CUI Abbottabad

Department of Computer Science

Web Technologies

Lecture -11-[CLO-2]

Javascript-Part7

COMSATS University Islamabad, Abbottabad Campus

Agenda

- Web Storage
- Cookies
- Session Storage
- Local Storage
- Twitter Search example
- Introduction to JSON
  - JSON Data types
  - Representation of data in JSON according to data types
  - Example programs

INTRODUCTION

- In this lecture we use HTML5’s web storage capabilities to create a web application that stores a user’s favorite Twitter searches on the computer for easy access at a later time.
- We also provide a brief introduction to JSON, a means for creating JavaScript objects—typically for transferring data over the Internet between client-side and server-side programs.

HTML WEB STORAGE--COOKIES

- Before HTML5, websites could store only small amounts of text-based information on a *user's* computer using **cookies**.
- A **cookie** is a *key/value pair* in which each *key* has a corresponding *value*. The key and value are both strings.
- Cookies are stored by the browser on the *user's* computer to maintain client-specific information during and between browser sessions.
- A website might use a cookie to record user preferences or other information that it can retrieve during the client's subsequent visits. For example, a website can retrieve the user's name from a cookie and use it to display a personalized greeting.
- Similarly, many websites used cookies during a browsing session to track user-specific information, such as the contents of an online shopping cart.

HTML WEB STORAGE--COOKIES

- When a user visits a website, the browser locates any cookies written by that website and sends them to the server.
- Cookies may be accessed only by the web server and scripts of the website from which the cookies originated (i.e., a cookie set by a script on amazon.com can be read only by amazon.com servers and scripts). The browser sends these cookies with *every* request to the server.

PROBLEMS WITH COOKIES

There are several problems with cookies :

- One is that they're extremely limited in size. Today's web apps often allow users to manipulate large amounts of data such as:
  - documents
  - thousands of emails.
- Some web applications allow so-called *offline access*—for example, a word-processing web application might allow a user to access documents locally, even when the computer is not connected to the Internet. Cookies cannot store entire documents.

PROBLEMS WITH COOKIES

- Another problem is that a user often opens many tabs in the same browser window.
- If the user browses the same site from multiple tabs, all of the site’s cookies are shared by the pages in each tab. This could be problematic in web applications that allow the user to purchase items. For example, if the user is purchasing different items in each tab, with cookies it’s possible that the user could accidentally purchase the same item twice.

LOCAL STORAGE AND SESSION STORAGE

As of HTML5, there are **two new mechanisms for storing key/value pairs** that help eliminate some of the problems with cookies.

1. localStorage

- Web applications can use the window object's **localStorage property** to store up to several megabytes of key/value-pair string data on the user's computer and can access:
  - **that data across browsing sessions and browser tabs.**
- Unlike cookies, data in the localStorage object is not sent to the web server with each request.
- Each website domain (such as deitel.com or google.com) has a separate localStorage object —all the pages from a given domain share one localStorage object.

LOCAL STORAGE AND SESSION STORAGE

- Typically,5MB are reserved for each localStorage object, but a web browser can ask the user if more space should be allocated when the space is full.

2. sessionStorage

- Web applications that need access to data for *only* a browsing session and that must keep that data *separate* among multiple tabs can use the window object's **sessionStorage property**.
- There's a separate sessionStorage object for every browsing session, including separate tabs that are accessing the same website.

TWITTER SEARCHES APP USING LOCALSTORAGE AND SESSIONSTORAGE

To demonstrate these new HTML5 storage capabilities, we'll implement a **Favorite Twitter Searches** app.

- Twitter's search feature is a great way to follow trends and see what people are saying about specific topics.
- The app we present here allows users to save their favorite(possibly lengthy) Twitter search strings with easy-to-remember, user-chosen, short tag names.
- Users can then conveniently follow the tweets on their favorite topics by visiting this web page and clicking the link for a saved search.

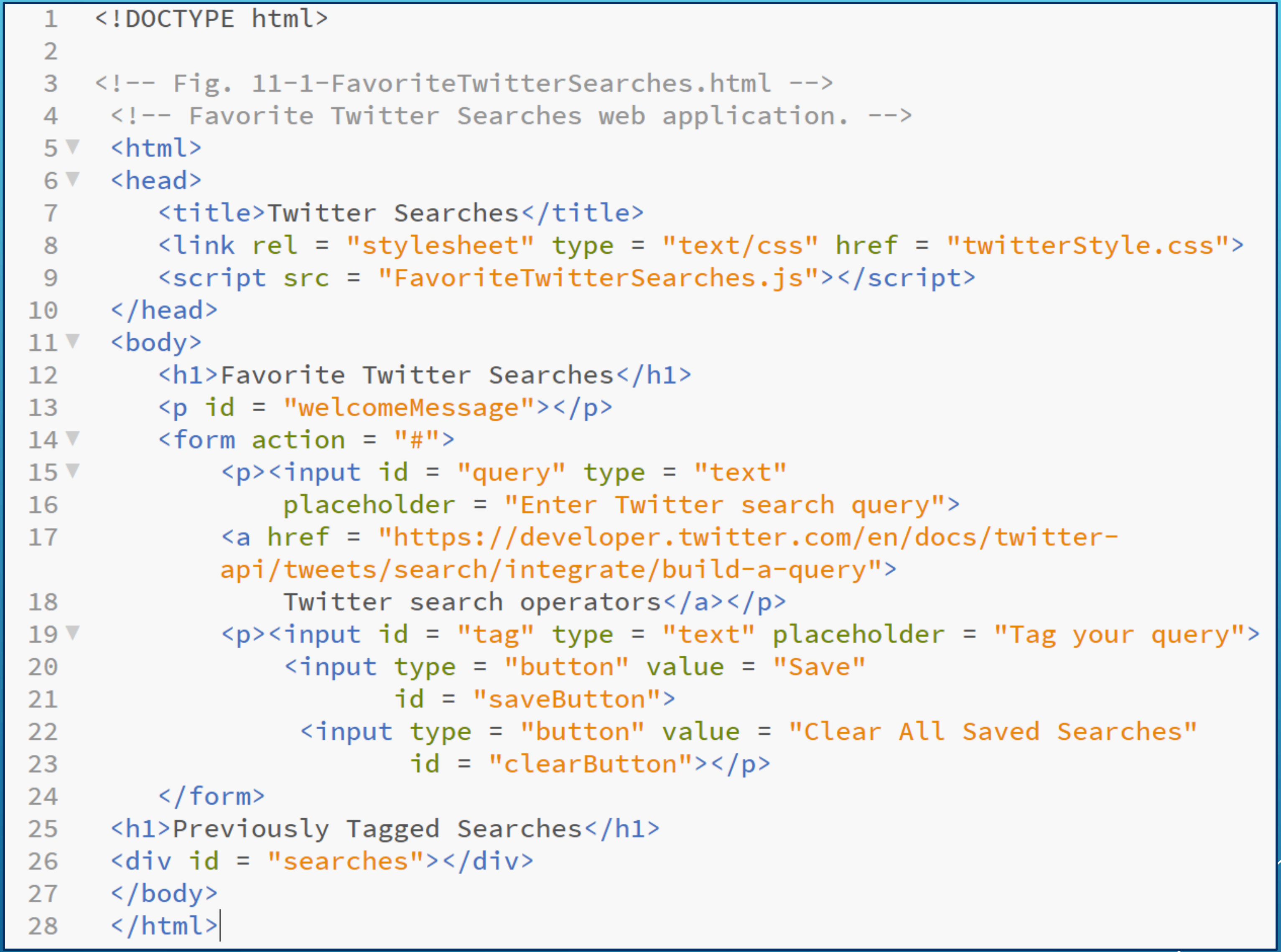
TWITTER SEARCHES APP USING LOCALSTORAGE AND SESSIONSTORAGE

- Twitter search queries can be finely tuned using Twitter's search operators (https://developer.twitter.com/en/docs/twitter-api/tweets/search/integrate/build-a-query)—but more complex queries are lengthy, time consuming and error prone to type.
- The user's favorite searches are saved using localStorage, so they're immediately available each time the user browses the app's web page.
- In the next example has three files
  - Listing-11-1-FavoriteTwitterSearches.html
  - Listing-11-2-twitterStyle.css file
  - Listing-11-3-FavoriteTwitterSearches.js file

FAVORITETWITTERSEARCHES.HTML FILE

- This HTML5 document provides a form (lines 14–24) that allows the user to enter new searches.
- Previously tagged searches are displayed in the div named searches (line 26).

