

PRACTICA 10:

Web Scraping



TAREA 1:

Paso 1: Diseño de la Página Web

Primero, crea la estructura básica de tu página web utilizando HTML y CSS.

HTML (index.html)

Crea un archivo **index.html** con la siguiente estructura:

```
<!DOCTYPE html>
<html lang="es">

<head>
  <meta charset="UTF-8">
  <title>Practica10</title>
  <link rel="stylesheet" href="style.css">
</head>

<body>
  <h1>Tienda Online</h1>
  <div id="productos"></div>
  <script src="script.js"></script>
</body>

</html>
```

```
<!DOCTYPE html> <html lang="es"> <head> <meta charset="UTF-8"> <title>Tienda
Online</title> <link rel="stylesheet" href="style.css"> </head> <body> <h1>Bienvenido a
Nuestra Tienda Online</h1> <div id="productos"></div> <script src="script.js"></script>
</body> </html>
```

CSS (style.css)

Crea un archivo **style.css** para dar estilo a la página:

```
#productos {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-around;
}

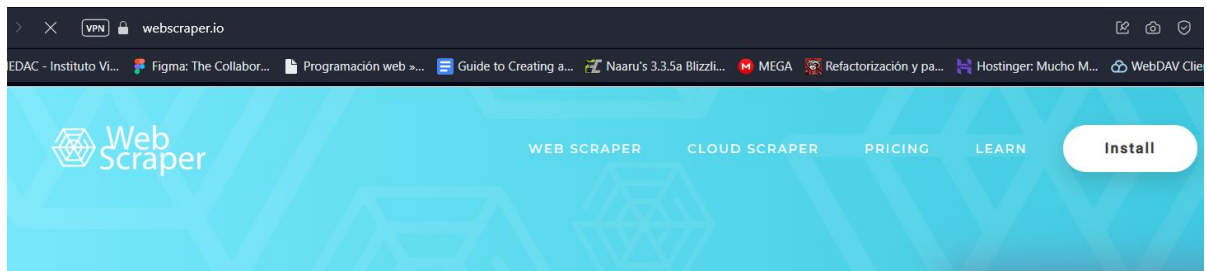
.producto {
  margin: 10px;
  text-align: center;
}
```

```
#productos { display: flex; flex-wrap: wrap; justify-content: space-around; } .producto { margin:
10px; text-align: center; }
```

