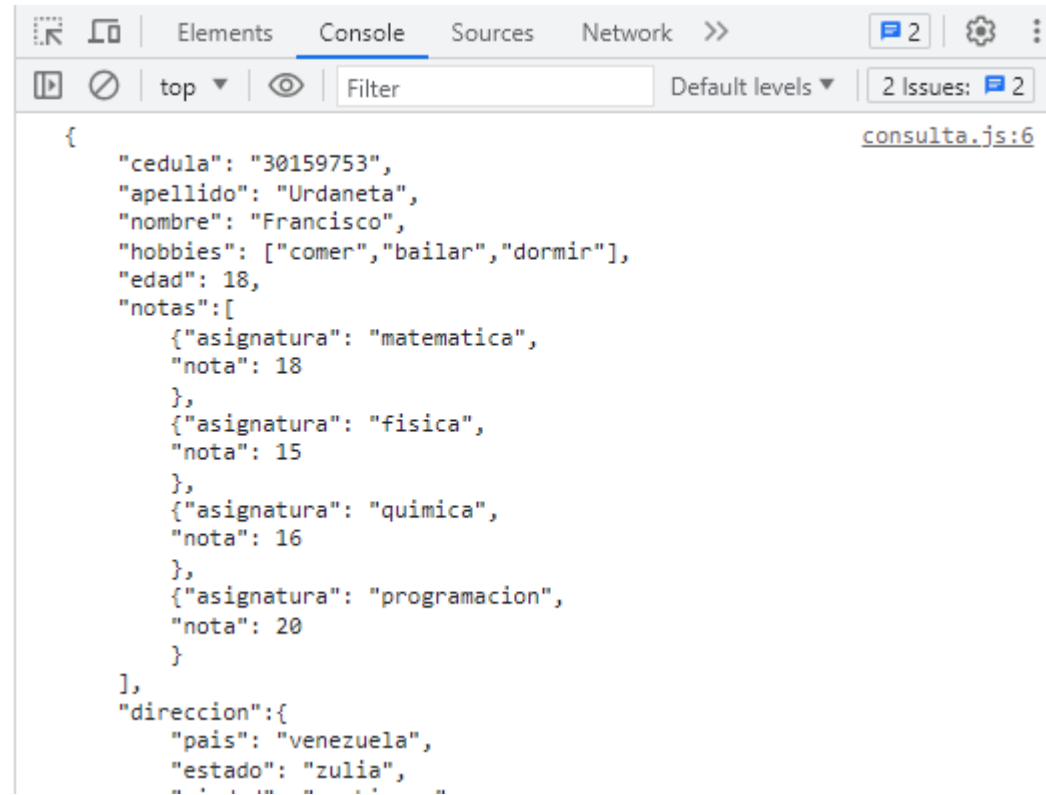

También existe el método `JSON.stringify()`, éste convierte un objeto o valor de JavaScript en una cadena de texto JSON, opcionalmente reemplaza valores si se indica una función de reemplazo, o si se especifican las propiedades mediante un array de reemplazo.

Ejemplos:

<code>JSON.stringify({});</code>	<code>// '{}'</code>
<code>JSON.stringify(true);</code>	<code>// 'true'</code>
<code>JSON.stringify('hola');</code>	<code>// '"hola"'</code>
<code>JSON.stringify([1, 'false', false]);</code>	<code>// '[1,"false",false]'</code>

Captura de un JSON en un script JS

```
async function consu(){  
  const respuesta = await fetch("http://127.0.0.1:5500/prueba1.json");  
  const arregloJson = await respuesta.text();  
  console.log(arregloJson);  
}  
consu();
```