FAVORITETWITTERSEARCHS.HTML



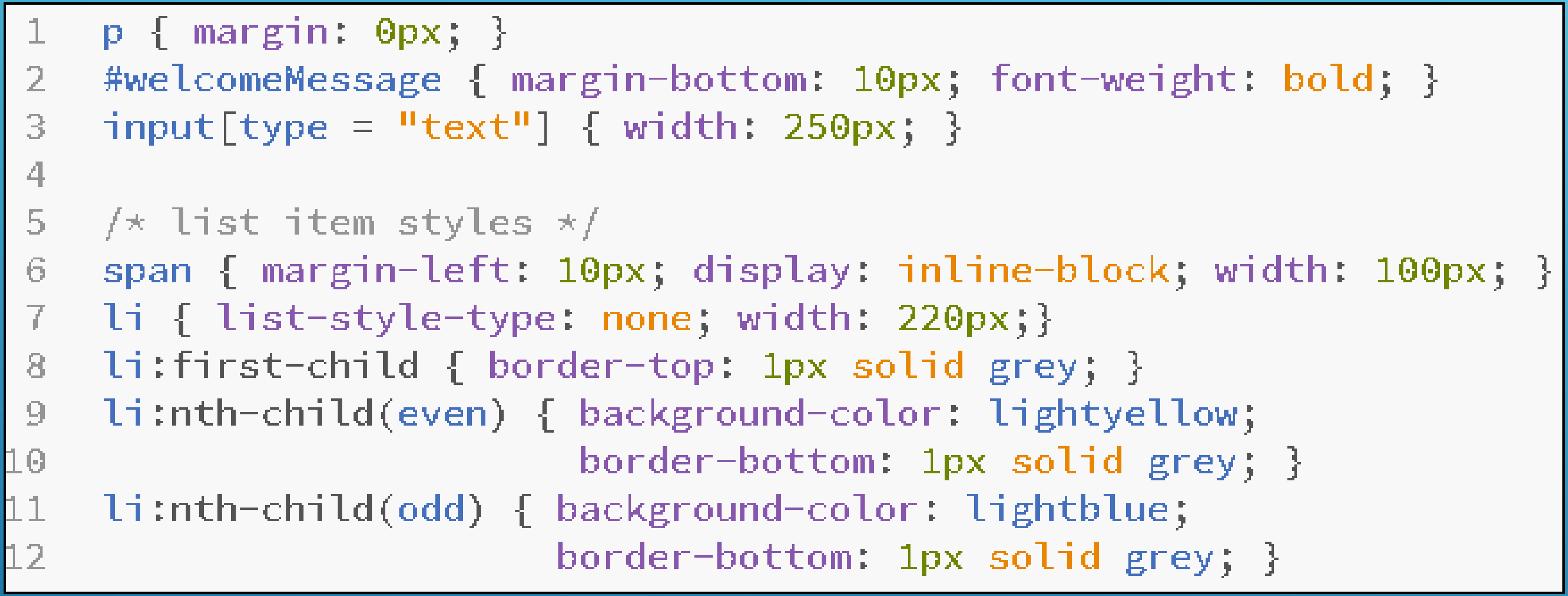
```html
<!DOCTYPE html>

<!-- Fig. 11-1-FavoriteTwitterSearches.html -->
 <!-- Favorite Twitter Searches web application. -->
 <html>
 <head>
    <title>Twitter Searches</title>
    <link rel = "stylesheet" type = "text/css" href = "twitterStyle.css">
    <script src = "FavoriteTwitterSearches.js"></script>
 </head>
 <body>
    <h1>Favorite Twitter Searches</h1>
    <p id = "welcomeMessage"></p>
    <form action = "#">
        <p><input id = "query" type = "text"
            placeholder = "Enter Twitter search query">
        <a href = "https://developer.twitter.com/en/docs/twitter-
        api/tweets/search/integrate/build-a-query">
            Twitter search operators</a></p>
        <p><input id = "tag" type = "text" placeholder = "Tag your query">
            <input type = "button" value = "Save"
                   id = "saveButton">
             <input type = "button" value = "Clear All Saved Searches"
                   id = "clearButton"></p>
    </form>
 <h1>Previously Tagged Searches</h1>
 <div id = "searches"></div>
 </body>
 </html>
```

Listing-11-1-FavoriteTwitterSearches.html

CSS FOR FAVORITE TWITTER SEARCHES

- Figure 11-2 contains the CSS styles for this app.
- Line 3 uses a CSS3 attribute selector to select all input elements that have the type "text" and sets their width to 250px.
- Each link that represents a saved search is displayed in a span that has a fixed width (line 6).
- To specify the width, we set the display property of the spans to inline-block.
- Line 8 specifies a **:first-child selector** that's used to select the first list item in the unordered list of saved searches that's displayed at the bottom of the web page.
- Lines 9–10 and 11–12 use **:nth-child selectors** to specify the styles of the odd (first, third, fifth, etc.) and even (second,fourth, sixth, etc.) list items, respectively.
- We use these selectors or alternate the background colors of the saved searches

TWITTERSTYLE.CSS FILE



```css
p { margin: 0px; }
#welcomeMessage { margin-bottom: 10px; font-weight: bold; }
input[type = "text"] { width: 250px; }

/* list item styles */
span { margin-left: 10px; display: inline-block; width: 100px; }
li { list-style-type: none; width: 220px;}
li:first-child { border-top: 1px solid grey; }
li:nth-child(even) { background-color: lightyellow;
                     border-bottom: 1px solid grey; }
li:nth-child(odd) { background-color: lightblue;
                    border-bottom: 1px solid grey; }
```

Listing-11-2-twitterStyle.css

JAVASCRIPT FOR FAVORITE TWITTER SEARCHES

- Figure 11-3 presents the JavaScript for the **Favorite Twitter Searches** app.
- When the HTML5 document in Fig. 11-1 loads, function start (lines 80–87) is called to register event handlers and call function loadSearches (lines 7–44).
- Line 9 uses the sessionStorage object to determine whether the user has already visited the page during this browsing session.
- The **getItem method** receives a name of a key as an argument. If the key exists, the method returns the corresponding string value; otherwise, it returns null.
- If this is the user’s first visit to the page during this browsing session, line 11 uses the **setItem method** to set the key "herePreviously" to the string "true", then lines 12–13 display a welcome message in the welcomeMessage paragraph element.

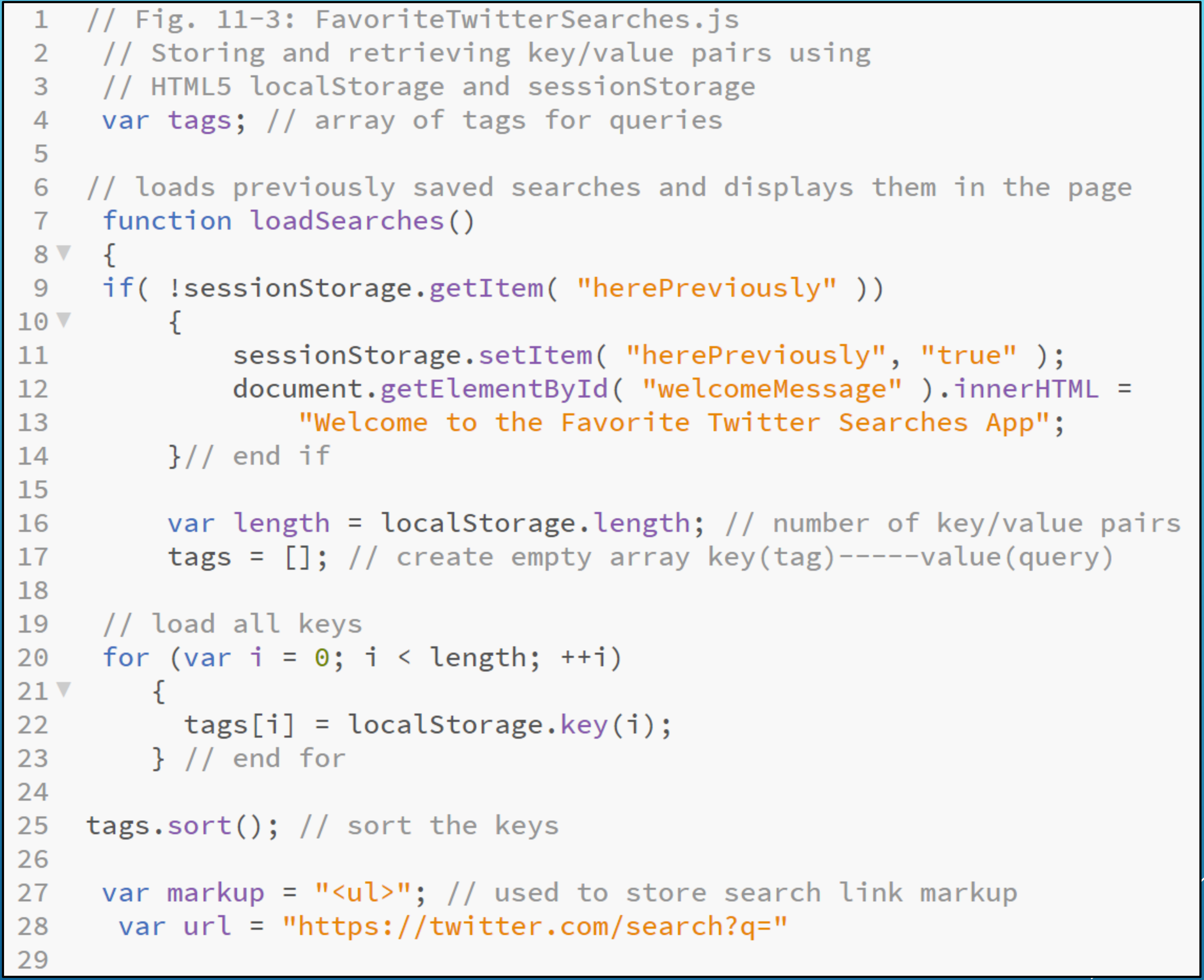
JAVASCRIPT FOR FAVORITE TWITTER SEARCHES

- Next, line 16 gets the localStorage object’s **length**, which represents the number of key/value pairs stored.
- Line-17 creates an array and assigns it to the script variable tags, then
- Lines 20–23 get the keys from the localStorage object and store them in the tags array.
- Method **key** (line 22) receives an index as an argument and returns the corresponding key.
- Line 25 sorts the tags array, so that we can display the searches in alphabetical order by tag name (i.e., key).
- Lines 27–42 build the unordered list of links representing the saved searches.
- Line 33 calls the local-Storage object’s getItem method to obtain the search string for a given tag and appends the search string to the Twitter search URL (line 28).

