NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE | NTU Academy for Professional and Continuing Education

(SCTP) Advanced Professional Certificate

Data Science and AI

MICROSERVICES 101

Schema Evolution Explained

In 6 Minutes!

https://www.youtube.com/watch?v=XG-EVX6PEFo

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

2.4 Data Extraction and Web Scraping

Module Overview

2.1 Introduction to Big Data and Data Engineering

2.2 Data Architecture

2.3 Data Encoding and Data Flow

2.4 Data Extraction and Web Scraping

2.5 Data Warehouse

2.6 Data Pipelines

2.7 Testing and Data Orchestration

2.8 Out of Core/Memory Processing

2.9 Big Data Ecosystem and Batch Processing

2.10 Event Streaming and Stream Processing

Agenda

- Recap
  - REST API
  - GraphQL
- Web Scraping
  - HTML & CSS
  - CSS Selectors

REST Architecture

HTTP Methods

Standard operations like GET, POST, PUT, DELETE for resource interaction

Loose Coupling

Systems designed for minimal interdependence and flexibility

Scalability

Ability to handle increasing demands efficiently

Uniform Interface

Consistent interaction with resources via URIs using JSON or XML

Status Codes

Standardized HTTP status codes indicating success or failure

Recap on REST API

- **REST** is an architectural style that uses standard HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources.
- Typically used for loosely-coupled systems.
- More flexible and scalable for web-based applications and APIs.
- Rely on standardized URIs and HTTP methods to interact with resources, often using JSON or XML as the data format.
- Provides a uniform interface and encourages the use of HTTP status codes to indicate the success or failure of an operation.

GraphQL Architecture

Data control

Clients control data request.

Efficiency

Reduces over and under fetching.

Single endpoint

One endpoint for interactions.

Schema definition

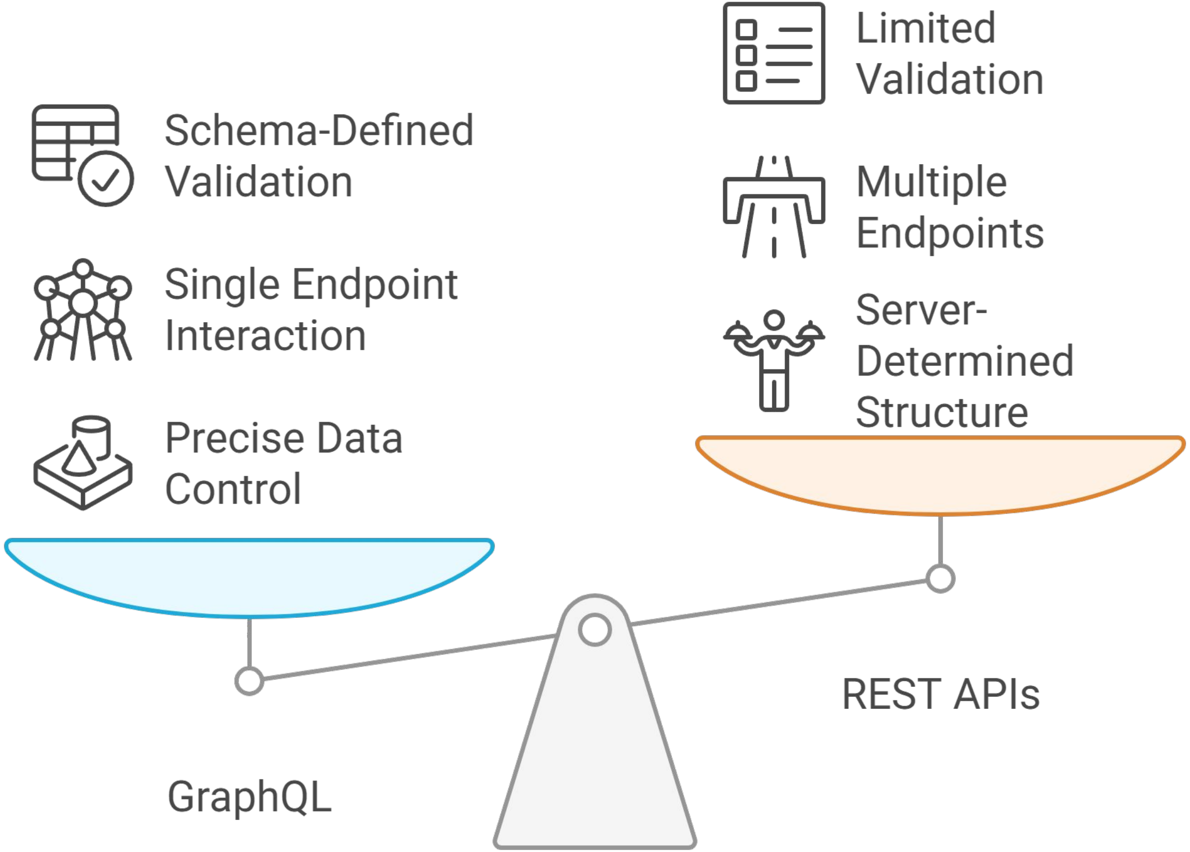
Data types and relationships defined.