Paso 2: Web Scraping para Obtener Datos de Productos

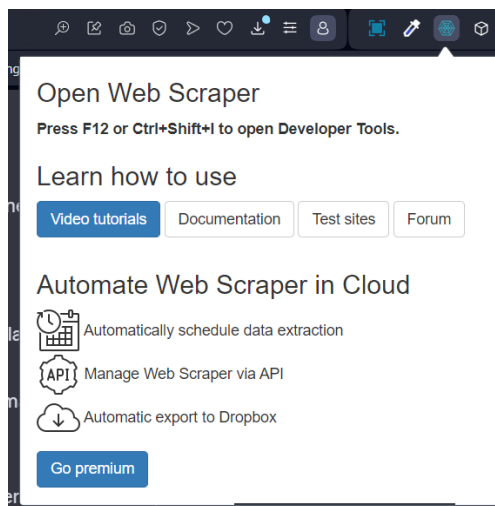
1. Descargamos la extensión WebScraper

Buscamos la extensión en Google o a través de este URL: <https://webscraper.io>

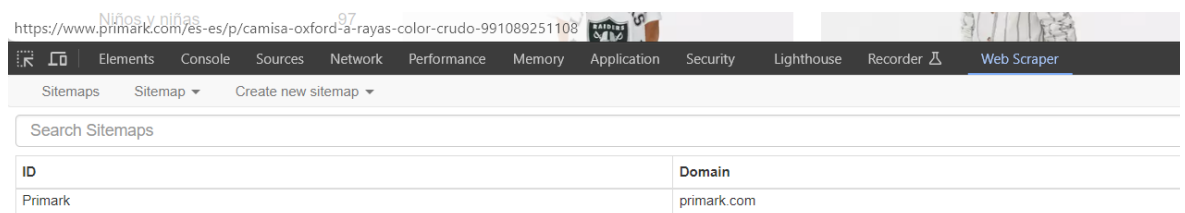


2. Abrimos la Extensión y Creamos un Nuevo Proyecto:

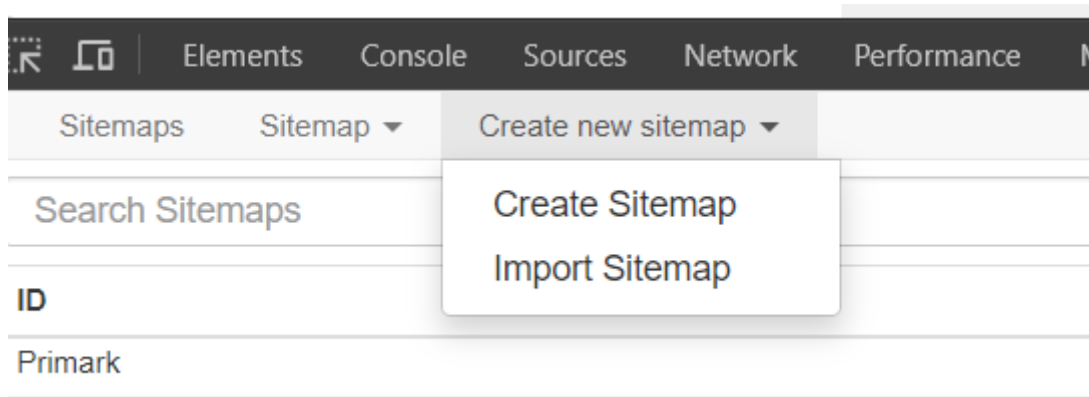
Una vez instalada, hacemos clic en el ícono de Web Scraper en la barra de extensiones de Chrome. Nos dirá de presionar unos comandos para abrir la consola con la herramienta.



Una vez presionados buscamos la pestaña de WebScraper



En la página de Web Scraper, hacemos clic en "Create new sitemap".



3. Configuramos el Sitemap:

Introducimos un nombre para nuestro sitemap en mi caso Primark y hacemos clic en "Create Sitemap".

A screenshot of the 'Create new sitemap' form. It has two input fields: 'Sitemap name' with the value 'Primark' and 'Start URL 1' with the value 'https://primark.com'. Below these fields is a 'Create Sitemap' button.

En la nueva página, introducimos la URL del sitio web que deseamos raspar y hacemos clic en "Add new selector".

4. Configuramos Selectores:

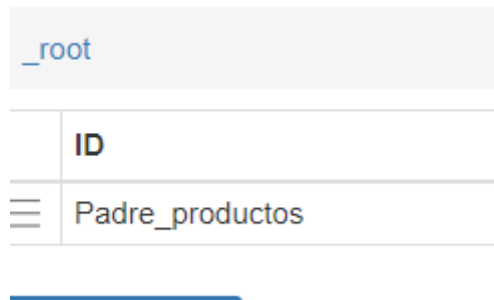
Seleccionamos el tipo de selector para los elementos que deseamos extraer (por ejemplo, "Elemento" para un elemento específico o "Enlace" para seguir enlaces).

A screenshot of the 'Add new selector' dialog box. It has a 'Category' dropdown menu with options: Text, Link, Image, Table, Element attribute, HTML, Element, Element scroll down, Element click, Grouped, Sitemap.xml Links, and Pagination. Below the dropdown is a 'Type' input field with the value 'Text'. At the bottom, there are three buttons: 'Select', 'Element preview', and 'Data preview'. There is also a 'Multiple' checkbox.

Hacemos clic derecho en el elemento en la vista previa de la página y seleccionamos "Select".

Configuramos las propiedades del selector, como nombre, tipo de dato, y atributos.