JAVASCRIPT FOR FAVORITE TWITTER SEARCHES

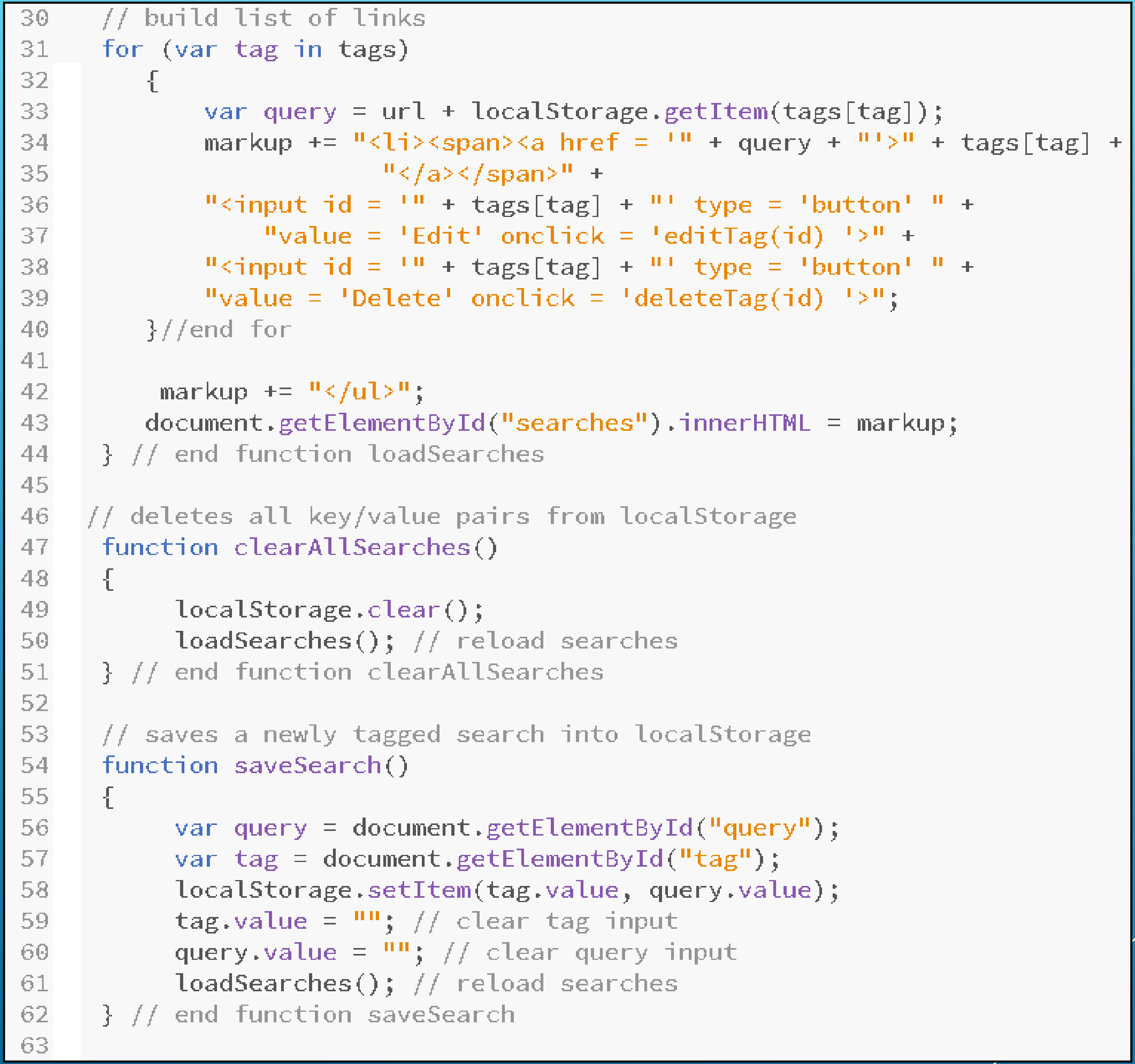
- Notice that, for simplicity, lines 37and 38 use the onclick attributes of the dynamically generated **Edit** and **Delete** buttons to set the buttons' event handlers—this is an older mechanism for registering event handlers.
- To register these with the elements' addEventListener method, we'd have to dynamically locate the buttons in the page after we've created them, then register the event handlers, which would require significant additional code.
- Separately, notice that each event handler is receiving the button input element's id as an argument—this enables the event handler to use the id value when handling the event.
- **[*Note*:** The localStorage and sessionStorage properties and methods we discuss throughout this section apply to both objects.]

JAVASCRIPT FILE



```
// Fig. 11-3: FavoriteTwitterSearches.js
 // Storing and retrieving key/value pairs using
 // HTML5 localStorage and sessionStorage
 var tags; // array of tags for queries

// loads previously saved searches and displays them in the page
 function loadSearches()
 {
 if( !sessionStorage.getItem( "herePreviously" ))
     {
         sessionStorage.setItem( "herePreviously", "true" );
         document.getElementById( "welcomeMessage" ).innerHTML =
             "Welcome to the Favorite Twitter Searches App";
     }// end if

     var length = localStorage.length; // number of key/value pairs
     tags = []; // create empty array key(tag)-----value(query)

 // load all keys
 for (var i = 0; i < length; ++i)
    {
      tags[i] = localStorage.key(i);
    } // end for

tags.sort(); // sort the keys

 var markup = "<ul>"; // used to store search link markup
  var url = "https://twitter.com/search?q="

```

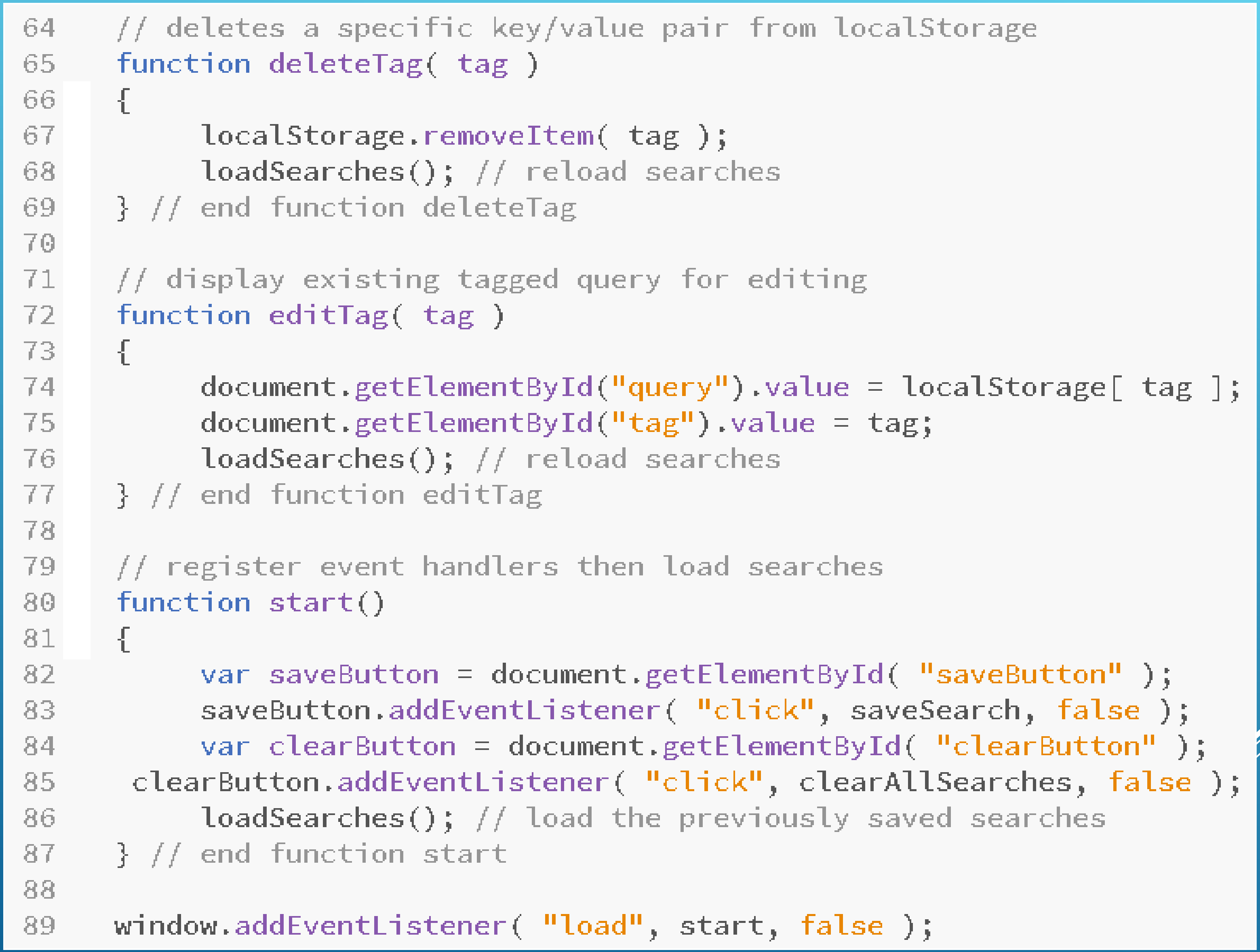
Listing-11-3-FavoriteTwitterSearchs.js



```
 // build list of links
 for (var tag in tags)
    {
        var query = url + localStorage.getItem(tags[tag]);
        markup += "<li><span><a href = '" + query + "'>" + tags[tag] +
                    "</a></span>" +
        "<input id = '" + tags[tag] + "' type = 'button' " +
            "value = 'Edit' onclick = 'editTag(id) '>" +
        "<input id = '" + tags[tag] + "' type = 'button' " +
        "value = 'Delete' onclick = 'deleteTag(id) '>";
    }//end for

     markup += "</ul>";
    document.getElementById("searches").innerHTML = markup;
 } // end function loadSearches

// deletes all key/value pairs from localStorage
 function clearAllSearches()
 {
      localStorage.clear();
      loadSearches(); // reload searches
 } // end function clearAllSearches

 // saves a newly tagged search into localStorage
 function saveSearch()
 {
      var query = document.getElementById("query");
      var tag = document.getElementById("tag");
      localStorage.setItem(tag.value, query.value);
      tag.value = ""; // clear tag input
      query.value = ""; // clear query input
      loadSearches(); // reload searches
 } // end function saveSearch
```