Contract

Validation and documentation improvement.

Recap on GraphQL

- **GraphQL** is an alternative query language/runtime for APIs.
- Clients request specific data and structure of response, giving them precise control over the information they receive.
- Unlike traditional REST APIs, where the server determines the response structure, hence reduces over-fetching and under-fetching of data.
- Single endpoint for client interactions.
- Defined by a schema that describes the types of data available and the relationships.
- Serves as a contract between the client and server, ensuring better validation and documentation.



Limited Validation

Multiple Endpoints

Server-Determined Structure

REST APIs

Schema-Defined Validation

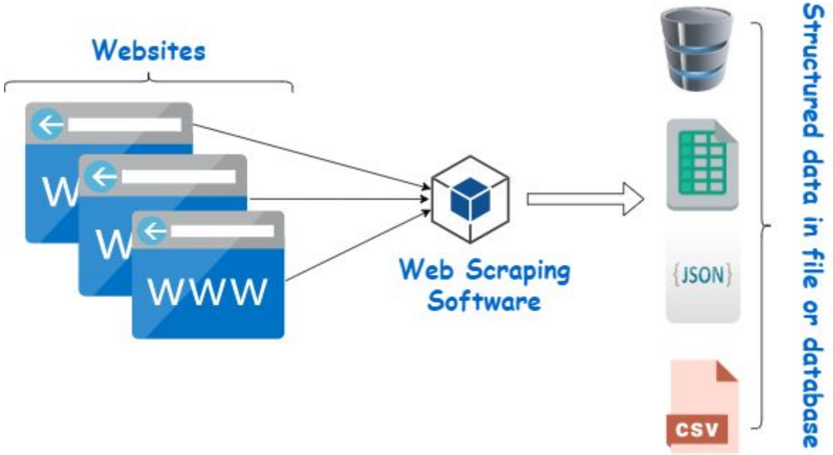
Single Endpoint Interaction

Precise Data Control

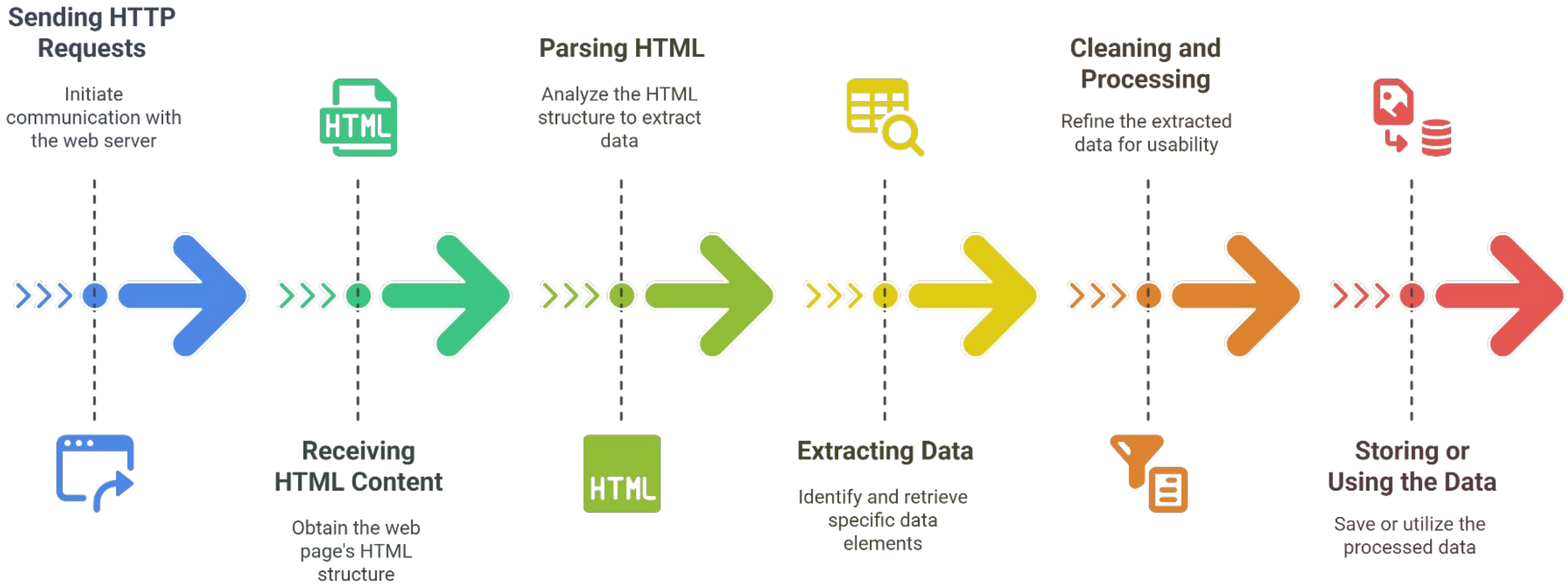
GraphQL

Web Scraping

- Web scraping is the process of *extracting data from websites* by sending HTTP requests, retrieving the web pages' HTML content, and parsing that content to extract the specific information you're interested in.
- Gather data from websites and use it for various purposes such as analysis, research, monitoring, or integration with other systems.



Web Scraping Stages



Sending HTTP Requests

Initiate communication with the web server

Receiving HTML Content

Obtain the web page's HTML structure

Parsing HTML

Analyze the HTML structure to extract data

Extracting Data

Identify and retrieve specific data elements

Cleaning and Processing

Refine the extracted data for usability

Storing or Using the Data

Save or utilize the processed data

Made with Napkin

Web Scraping Stages

1. **Sending HTTP Requests**: Send an HTTP request to the web server that hosts the page. The request might include headers (user-agent, cookies, etc.) and optional parameters that simulate a web browser's behavior.

2. **Receiving HTML Content**: The web server responds to your request by sending back the HTML content of the web page. It contains the structure, text, and elements of the page.

