Fichero: json.js

```
/**
 * API para obtener datos con AJAX en JSON de una forma fácil y sencilla. Admite tanto GET como POST.
 * Para comprender cómo funciona el método que acontece, es necesario ver cómo funcionan las promesas de JavaScript:
 * https://desarrolloweb.com/articulos/introduccion-promesas-es6.html
 * @author Santiago San Pablo Raposo
 * @version 25.02.2023
 */


/**
 * Función que permite obtener un JSON desde un servidor web. Sin los parámetros opcionales, el comportamiento por defecto es usar el método GET sin parámetros.
 * @param {string} url Especifica la URL desde la que se quiere obtener el JSON.
 * @param {string} metodo (Opcional) Especifica mediante una cadena de texto el metodo que se quiere emplear en la cabecera de la petición HTTP.
 * @param {object} datos (Opcional) Especifica un objeto literal con los parámetros que deseas enviar al servidor.
 * @returns Un JSON / objeto literal con los datos que devuelva la API en el servidor en la ruta especificada.
 */
function obtenerJSON(url, metodo = "GET", datos = null) {
    return new Promise((resolve, reject) => {
        /*-- Verificar datos de los parámetros --*/
        if (metodo != "GET" && metodo != "POST" && metodo != "PUT" && metodo != "DELETE") {
            reject("El método especificado no es correcto. Especifica GET, POST, PUT o DELETE");
        }

        if (datos != null && typeof(datos) != "object") {
            reject("Especifica los parámetros de la petición en un objeto literal/JSON");
        }

        /*-- Efectúa la petición al servidor --*/
        switch (metodo) {
            case "GET":
                /*-- Prepara los parámetros en un objeto de tipo URL --*/
                let direccion = new URL(url);
                for (let key in datos) {
                    direccion.searchParams.append(key, datos[key]);
                }

                /*-- Realiza la petición al servidor --*/
                fetch(direccion, {method: metodo})
                    .then(response => {
                        console.dir(response);
```

Fichero: json.js

Fichero: json.js

```
                    if (response.ok) {
                        return response.text();
                    } else {
                        reject("No se ha podido acceder a ese recurso. Status: " + response.status);
                    }
                })
                .then(responseText => {
                    let objJson = JSON.parse(responseText);
                    resolve(objJson);
                })
                .catch(err => reject(err));
            break;
        case "POST":
            /*-- Prepara los parámetros en un objeto de tipo URLSearchParams --*/
            let parametros = new URLSearchParams();
            for (let key in datos) {
                parametros.append(key, datos[key]);
            }

            /*-- Realiza la petición al servidor --*/
            fetch(url, {method: metodo, body: datos})
                .then(response => {
                    if (response.ok) {
                        return response.text();
                    } else {
                        reject("No se ha podido acceder a ese recurso. Status: " + response.status);
                    }
                })
                .then(responseText => {
                    let objJson = JSON.parse(responseText);
                    resolve(objJson);
                })
                .catch(err => reject(err));
            break;
        default:
            /*-- PUT y DELETE no están implementados, ya que no he estudiado cómo funcionan --*/
            reject("No implementado el método " + metodo + ".");
            break;
    }
  });
}
```

Fichero: ejercicio1.html

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejercicio 1</title>

    <script src="json.js"></script>
    <script src="ejercicio1.js"></script>
    <link rel="stylesheet" href="ejercicio1.css">
</head>
<body>
    <form>
        <input type="text" placeholder="Escribe un nombre">
        <input type="submit" value="Buscar">
    </form>
    <table hidden>
        <thead>
            <th>Nombre</th>
            <th>Calle</th>
            <th>Ciudad</th>
        </thead>
        <tbody>

        </tbody>
    </table>
    <p class="error" hidden></p>
</body>
</html>
```

Fichero: ejercicio1.html

Fichero: ejercicio1.js

```
/*=============================================
            VARIABLES GLOBALES
=============================================*/
let datos;
let tbody;

/*=============================================
            FUNCIONES Y EVENTOS
=============================================*/

document.addEventListener("DOMContentLoaded", () => {
    /*-- Obtiene los objetos de la pagina --*/
    const formulario = document.querySelector("form");
    const tabla = document.querySelector("table");
    tbody = tabla.querySelector("tbody");
    const error = document.querySelector(".error");

    /*-- Controla eventos --*/
    formulario.addEventListener("submit", form_onSubmit);

    /*-- Obtiene los datos de la API y los vuelca en la tabla --*/
    obtenerJSON('https://jsonplaceholder.typicode.com/users')
    .then(obj => {
        console.dir(obj)
        cargarTabla(obj, tbody);

        /*-- Muestra la tabla ya rellena --*/
        tabla.hidden = false;
        error.hidden = true;

        /*-- Almacena el objeto JSON en una variable global --*/
        datos = obj;
    })
    .catch(err => {
        tabla.hidden = true;
        error.textContent = err;
        error.hidden = false;
    })
});