Listing-11-3-FavoriteTwitterSearchs.js

JAVASCRIPT FILE



```
// deletes a specific key/value pair from localStorage
function deleteTag( tag )
{
     localStorage.removeItem( tag );
     loadSearches(); // reload searches
} // end function deleteTag

// display existing tagged query for editing
function editTag( tag )
{
     document.getElementById("query").value = localStorage[ tag ];
     document.getElementById("tag").value = tag;
     loadSearches(); // reload searches
} // end function editTag

// register event handlers then load searches
function start()
{
     var saveButton = document.getElementById( "saveButton" );
     saveButton.addEventListener( "click", saveSearch, false );
     var clearButton = document.getElementById( "clearButton" );
 clearButton.addEventListener( "click", clearAllSearches, false );
     loadSearches(); // load the previously saved searches
} // end function start

window.addEventListener( "load", start, false );
```

Listing-11-3-FavoriteTwitterSearches.js

JAVASCRIPT FOR FAVORITE TWITTER SEARCHES

- Function clearAllSearches (lines 47–51) is called when the user clicks the **Clear All Saved Searches** button.
- The **clear method** of the localStorage object (line 49) removes all key/value pairs from the object.
- We then call loadSearches to refresh the list of saved searches in the web page.
- Function saveSearch (lines 54–62) is called when the user clicks **Save** to save a search.
- Line 58 uses the setItem method to store a key/value pair in the localStorage object. If the key already exits, setItem replaces the corresponding value; otherwise, it creates a new key/value pair.
- We then call loadSearches to refresh the list of saved searches in the web page.

JAVASCRIPT FOR FAVORITE TWITTER SEARCHES

- Function deleteTag (lines 65–69) is called when the user clicks the **Delete** button next to a particular search. The function receives the tag representing the key/value pair todelete, which we set in line 38 as the button’s id.
- Line 67 uses the **removeItem method** toremove a key/value pair from the localStorage object.
- We then call loadSearches to refresh the list of saved searches in the web page.

JAVASCRIPT FOR FAVORITE TWITTER SEARCHES

- Function editTag (lines 72–77) is called when the user clicks the **Edit** button next to a particular search. The function receives the tag representing the key/value pair to edit, which we set in line 36 as the button’s id. In this case, we display the corresponding key/value pair’s contents in the input elements with the ids "tag" and "query", respectively, so the user can edit them.
- Line 74 uses the [] operator to access the value for a specified key (tag)—this performs the same task as calling getItem on the localStorage object.
- We then call loadSearches to refresh the list of saved searches in the web page

OUTPUT FAVORITETWITTERSEARCHS.HTML

- Figure-11-1(a) shows the app when it's loaded for the first time.
- The app uses sessionStorage to determine whether the user has visited the page previously during the current browsing session. If not, the app displays a welcome message.
- The user can save many searches and view them in alphabetical order.
- Search queries and their corresponding tags are entered in the text inputs at the top of the page.
- Clicking the **Save** button adds the new search to the favorites list.
- Clicking the link for a saved search requests the search page from Twitter's website, passing the user's saved search as an argument, and displays the search results in the web browser.

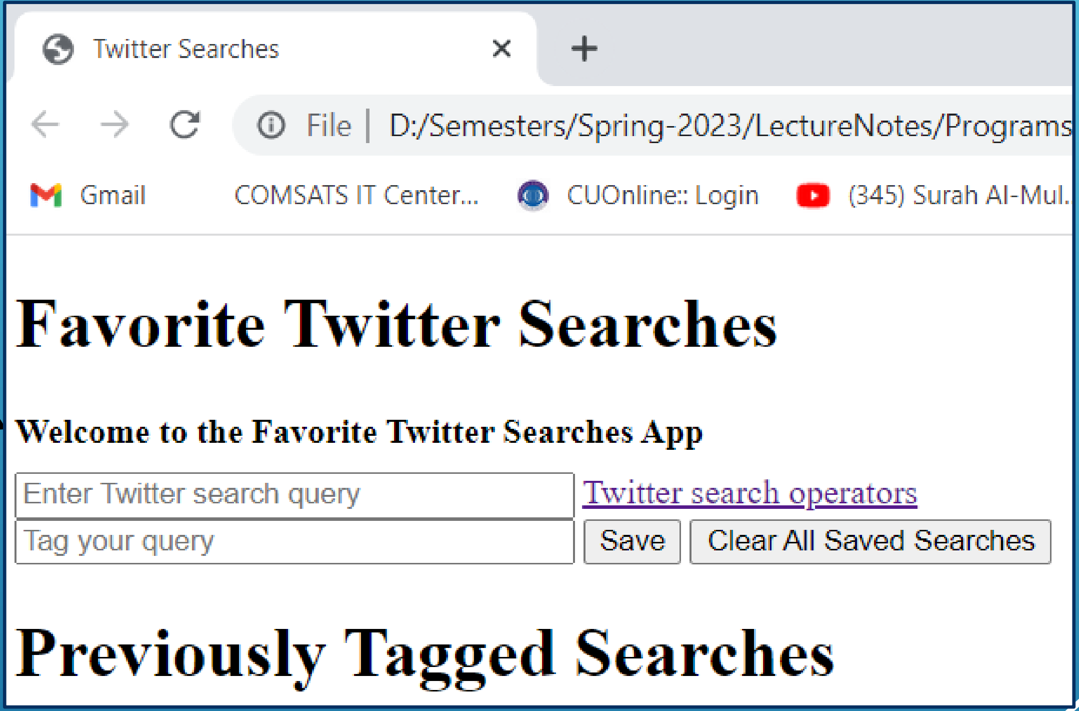
OUTPUT FAVORITETWITTERSEARCHS.HTML

(a)When app loaded for the first time in this browsing session and there are no tagged searches

Welcome message appears only on the first visit to the page during this browsing session

Enter twitter search query here

Tag your search



Favorite Twitter Searches

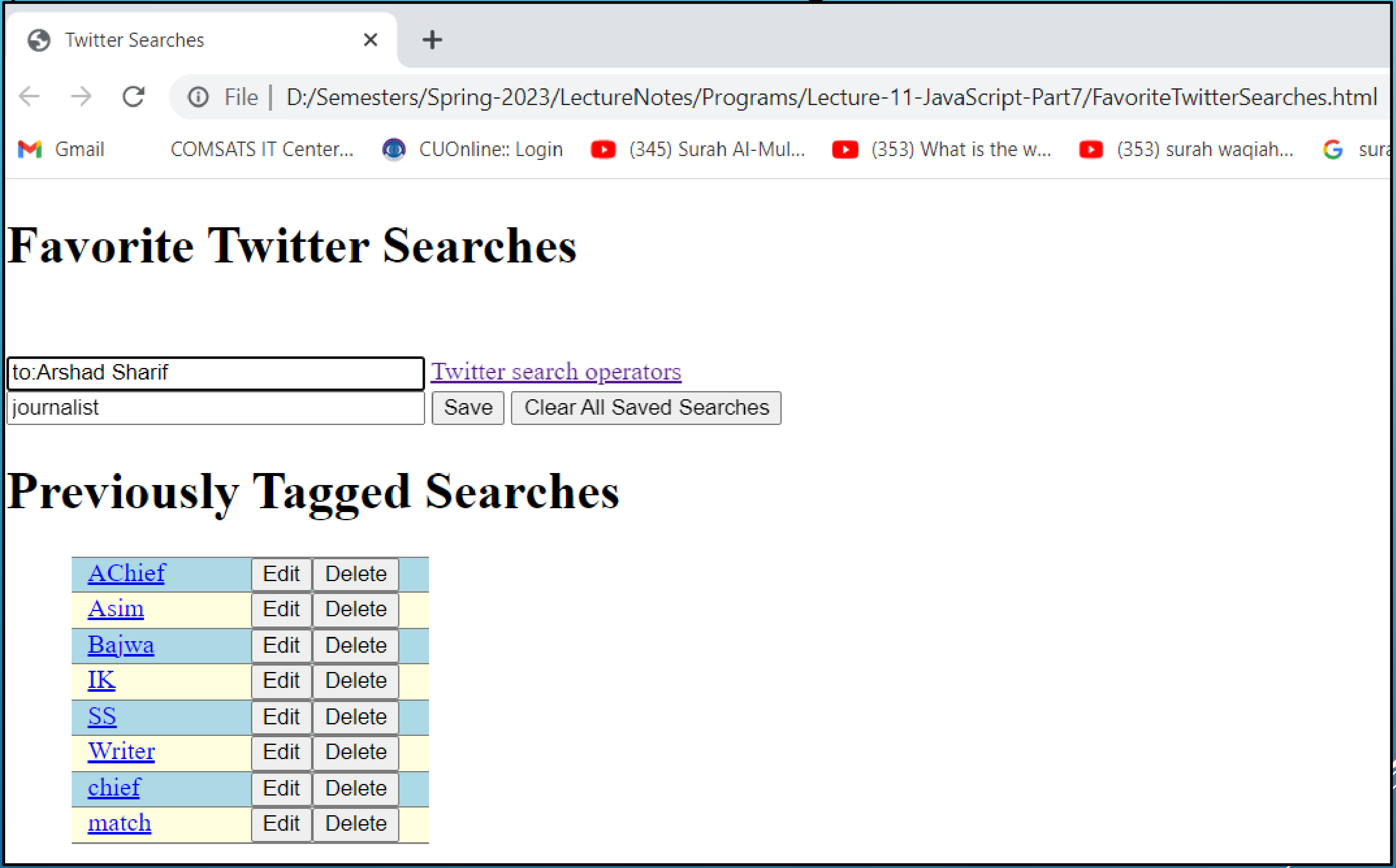
Welcome to the Favorite Twitter Searches App