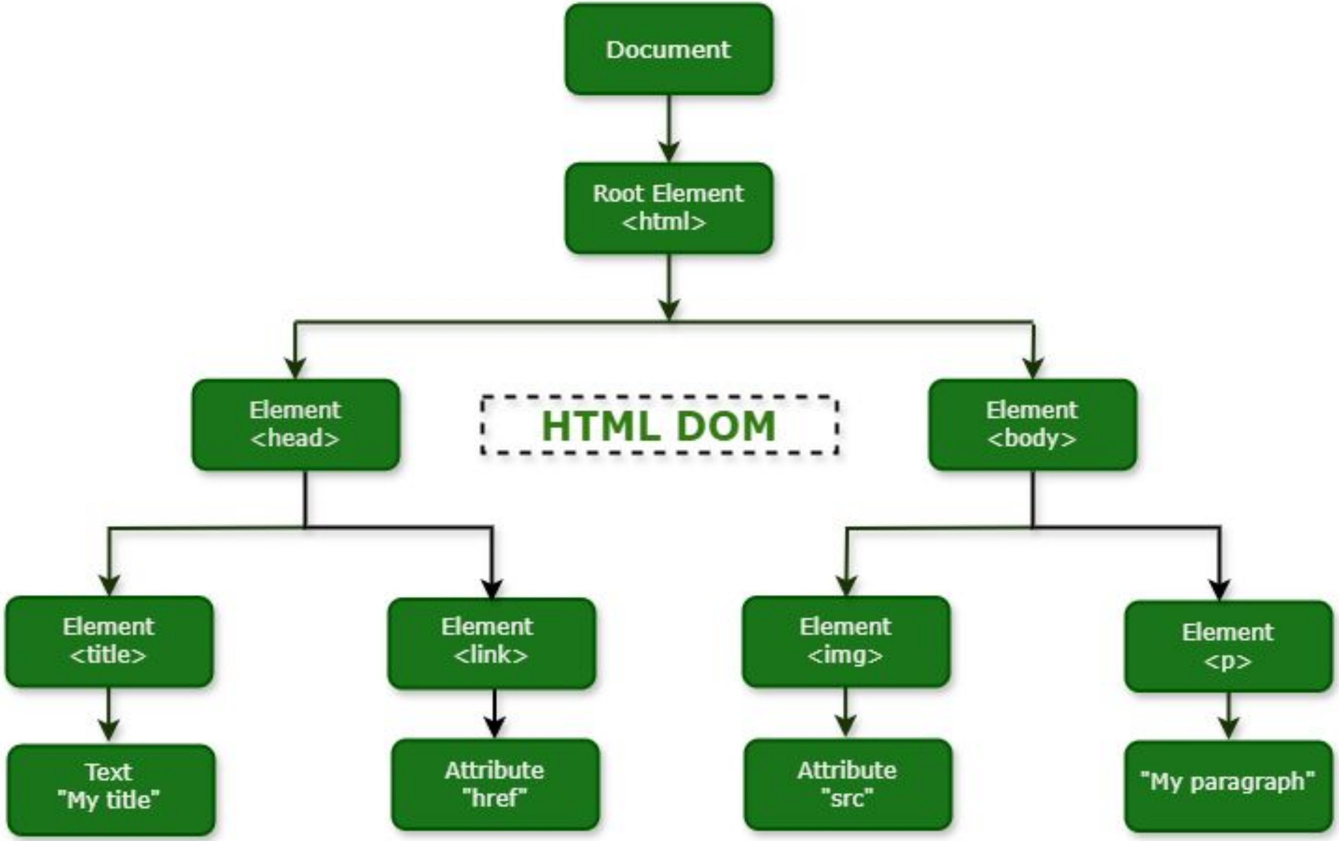
3. **Parsing HTML**: Once you have the HTML content, use a parsing library or tool to navigate through the structure of the HTML and extract the relevant data.

4. **Extracting Data**: Within the HTML structure, identify the specific elements that contain the data you need, such as text, links, images, tables, etc. Use the parsing library's methods to locate and extract these elements.

5. **Cleaning and Processing**: The data might contain unwanted tags, formatting, or noise. Clean and process the data to ensure that you only have the required information in a usable format.

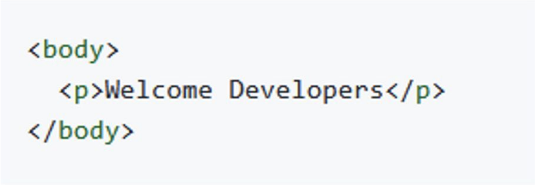
6. **Storing or Using the Data**: After extracting and cleaning the data, store it in a database or a file. It can be used for any downstream applications.

HTML Document Object Model



https://www.tutorialspoint.com/html/html_dom.htm

HTML



```
<body>
  <p>Welcome Developers</p>
</body>
```

Basic structure of a HTML

- The <body> of a HTML page contains the document’s content.
- Tags are containers to hold information. Each tag will have a left angle bracket (<) and a right angle bracket (>).
  - <p> is a tag for a paragraph of text.
- Element - Elements usually consist of an opening and closing tag. It represents some content or information to the browser.
  - Element with opening and closing tags: `<p>hello</p>`
  - Element with self closing tags: `<br />`
  - The ending slash is optional, so `<br />` can be typed as `<br>`

HTML Attributes

- All HTML elements can have attributes
- Attributes provide additional information about the elements
- Attributes are specified in the starting tag of the element
- Usually, they come in name/value pairs, like `name=“value”`, where name is the name of the attribute

Example:

```
<a href=”www.google.com”>Visit Google</a>
```

`<a>` is a hyperlink tag, `href` is an attribute with the URL value `“www.google.com”`

```
<img src=”image.jpg”>
```

`<img>` is an image tag, `src` is an attribute with value `“image.jpg”`, which specifies the path to the image to be displayed.