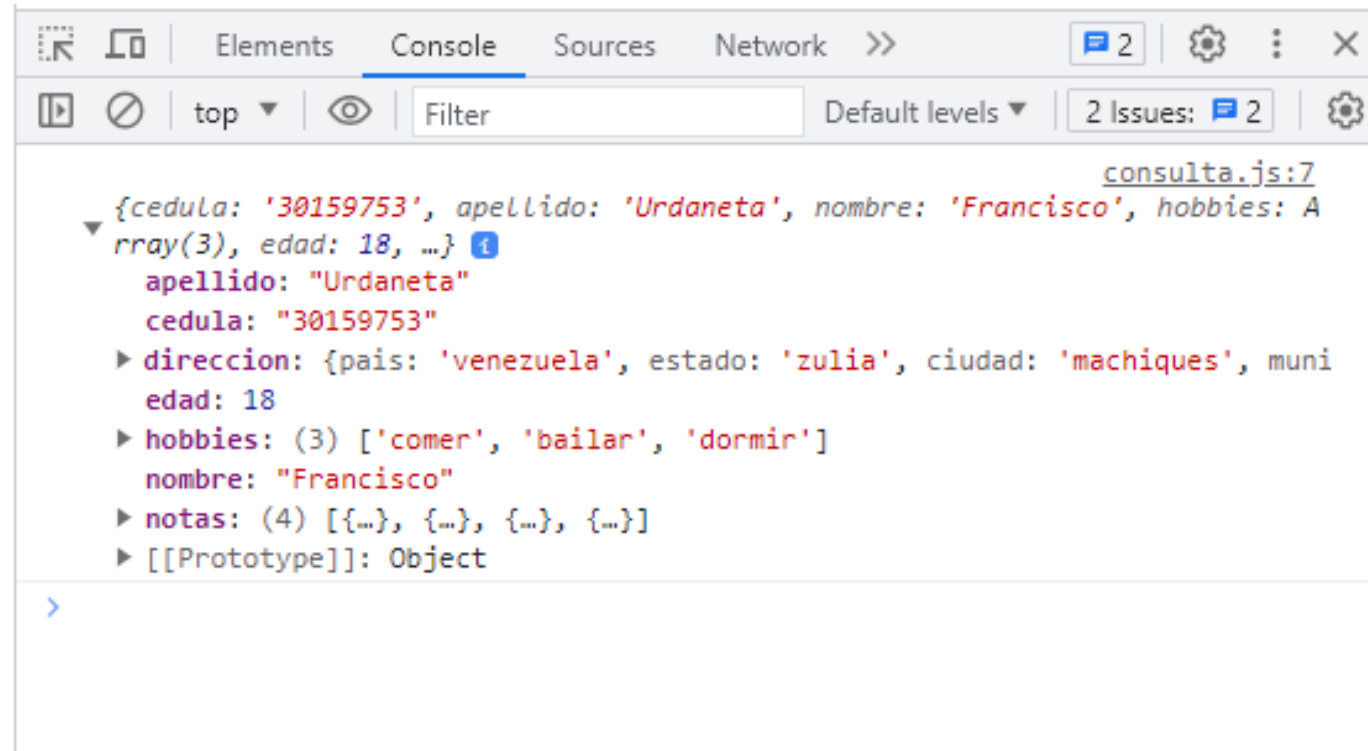


The screenshot shows a web browser's developer console with the 'Console' tab selected. It displays a JSON object returned from a fetch call. The JSON structure includes personal information like 'cedula', 'apellido', 'nombre', 'hobbies', and 'edad', as well as an array of 'notas' (grades) for different subjects and a 'direccion' object with 'pais' and 'estado'.

```
{  
  "cedula": "30159753",  
  "apellido": "Urdaneta",  
  "nombre": "Francisco",  
  "hobbies": ["comer", "bailar", "dormir"],  
  "edad": 18,  
  "notas": [  
    {"asignatura": "matematica",  
     "nota": 18},  
    {"asignatura": "fisica",  
     "nota": 15},  
    {"asignatura": "quimica",  
     "nota": 16},  
    {"asignatura": "programacion",  
     "nota": 20},  
  ],  
  "direccion": {  
    "pais": "venezuela",  
    "estado": "zulia",  
  }  
}
```