Previously Tagged Searches

Listing-11-1(a)-Output-FavoriteTwitterSearchs.html

OUTPUT FAVORITETWITTERSEARCHS.HTML

(b)App with several saved searches and the user saving a new search


Twitter Searches

File | D:/Semesters/Spring-2023/LectureNotes/Programs/Lecture-11-JavaScript-Part7/FavoriteTwitterSearches.html

Gmail COMSATS IT Center... CUOnline:: Login (345) Surah Al-Mul... (353) What is the w... (353) surah waqiah... sura

Favorite Twitter Searches

to:Arshad Sharif [Twitter search operators]

journalist [Save] [Clear All Saved Searches]

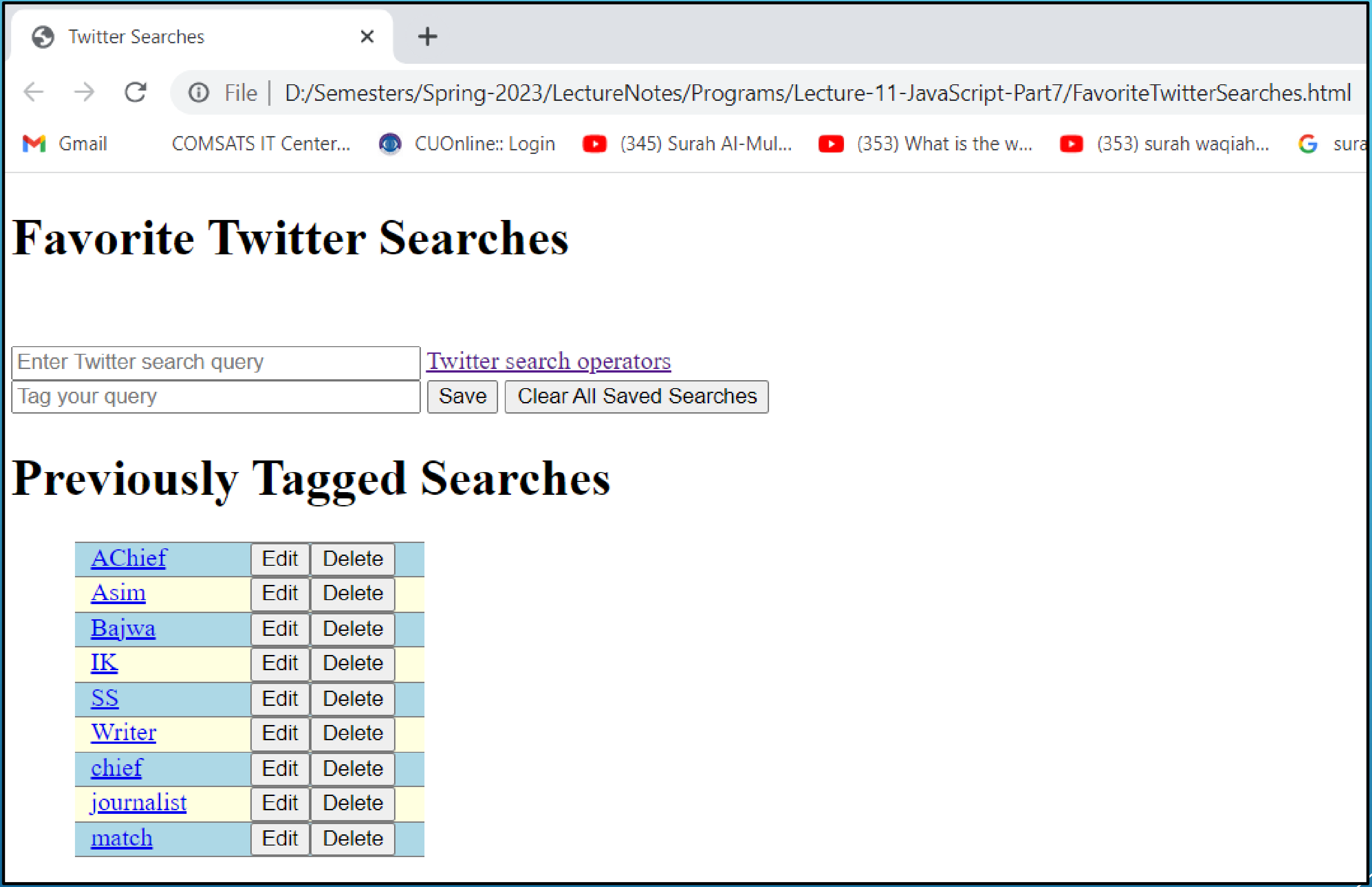
Previously Tagged Searches

AChief	Edit	Delete
Asim	Edit	Delete
Bajwa	Edit	Delete
IK	Edit	Delete
SS	Edit	Delete
Writer	Edit	Delete
chief	Edit	Delete
match	Edit	Delete

Listing-11-1(b)-Output-FavoriteTwitterSearchs.html

OUTPUT FAVORITETWITTERSEARCHS.HTML

(c)App after new search is saved—the user is about to click the Dietel search



Twitter Searches

File | D:/Semesters/Spring-2023/LectureNotes/Programs/Lecture-11-JavaScript-Part7/FavoriteTwitterSearches.html

Gmail   COMSATS IT Center...   CUOnline:: Login   (345) Surah Al-Mul...   (353) What is the w...   (353) surah waqiah...   sura