Yo el primer selector que me he creado lo he llamado “Padre_selector”

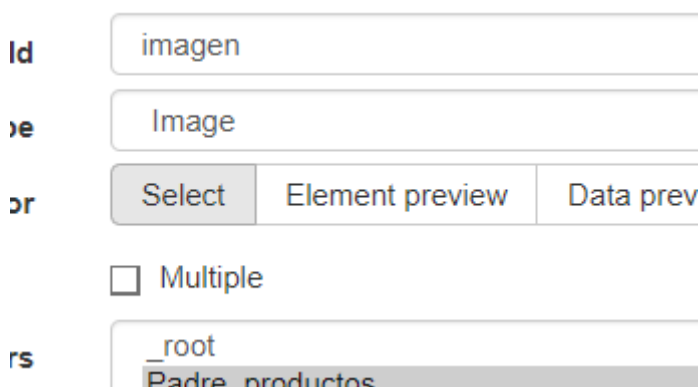


y dentro los selectores de título, precio e imagen.



5. Creamos Campos:

- Después de configurar los selectores, hacemos clic en cada uno en el apartado de “Edit” y “Select” para seleccionar para cada pieza de información que deseamos extraer (título, precio, imagen).



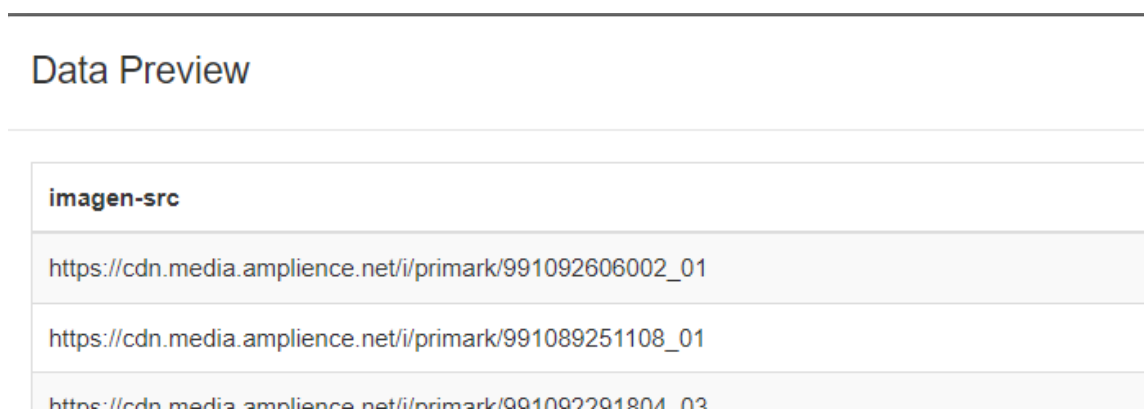
- Seleccionamos los elementos, asignamos un nombre al campo, seleccionamos el tipo selector correspondiente y seleccionamos al selector padre.

6. Ejecutamos la Prueba:

1. Volvemos al selector /root.



2. Hacemos clic en "DataPreview"

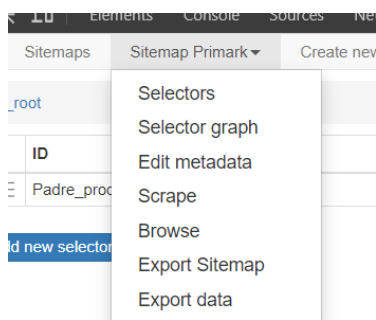


7. Revisamos los Datos Extraídos:

1. Después de ejecutar la prueba, revisamos la vista previa para asegurarnos de que los datos se extraigan correctamente.

8. Exportamos Datos a CSV:

1. Hacemos clic en "Export data" y seleccionamos "CSV".



2. Elegimos un nombre de archivo y la ubicación de guardado.
3. Hacemos clic en "Export" para guardar los datos y lo guardamos como productos.csv

Paso 3: Convertir CSV a JSON

Utilizamos un script de Python para convertir los datos de CSV a JSON.