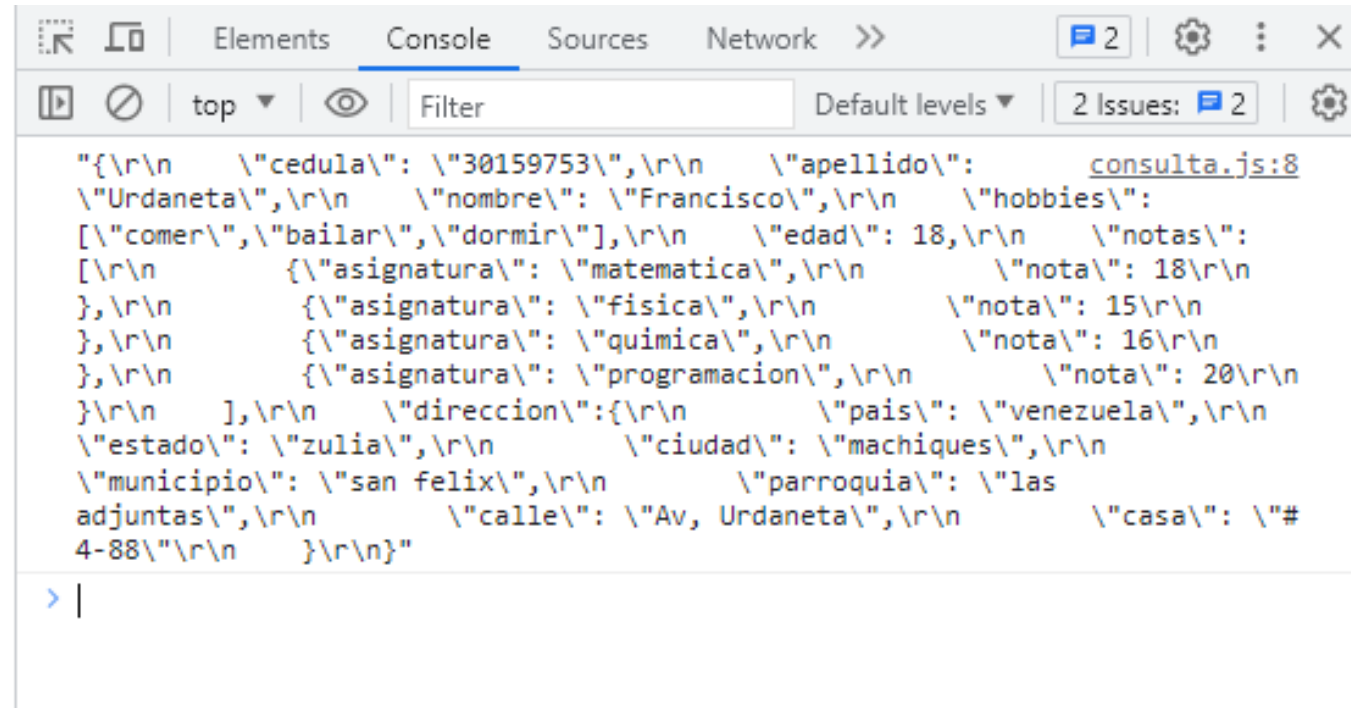
Captura de un JSON en un script JS

```
async function consu(){  
  const respuesta = await fetch("http://127.0.0.1:5500/prueba1.json");  
  const arregloJson = await respuesta.text();  
  console.log(JSON.parse(arregloJson));  
}  
consu();
```



Captura de un JSON en un script JS

```
async function consu(){  
  const respuesta = await fetch("http://127.0.0.1:5500/prueba1.json");  
  const arregloJson = await respuesta.text();  
  console.log(JSON.stringify(arregloJson));  
}  
consu();
```



```
{  
  "cedula": "30159753",  
  "apellido": "Urdaneta",  
  "nombre": "Francisco",  
  "hobbies": ["comer", "bailar", "dormir"],  
  "edad": 18,  
  "notas": [  
    {"asignatura": "matematica", "nota": 18},  
    {"asignatura": "fisica", "nota": 15},  
    {"asignatura": "quimica", "nota": 16},  
    {"asignatura": "programacion", "nota": 20}  
  ],  
  "direccion": {  
    "pais": "venezuela",  
    "estado": "zulia",  
    "ciudad": "machiques",  
    "municipio": "san felix",  
    "parroquia": "las adjuntas",  
    "calle": "Av, Urdaneta",  
    "casa": "#4-88"  
  }  
}
```