Favorite Twitter Searches

Enter Twitter search query   Twitter search operators

Tag your query   Save   Clear All Saved Searches

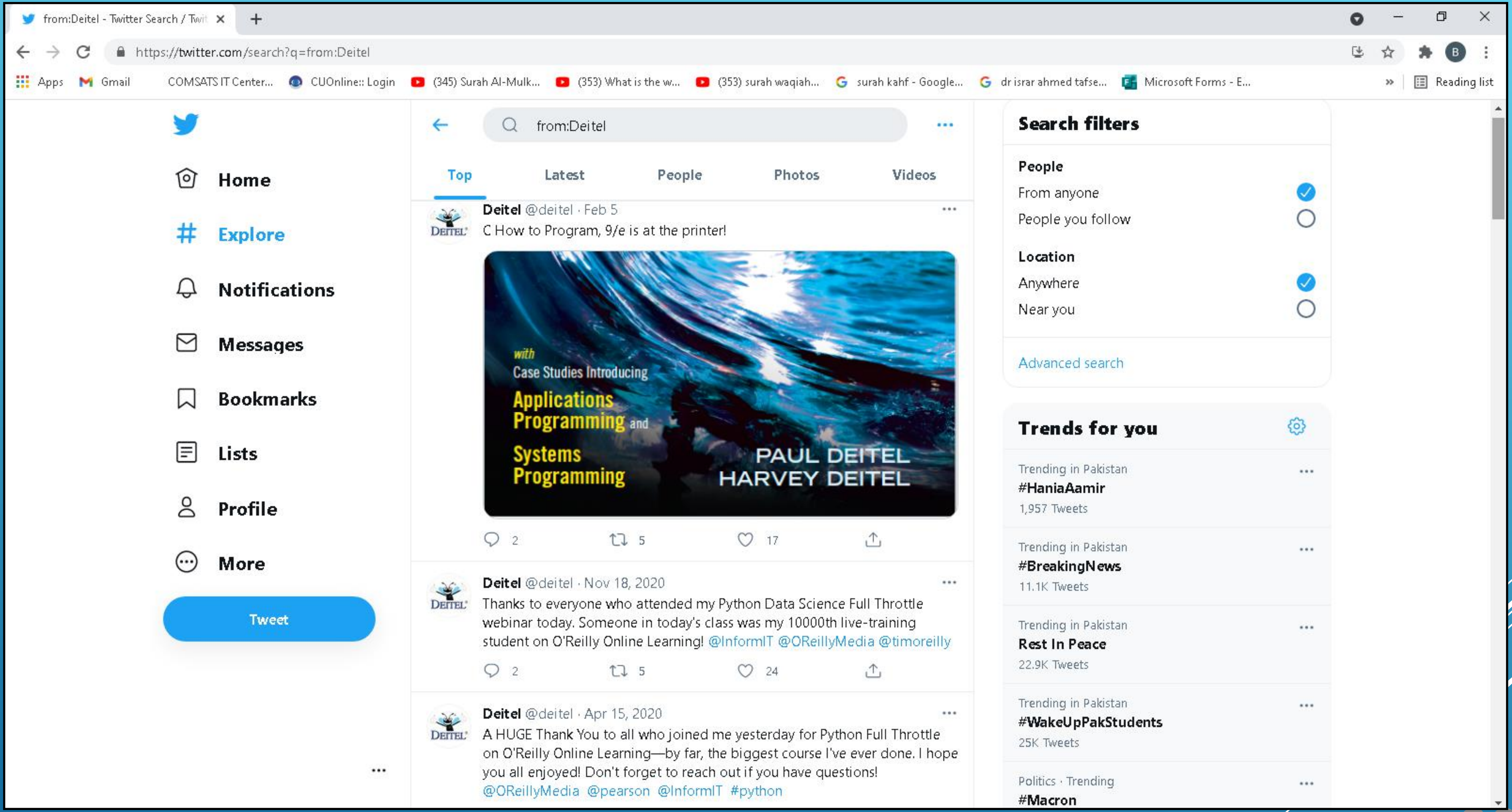
Previously Tagged Searches

AChief	Edit	Delete
Asim	Edit	Delete
Bajwa	Edit	Delete
IK	Edit	Delete
SS	Edit	Delete
Writer	Edit	Delete
chief	Edit	Delete
journalist	Edit	Delete
match	Edit	Delete

Listing-11-11(c)-Output-FavoriteTwitterSearchs.html

OUTPUT FAVORITETWITTERSEARCHS.HTML

Results of touching the Deitel link



Deitel @deitel · Feb 5
C How to Program, 9/e is at the printer!

2 5 17

Deitel @deitel · Nov 18, 2020
Thanks to everyone who attended my Python Data Science Full Throttle webinar today. Someone in today's class was my 10000th live-training student on O'Reilly Online Learning! @InformIT @OReillyMedia @timoreilly

2 5 24

Deitel @deitel · Apr 15, 2020
A HUGE Thank You to all who joined me yesterday for Python Full Throttle on O'Reilly Online Learning—by far, the biggest course I've ever done. I hope you all enjoyed! Don't forget to reach out if you have questions!
@OReillyMedia @pearson @InformIT #python

People
From anyone [✓]
People you follow

Location
Anywhere [✓]
Near you

Trending in Pakistan
#HaniaAamir
1,957 Tweets

Trending in Pakistan
#BreakingNews
11.1K Tweets

Trending in Pakistan
Rest In Peace
22.9K Tweets

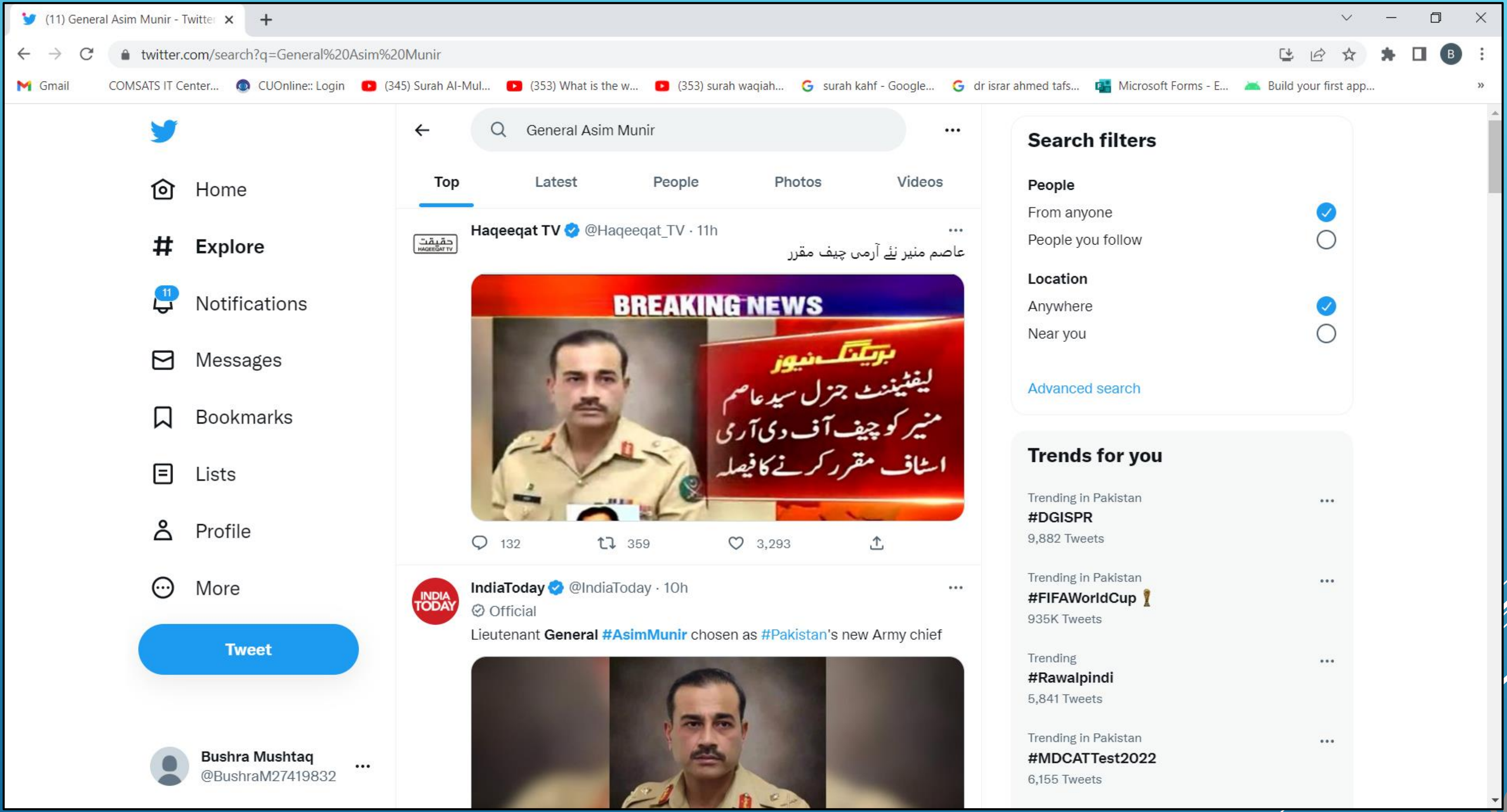
Trending in Pakistan
#WakeUpPakStudents
25K Tweets

Politics · Trending
#Macron

Listing-11-1(d)-Output-FavoriteTwitterSearchs.html

OUTPUT FAVORITETWITTERSEARCHS.HTML

Results of touching the AChief link



(11) General Asim Munir - Twitter

twitter.com/search?q=General%20Asim%20Munir

Haqeeqat TV @Haqeeqat_TV · 11h

عاصم منیر نئے آرمی چیف مقرر

BREAKING NEWS
بریکنگ نیوز
لیفٹیننٹ جنرل سید عاصم منیر کو چیف آف دی آرمی اسٹاف مقرر کرنے کا فیصلہ

132 359 3,293

IndiaToday @IndiaToday · 10h

Official

Lieutenant **General** #AsimMunir chosen as #Pakistan's new Army chief

Search filters

People

From anyone ✓

People you follow ○

Location

Anywhere ✓

Near you ○

Advanced search

Trends for you

Trending in Pakistan
#DGISPR
9,882 Tweets

Trending in Pakistan
#FIFAWorldCup
935K Tweets

Trending
#Rawalpindi
5,841 Tweets

Trending in Pakistan
#MDCATTest2022
6,155 Tweets

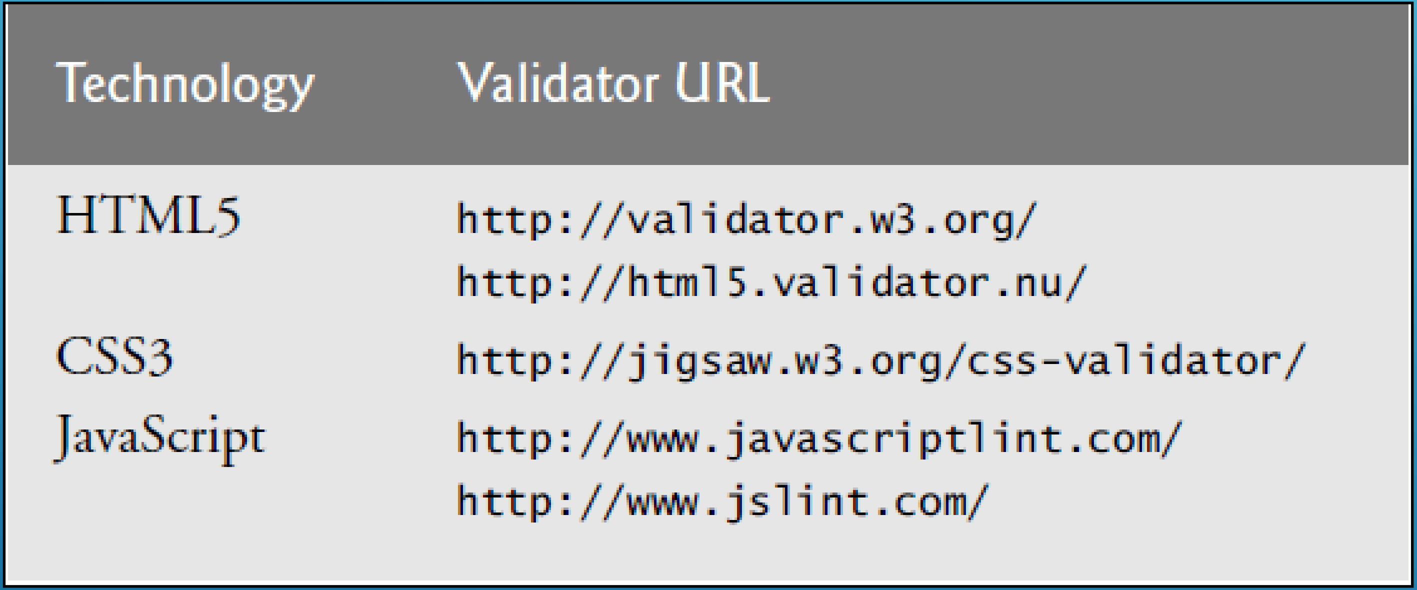
Listing-11-1(e)-Output-FavoriteTwitterSearchs.html