HTML Id and Class

- Class and id are two attributes that can be used to identify elements.
- They will be very useful when we learn CSS and JavaScript.

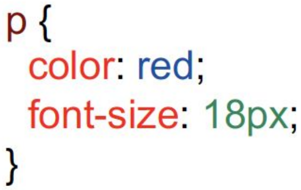
```
<div id=“top”>

<p class=“info"> Tesla stock went up yesterday </p>

<p class=“info”> Tesla stock went down today </p>

</div>
```

- **Classes** are usually used to identify *more than one element* but **ids** are used to *identify one unique element*.

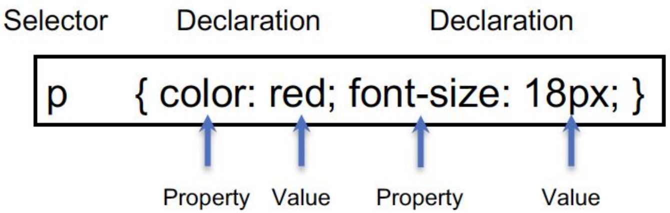
- You may only use the *same id attribute value once* per html page.

CSS

- We use CSS to style elements of HTML, the example above sets the colour of the previous paragraph to red and font size to 18px.



```
p {
  color: red;
  font-size: 18px;
}
```



Selector Declaration Declaration

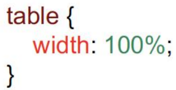
```
p    { color: red; font-size: 18px; }
```

Property Value Property Value

CSS Selectors

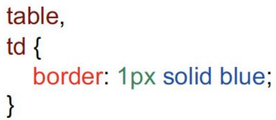
- The first example selects the *elements* with the `<table>` tag.
- The second example selects all the *elements* with `<table>` and `<td>` tags.
- Use the *greater than* sign (`>`) selector in CSS to select a child element with a specific parent.
- In the third example, we select `<th>` with a parent `<tr>` and that `<tr>` should have a parent `<thead>`.
- Having a *space* instead of `>` means that it will select any `<th>` inside `<thead>` and `<tr>` might not be a direct child element, it can have any other elements in between.

1



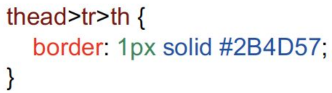
```
table {
   width: 100%;
}
```

2



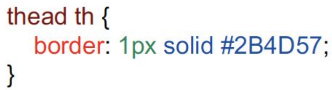
```
table,
td {
   border: 1px solid blue;
}
```