/**
 * Función que carga los datos de un objeto JSON en una tabla HTML.
 * @param {object} objJSON Especifica el objeto JSON con los datos del servidor.
 * @param {object} tabla Especifica el objeto HTML <table>.
 * @param {function} condicion Especifica una función que se ejecutará como algoritmo para
filtrar los resultados. Esta función deberá tener un parámetro elem, que corresponderá con un
elemento del objeto JSON.
 */
function cargarTabla(objJSON, tabla, condicion = null) {
    /*-- Define el algoritmo --*/
    function cargar(elem) {
        /*-- Crea una nueva fila --*/
        let fila = document.createElement("tr");
        tabla.append(fila);

        /*-- Crea una celda para el Nombre --*/
        let nombre = document.createElement("td");
        nombre.textContent = elem.name;
        fila.append(nombre);
```

Fichero: ejercicio1.js

```
        /*-- Crea una celda para la calle --*/
        let calle = document.createElement("td");
        calle.textContent = elem.address.street;
        fila.append(calle);

        /*-- Crea una celda para la ciudad --*/
        let ciudad = document.createElement("td");
        ciudad.textContent = elem.address.city;
        fila.append(ciudad);
    }

    // if (typeof(objJSON) != "object") {
    //     throw new Error("El parámetro 'objJSON' debe ser un objeto JSON.")
    // }

    // if (typeof(tabla) != "object") {
    //     throw new Error("El parámetro 'tabla' debe corresponder con el objeto de la tabla
HTML.")
    // }

    // if (typeof(condicion) != "function") {
    //     throw new Error("El parámetro 'condicion' debe ser una función.")
    // }

    /*-- Lo ejecuta solo si cumple la condicion (en caso de tenerla, si no la tiene, tambien
se ejecuta) --*/
    for (let i of objJSON) {
        if (condicion == null) {
            cargar(i);
        } else {
            if (condicion(i)) {
                // console.dir(i);
                cargar(i);
            }
        }
    }
}

function form_onSubmit(evento) {
    /*-- Previene la acción submit por defecto --*/
    evento.preventDefault();

    /*-- Variables --*/
    const form = evento.target;

    /*-- Actualiza la tabla con los nuevos datos solicitados --*/
    tbody.innerHTML = ""; // Vacía el tbody para volver a rellenarlo

    cargarTabla(datos, tbody, (elem) => {
        let condicion = elem.name.includes(form[0].value);
        return condicion;
    });
}
```

Fichero: ejercicio1.css

```css
body {
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-size: 12px;
}

table {
    /* border-style: solid;
    border-width: 1px;
    border-color: #cccccc; */
    border-collapse: collapse;
}

table thead th {
    border-style: solid;
    border-width: 1px;
    border-color: #cccccc;
    margin: 0;
    padding: 0.5em;

    background-color: #f0f8ff;
}

table td {
    border-style: solid;
    border-width: 1px;
    border-color: #cccccc;
    padding: 0.5em;
}

table tr:nth-child(odd) {
    /* https://www.w3schools.com/CSSref/sel_nth-child.php */
    background-color: #f0f8ff;
}

.error {
    color: red
}
```

Fichero: ejercicio1.css

Fichero: ejercicio2.html

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejercicio 2</title>

    <script src="json.js"></script>
    <script src="ejercicio2.js"></script>
    <link rel="stylesheet" href="ejercicio2.css">
</head>
<body>
    <div class="container">
        <header class="row">
            <select>

            </select>
        </header>
        <main class="row">
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
        </main>
    </div>
</body>
</html>
```