OUTPUT FAVORITETWITTERSEARCHS.HTML

- Figure-11-1(b) shows the app with several previously saved searches.
- Figure 11-1(c) shows the user entering a new search.
- Figure 23-11-1(d) shows the result of accessing the **Dietel** link, which searches for tweets from Dietel.
- Figure 23-11-1(e) shows the result of accessing the **AChief** link, which searches for tweets for Army Chief.
- You can edit the searches using the **Edit** buttons to the right of each search link. This enables you to tweak your searches for better results after you save them as favorites.
- Touching the **Clear All Saved Searches** button removes all the searches from the favorites list.
- Some browsers support localStorage and sessionStorage only for web pages that are downloaded from a web server, not for web pages that are loaded directly from the local file system. So, in that case we can post our app online for testing

VALIDATION OF HTML, CSS AND JAVASCRIPT CODE

- As you'll see, JavaScript programs typically have HTML5 and CSS3 portions as well.
- You must use proper HTML5, CSS3 and JavaScript syntax to ensure that browsers process your documents properly.
- Following figure lists the validators we used to validate the code



Technology	Validator URL
HTML5	`http://validator.w3.org/` `http://html5.validator.nu/`
CSS3	`http://jigsaw.w3.org/css-validator/`
JavaScript	`http://www.javascriptlint.com/` `http://www.jslint.com/`

USING JSON TO REPRESENT OBJECTS

In 1999, JSON (JavaScript Object Notation) is:

- **A Format for representing, storing and transporting data**
- a simple way to represent JavaScript objects as strings
- was introduced as an alternative to XML as a data-exchange technique.
- often used when data is sent from server to a web page
- Commonly used for APIs and Configs
- Lightweight data exchange format(used to create such format through which one language can communicate other language)
- Easy to read/write
- Language independent
- Self describing and easy to understand

USING JSON TO REPRESENT OBJECTS

- JSON is a superset over Javascript, which means everything you write in JSON is valid in Javascript.
- It also integrates easily with most languages
- Almost every single major language has some form of library or built-in functionality to parse JSON strings into objects and classes in that language, which makes working with JSON data is extremely easy inside of the programming language

DATA TYPES IN JSON

In JSON Each value can be of following types:

- a string
- number(integers, floats, signed/unsigned, exponential)
- Boolean
- Arrays
- null
- objects

USING JSON TO REPRESENT OBJECTS

- JSON has gained acclaim due to its simple format, making objects easy to read, create and parse.
- Each JSON object is represented as a list of property names and values contained in curly braces, in the following format:

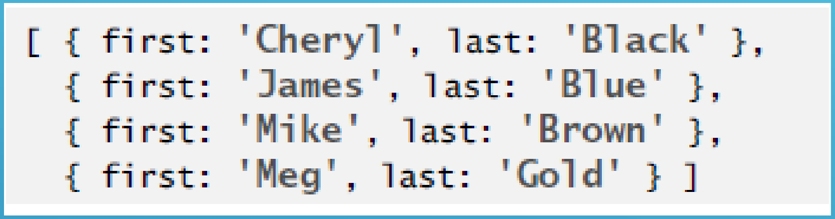
```
{ propertyName1 : value1, propertyName2 : value2 }
```

- Arrays are represented in JSON with square brackets in the following format:

```
[ value0, value1, value2 ]
```

USING JSON TO REPRESENT OBJECTS

- To appreciate the simplicity of JSON data, examine this representation of an array of address-book entries:

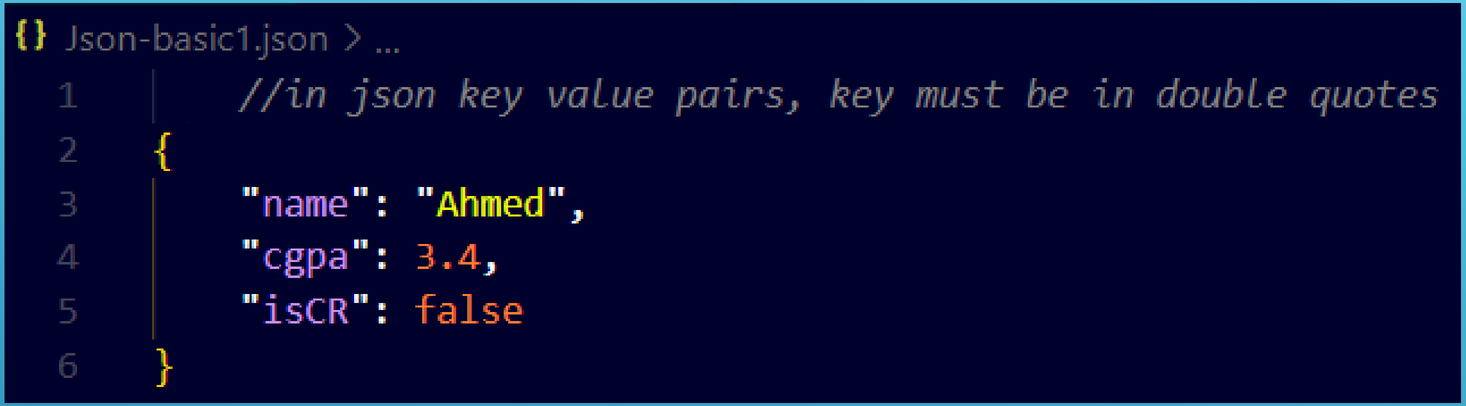


```
[ { first: 'Cheryl', last: 'Black' },
  { first: 'James', last: 'Blue' },
  { first: 'Mike', last: 'Brown' },
  { first: 'Meg', last: 'Gold' } ]
```

- JSON provides a straightforward way to manipulate objects in JavaScript, and many other programming languages now support this format.
- In addition to simplifying object creation, JSON allows programs to easily extract data and efficiently transmit it across the Internet.

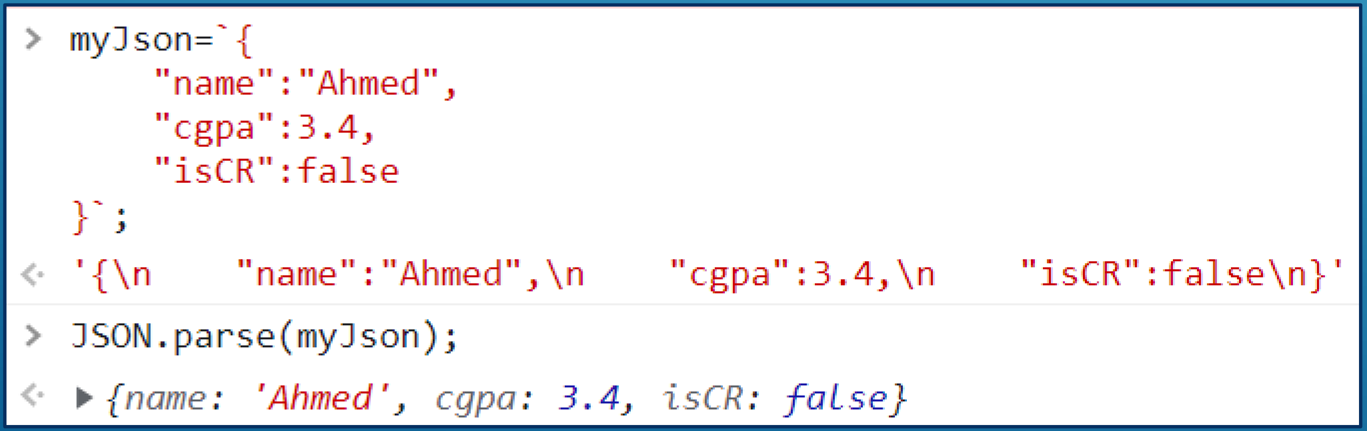
EXAMPLE PROGRAM IN JSON

Following is the sample JSON file



```
{} Json-basic1.json > ...
    //in json key value pairs, key must be in double quotes
{
    "name": "Ahmed",
    "cgpa": 3.4,
    "isCR": false
}
```

Open console tab in browser window and create a string variable name myJson and assign it the ``. Inside the back tick paste the above json-basic1.json file code. Run the command JSON.parse(myJson);You will find the object return by JSON.parse(myJson); command



```
> myJson=`{
      "name":"Ahmed",
      "cgpa":3.4,
      "isCR":false
  }`;
< '{\n    "name":"Ahmed",\n    "cgpa":3.4,\n    "isCR":false\n}'
> JSON.parse(myJson);
< ▶{name: 'Ahmed', cgpa: 3.4, isCR: false}
```