3



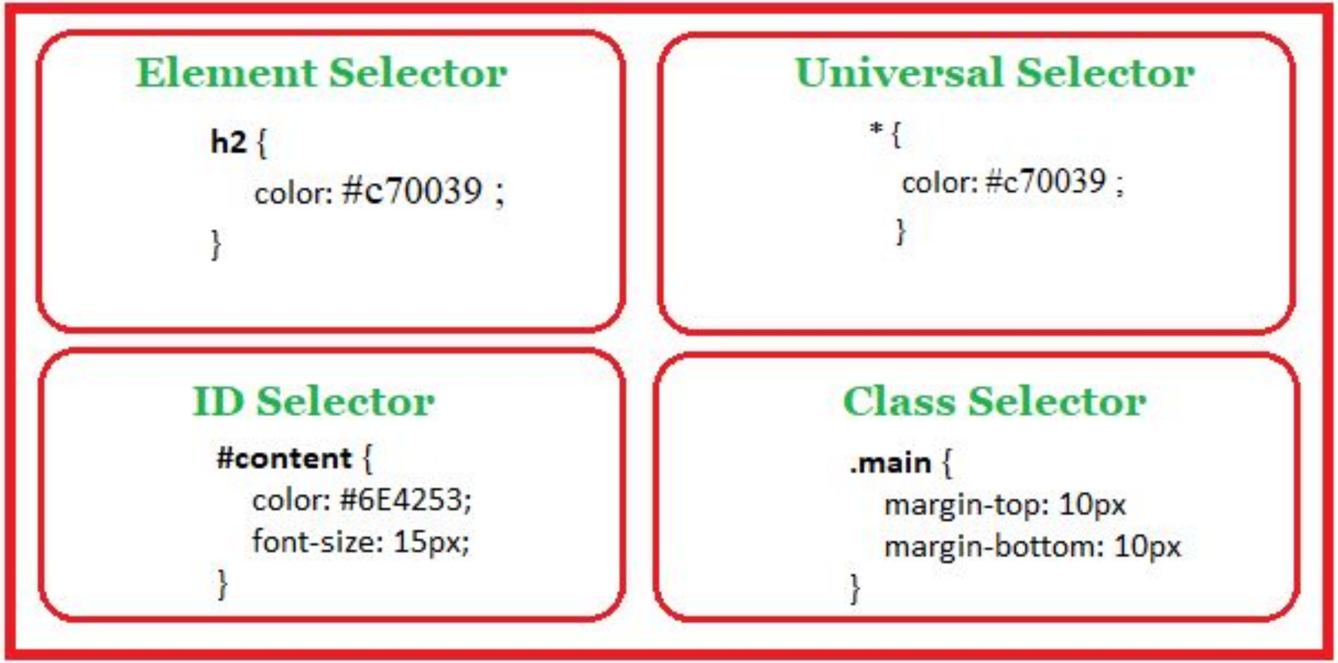
```
thead>tr>th {
   border: 1px solid #2B4D57;
}
```