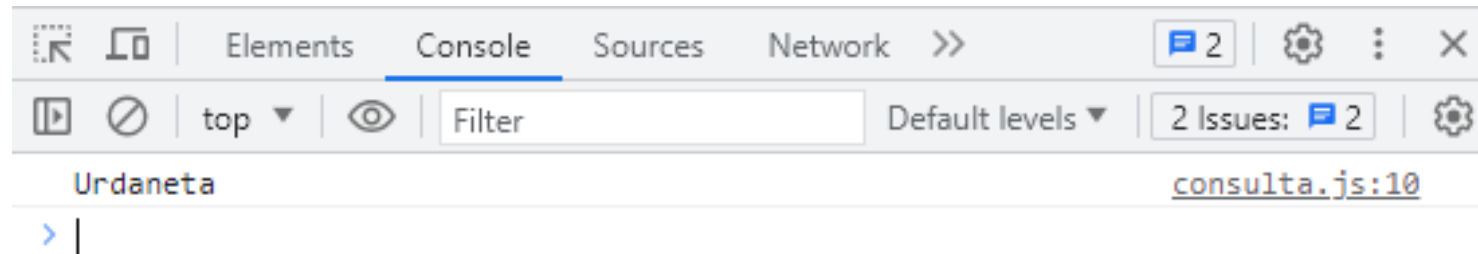
Captura de un JSON en un script JS

```
async function consu(){  
  const respuesta = await fetch("http://127.0.0.1:5500/prueba1.json");  
  const arregloJson = await respuesta.json();  
  console.log(arregloJson);  
}  
consu();
```



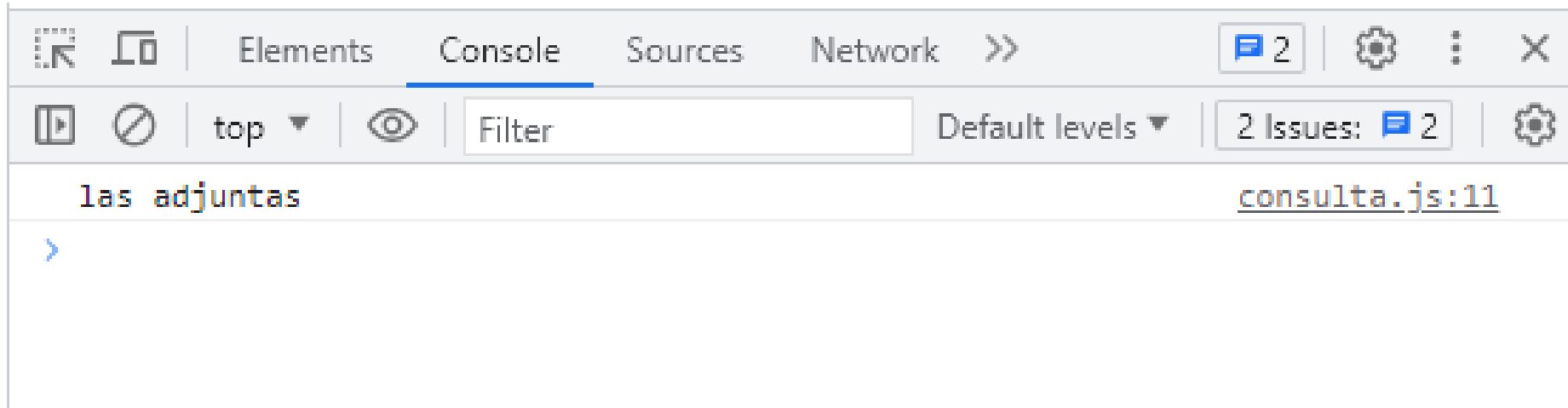
Captura de un JSON en un script JS

```
async function consu(){  
  const respuesta = await fetch("http://127.0.0.1:5500/prueba1.json");  
  const arregloJson = await respuesta.json();  
  console.log(arregloJson.apellido);  
}  
consu();
```