Python (csv_to_json.py)

```
import csv
import json

productos = [] # List to store product data

# Reading data from CSV file
with open('productos.csv', 'r', encoding='utf-8') as csvfile:
    csvreader = csv.DictReader(csvfile)
    for row in csvreader:
        productos.append(row)

# Writing data to JSON file
with open('productos.json', 'w', encoding='utf-8') as jsonfile:
    json.dump(productos, jsonfile, indent=4)
```

```
import csv import json productos = [] with open('productos.csv', 'r', encoding='utf-8') as csvfile:
csvreader = csv.DictReader(csvfile) for row in csvreader: productos.append(row) with
open('productos.json', 'w', encoding='utf-8') as jsonfile: json.dump(productos, jsonfile,
indent=4)
```

Y entramos en la consola, buscamos el directorio del .py y lo lanzamos con el comando

Python csv_to_json.py

```
C:\Users\Personal\Documents\DAW\2DAW\Desarrollo_Servidor\3trimestre>python csv_to_json.py
```

Paso 4: Configuración del Servidor Web Node.js

Creamos un servidor básico con Node.js para servir tu página web y los datos de los productos.

Node.js (server.js)

```
JS server.js > ...
1  const express = require('express');
2  const app = express();
3  const port = 3000; app.use(express.static('.'));
4
5  // Sirve los archivos estáticos de tu proyecto
6  app.get('/productos', (req, res) => {
7    res.sendFile(__dirname + '/productos.json');
8  });
9  app.listen(port, () => {
10    console.log(`Servidor escuchando en http://localhost:${port}`);
11  });
```

javascriptCopy code

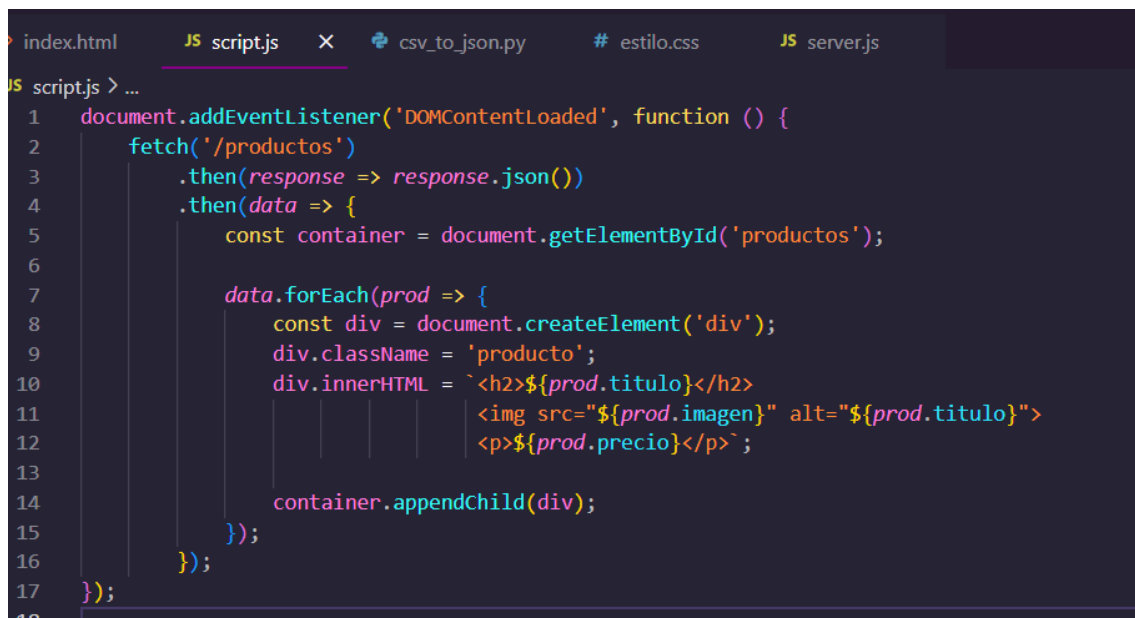
```
const express = require('express'); const app = express(); const port = 3000;
app.use(express.static('.')); // Sirve los archivos estáticos de tu proyecto app.get('/productos',
(req, res) => { res.sendFile(__dirname + '/productos.json'); }); app.listen(port, () => {
console.log(`Servidor escuchando en http://localhost:${port}`); });
```

Para ejecutar este servidor, necesitas tener Node.js y el paquete Express con el comando “npm install express” instalados. Este servidor servirá tu archivo HTML y los datos de los productos en formato JSON.