Fichero: ejercicio2.html

Fichero: ejercicio2.js

```
/*============================================
            VARIABLES GLOBALES
============================================*/
let main;
let categorias = new Set(); // https://es.javascript.info/map-set#set

/*============================================
            FUNCIONES Y EVENTOS
============================================*/

document.addEventListener("DOMContentLoaded", () => {
    /*-- Obtiene los objetos de la pagina --*/
    const select = document.querySelector("header select");
    main = document.querySelector("main");

    /*-- Controla eventos --*/
    select.addEventListener("change", sl_onChange);

    /*-- Obtiene los datos de la API y los vuelca en el main --*/
    main.innerHTML = "";
    getDatosFromAPI();
});

/**
 * Función que permite obtener los datos de la API realizando una petición HTTP, con una serie de parámetros (opcionales).
 * @param {object} parametros (Opcional) Objeto literal que contiene los parámetros que se le pasarán a la petición HTTP.
 */
function getDatosFromAPI(parametros = null) {
    obtenerJSON("https://www.raulserranoweb.es/tienda/rest.php", "GET", parametros)
        .then(obj => {
            // console.dir(obj); // Debug
            /*-- Carga las opciones del select --*/
            if (categorias.size == 0) {
                cargarSelect(obj)
            }

            /*-- Muestra la tabla ya rellena --*/
            cargarArticulos(obj, main);

            /*-- Almacena el objeto JSON en una variable global --*/
            datos = obj;
        })
        .catch(err => {
            let error = document.createElement("p");
            error.className = "error";
            error.textContent = err;
            main.append(error);
        });
}

/**
 * Función que carga las categorías de bicicletas en el select.
 * @param {object} objJSON Especifica el objeto JSON con los datos del servidor.
 */
function cargarSelect(objJSON) {
    /*-- Obtiene el select a manipular --*/
    const select = document.querySelector("header select");
```

```
    /*-- Carga las categorías en un conjunto (Set) y después las ordena alfabéticamente --*/
    for (let elem of objJSON) {
        categorias.add(elem.cat);
    }
    categorias = Array.from(categorias).sort();

    /*-- Despues las carga en el select --*/
    let opcionTodas = document.createElement("option");
    opcionTodas.value = "Todas";
    opcionTodas.textContent = "Todas";
    select.append(opcionTodas);
    for (let elem of categorias) {
        let opcion = document.createElement("option");
        opcion.value = elem;
        opcion.textContent = elem;
        select.append(opcion);
    }
}

/**
 * Función que carga los datos de un objeto JSON en una tabla creada a base de divs y flex como sistema de posicionamiento de los mismos.
 * @param {object} objJSON Especifica el objeto JSON con los datos del servidor.
 * @param {object} container Especifica el objeto HTML en el que se insertará.
 * @param {function} condicion Especifica una función que se ejecutará como algoritmo para filtrar los resultados. Esta función deberá tener un parámetro elem, que corresponderá con un elemento del objeto JSON.
 */
function cargarArticulos(objJSON, container, condicion = null) {
    /*-- Define el algoritmo --*/
    function cargar(elem) {
        /*-- Crea un nuevo elemento dentro del main --*/
        let articulo = document.createElement("article");
        articulo.className = "col";
        container.append(articulo);

        /*-- Crea una img --*/
        let contenedorImg = document.createElement("div");
        let imagen = document.createElement("img");
        imagen.src = "https://raulserranoweb.es/tienda/imagenes_art/" + elem.cod;
        contenedorImg.append(imagen);
        articulo.append(contenedorImg);

        /*-- Crea la descripcion del producto con la informacion requerida por el enunciado --*/
        let divText = document.createElement("div");
        let nombre = elem.nom;
        let descripcion = elem.des;
        let categoria = elem.cat;

        divText.insertAdjacentHTML("beforeend",
            "<b>Nombre:</b> " + nombre + "<br>" +
            "<b>Descripci&oacute;n:</b> " + descripcion + "<br>" +
            "<b>Categor&iacute;a:</b> " + categoria
        );

        articulo.append(divText);
    }

    /*-- Lo ejecuta solo si cumple la condicion (en caso de tenerla, si no la tiene, tambien se ejecuta) --*/
```

Fichero: ejercicio2.js

```
    for (let i of objJSON) {
        if (condicion == null) {
            cargar(i);
        } else {
            if (condicion(i)) {
                // console.dir(i);
                cargar(i);
            }
        }
    }
}

function sl_onChange(evento) {
    /*-- Variables --*/
    const sl = evento.target;

    /*-- Actualiza la tabla con los nuevos datos solicitados --*/
    main.innerHTML = ""; // Vacía el tbody para volver a rellenarlo
    getDatosFromAPI(sl.options.selectedIndex == 0 ? null : {cat:
sl.options[sl.options.selectedIndex].value})
}
```

Fichero: ejercicio2.css

```css
* {
    box-sizing: border-box;
    /*-- Variables --*/
    --breakpoint-sm: 576px;
    --sizeMaxWrapper: 1200px;
    --sizeMinArticle: 250px;
    --sizeMinImg: 150px;
}

body {
    margin: 0;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 14px;
}

header {
    padding: 0.5em;
    background-color: beige
    position: sticky;
    top: 0;
}

.error {
    color: red
}

.container {
    margin: 0 auto;
    max-width: var(--sizeMaxWrapper);
}

.row {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    margin-bottom: 0.5em;
}

.col {
    width: 100%;

    /* background-color: red;
    border-width: 1px;
    border-color: black;
    border-style: solid; */
}

main article {
    text-align: center;
    /* min-height: 33vh; */
    min-height: var(--sizeMinArticle);
    padding: 1em;
}

main article img {
    /* display: block; */
    object-fit: scale-down;
    width: 100%;
    /* height: 30vh; */
```

Fichero: ejercicio2.css

Fichero: ejercicio2.css

```
    height: var(--sizeMinImg);
}

@media screen and (min-width: 576px) { /* var(--breakpoint-sm) */
    .col {
        width: 50%;
    }

    main article img {
        /* height: 22vh; */
        height: calc(var(--sizeMinImg) + 5vw - 0.05*var(--breakpoint-sm));
    }
}
```

Fichero: ejercicio2.css