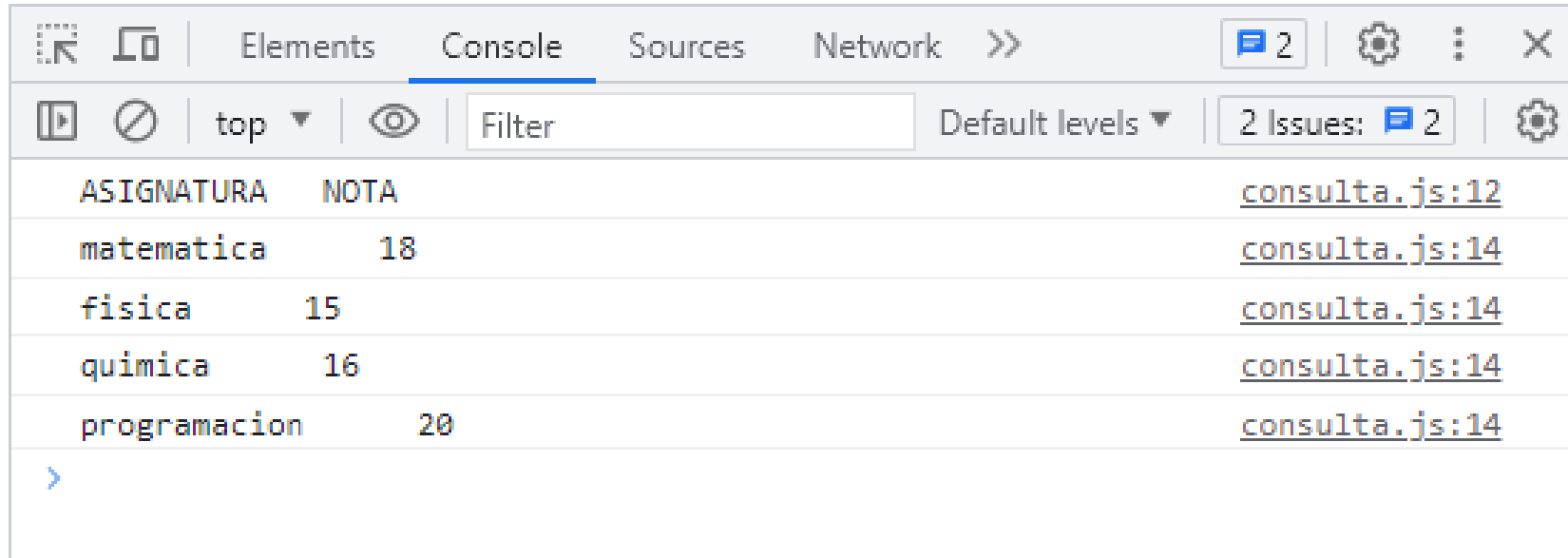
Captura de un JSON en un script JS

```
async function consu(){  
  const respuesta = await fetch("http://127.0.0.1:5500/prueba1.json");  
  const arregloJson = await respuesta.json();  
  console.log(arregloJson.direccion.parroquia);  
}  
consu();
```



Captura de un JSON en un script JS

```
async function consu(){  
  const respuesta = await fetch("http://127.0.0.1:5500/prueba1.json");  
  const arregloJson = await respuesta.json();  
  console.log(`ASIGNATURA  NOTA\n`);  
  arregloJson.notas.forEach(element => {  
    console.log(`${element.asignatura}    ${element.nota}`);  
  });  
}  
consu();
```



ASIGNATURA	NOTA	consulta.js:12
matematica	18	consulta.js:14
fisica	15	consulta.js:14
quimica	16	consulta.js:14
programacion	20	consulta.js:14

Captura de un JSON en un script JS

```
async function consu(){  
  const respuesta = await fetch("http://127.0.0.1:5500/prueba1.json");  
  const arregloJson = await respuesta.json();  
  console.log(arregloJson.direccion["casa"]);  
}  
consu();
```