Alternatively:



```
thead th {
   border: 1px solid #2B4D57;
}
```

CSS Selectors



Element Selector

```
h2 {
    color: #c70039 ;
}
```

Universal Selector

```
* {
    color: #c70039 ;
}
```

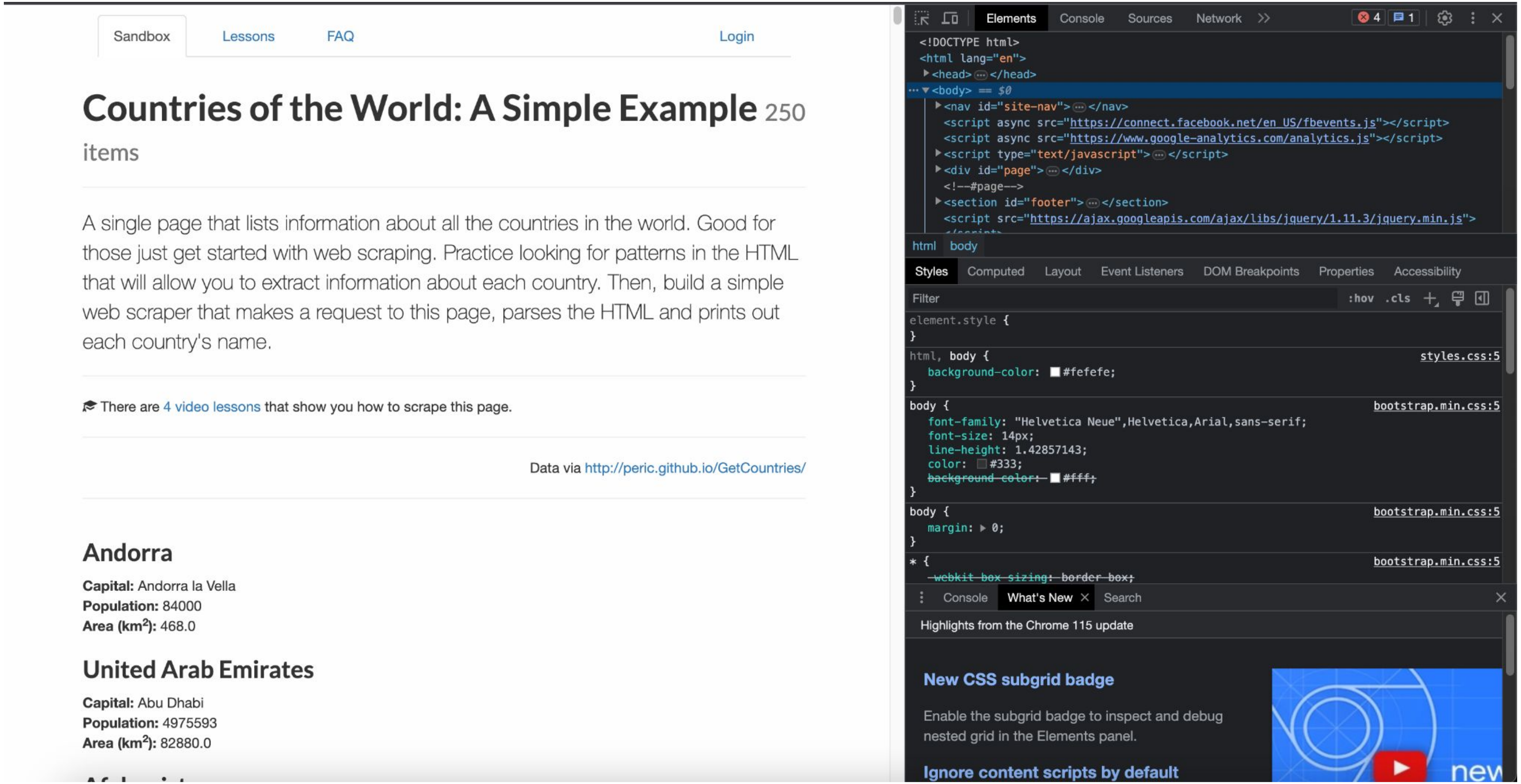
ID Selector

```
#content {
    color: #6E4253;
    font-size: 15px;
}
```

Class Selector

```
.main {
    margin-top: 10px
    margin-bottom: 10px
}
```

Browser Developer Tools Demo



Countries of the World: A Simple Example 250 items

A single page that lists information about all the countries in the world. Good for those just get started with web scraping. Practice looking for patterns in the HTML that will allow you to extract information about each country. Then, build a simple web scraper that makes a request to this page, parses the HTML and prints out each country's name.

There are 4 video lessons that show you how to scrape this page.

Data via http://peric.github.io/GetCountries/

Andorra

Capital: Andorra la Vella
Population: 84000
Area (km^2): 468.0

United Arab Emirates

Capital: Abu Dhabi
Population: 4975593
Area (km^2): 82880.0

```
<!DOCTYPE html>
<html lang="en">
 ▶<head> ⋯ </head>
⋯▼<body> == $0
   ▶<nav id="site-nav"> ⋯ </nav>
    <script async src="https://connect.facebook.net/en_US/fbevents.js"></script>
    <script async src="https://www.google-analytics.com/analytics.js"></script>
   ▶<script type="text/javascript"> ⋯ </script>
   ▶<div id="page"> ⋯ </div>
    <!--#page-->
   ▶<section id="footer"> ⋯ </section>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js">
```

```
element.style {
}
html, body {                                                    styles.css:5
  background-color: #fefefe;
}
body {                                                   bootstrap.min.css:5
  font-family: "Helvetica Neue",Helvetica,Arial,sans-serif;
  font-size: 14px;
  line-height: 1.42857143;
  color: #333;
  ~~background-color: #fff;~~
}
body {                                                   bootstrap.min.css:5
  margin: ▶ 0;
}
* {                                                      bootstrap.min.css:5
  ~~-webkit-box-sizing: border-box;~~
```

Highlights from the Chrome 115 update

New CSS subgrid badge

Enable the subgrid badge to inspect and debug nested grid in the Elements panel.

Ignore content scripts by default

End of Lesson - Exit Ticket

Survey Link

https://www.menti.com