Paso 5: JavaScript para Cargar y Mostrar Datos en el Cliente

Finalmente, utiliza JavaScript en el lado del cliente para solicitar y mostrar los datos de los productos. Es importante asegurarse que las variables tienen el mismo nombre que las asignadas en el csv.

JavaScript (script.js)



```
1 document.addEventListener('DOMContentLoaded', function () {
2   fetch('/productos')
3     .then(response => response.json())
4     .then(data => {
5       const container = document.getElementById('productos');
6
7       data.forEach(prod => {
8         const div = document.createElement('div');
9         div.className = 'producto';
10        div.innerHTML = `<h2>${prod.titulo}</h2>
11                          
12                          <p>${prod.precio}</p>`;
13
14        container.appendChild(div);
15      });
16    });
17 });
```

javascriptCopy code

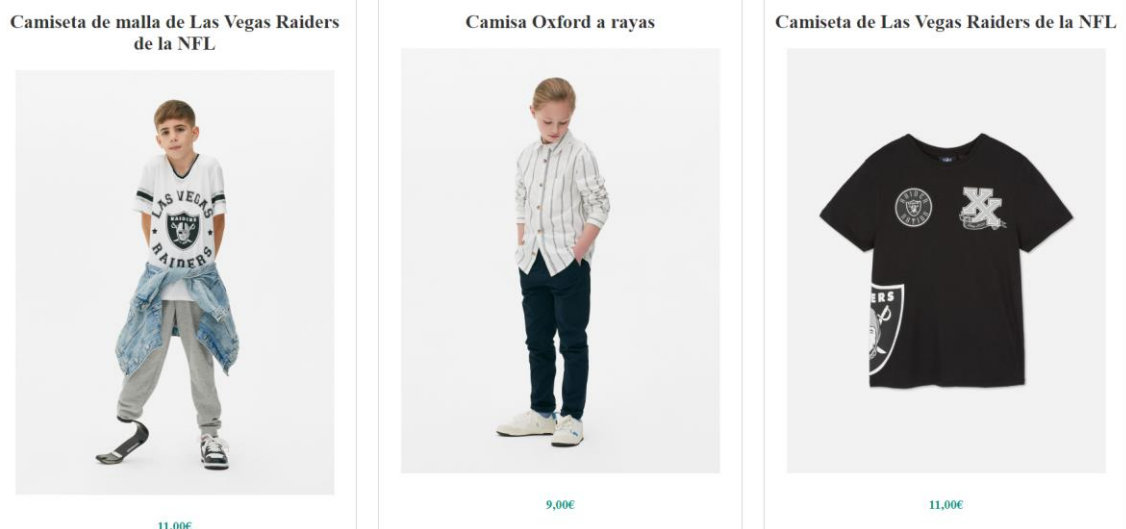
```
document.addEventListener('DOMContentLoaded', function() { { fetch('/productos')
.then(response => response.json()) .then(data => { const container =
document.getElementById('productos'); data.forEach(prod => { const div =
document.createElement('div'); div.className = 'producto'; div.innerHTML =
`<h2>${prod.Título}</h2><p>${prod.Precio}</p>`; container.appendChild(div); }); }); });
```

Este script hace una solicitud al servidor para obtener los datos de los productos y luego los muestra en la página web.

Lanzamos el servidor por comando: node server.js

```
C:\Users\Personal\Documents\DAW\2DAW\Desarrollo_Servidor\3trimestre>node server.js
Servidor escuchando en http://localhost:3000
```

Chequeamos el URL: <http://localhost:3000>



Tarea 5: Integración de Google Maps en la Web

La última parte de nuestro proyecto consistió en integrar un mapa de Google Maps en nuestro sitio web para mostrar la ubicación de la tienda. Esto fue particularmente útil para mejorar la experiencia del usuario proporcionando una referencia visual de la ubicación de la tienda.

- **Integración de Google Maps:**

1. **Obtener la API Key:** Primero, obtuvimos una API Key desde la consola de Google Cloud. Esto implicó crear un proyecto en la consola de Google Cloud, habilitar la API de Google Maps y obtener la clave API.
2. **HTML (index.html):**

htmlCopy code

```
<div id="mapa"></div> <script src="https://maps.googleapis.com/maps/api/js?key=TU_API_KEY&callback=iniciarMapa"></script>
```

Agregamos un **div** con el ID 'mapa' en nuestro archivo HTML donde se mostraría el mapa. Además, incluimos el script de Google Maps API, reemplazando **TU_API_KEY** con nuestra clave API real.

3. **JavaScript para Google Maps (mapa.js):**

javascriptCopy code

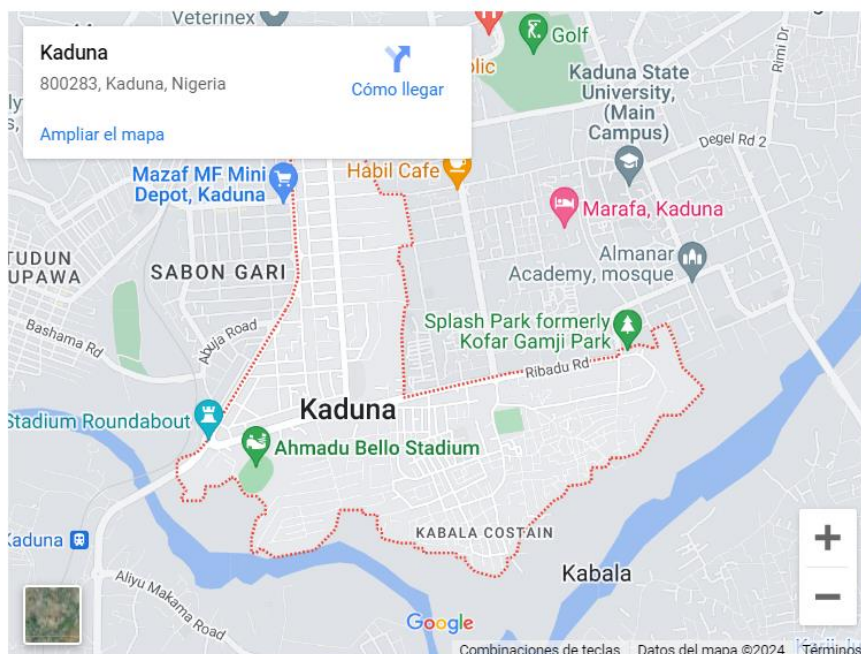
```
function iniciarMapa() { var ubicacion = { lat: -34.397, lng: 150.644 }; // Coordenadas de ejemplo
var mapa = new google.maps.Map(document.getElementById('mapa'), { zoom: 8, center:
ubicacion }); var marcador = new google.maps.Marker({ position: ubicacion, map: mapa }); }
```

Creamos una función **iniciarMapa** para inicializar el mapa de Google Maps. Esta función configura la ubicación inicial del mapa y coloca un marcador en la ubicación de la tienda.

Al integrar Google Maps, nos aseguramos de que los usuarios pudieran obtener una visualización clara de la ubicación física de la tienda, lo que añadió un valor significativo a la experiencia del usuario en nuestra tienda online. Además, esta integración nos brindó experiencia práctica en el trabajo con APIs de terceros y su incorporación en aplicaciones web.

EXTRA MAPS GOOGLE:

Tienda Online



```
<iframe
  src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d15691.
6346046109!2d7.428940692007741!3d10.50786093399893!2m3!1f0!2f0!3f0!3m2!1i
1024!2i768!4f13.1!3m3!1m2!1s0x104d355834371775%3A0x480195979abfe174!2sKad
una%20800283%2C%20Kaduna%2C%20Nigeria!5e0!3m2!1ses!2ses!4v1705947629452!5
m2!1ses!2ses"
  width="600" height="450" style="border:0;" allowfullscreen=""
  Loading="lazy"
  referrerpolicy="no-referrer-when-downgrade"></iframe>
```