EXAMPLE PROGRAM IN JSON

Following is the sample JSON file

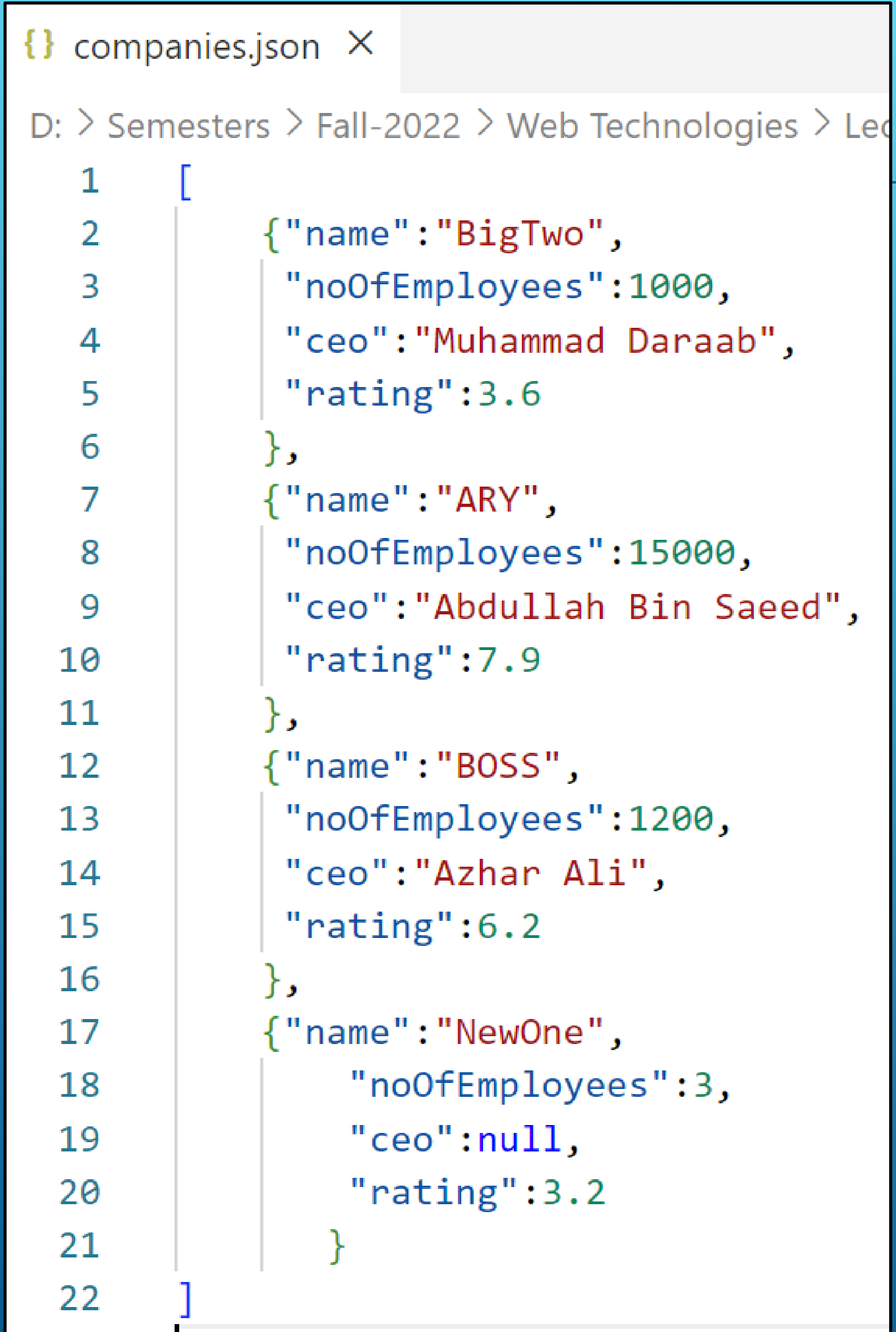
```
 {
        "Name": "Ali",
        "fovoritNumber":7,
        "is Programmer":true,
        "hobbies":["table tennice","running"],
        "friends":[{   "Name": "Abdul Moiz",
                     "fovoritNumber":9,
                     "is Programmer":false,
                     "hobbies":["football","cycling"]  },
                       {    "Name": "Waqas Raza",
                     "fovoritNumber":11,
                     "is Programmer":true,
                     "hobbies":["playing cricket","cycling"]
                       }
                    ]
 }
```

EXAMPLE PROGRAM IN JSON

Let take an example to use JSON inside the file, following are the steps:

1. Create the file with extension json, example Listing 11-3-companies.json

2. Inside the file we are going to put array of objects
3. Create the html file names Listing 11-4-companiesWeb.html and copy/paste the code of Listing-11-3-companies.json inside the script tag in html file

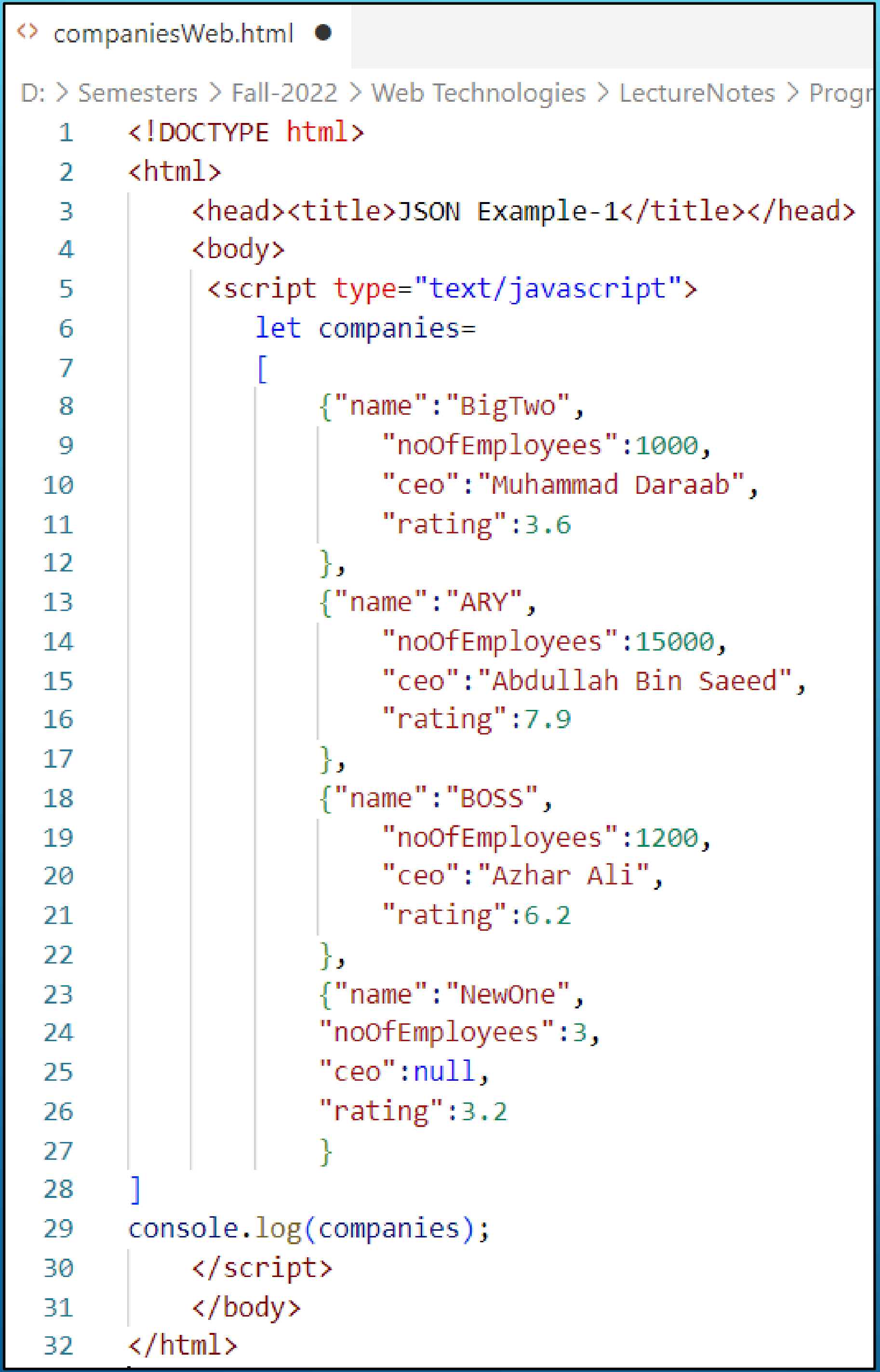


```
[
    {"name":"BigTwo",
     "noOfEmployees":1000,
     "ceo":"Muhammad Daraab",
     "rating":3.6
    },
    {"name":"ARY",
     "noOfEmployees":15000,
     "ceo":"Abdullah Bin Saeed",
     "rating":7.9
    },
    {"name":"BOSS",
     "noOfEmployees":1200,
     "ceo":"Azhar Ali",
     "rating":6.2
    },
    {"name":"NewOne",
        "noOfEmployees":3,
        "ceo":null,
        "rating":3.2
       }
]
```

EXAMPLE PROGRAM-1 IN JSON

Listing-11-3-companies.json

EXAMPLE PROGRAM-1 IN JSON



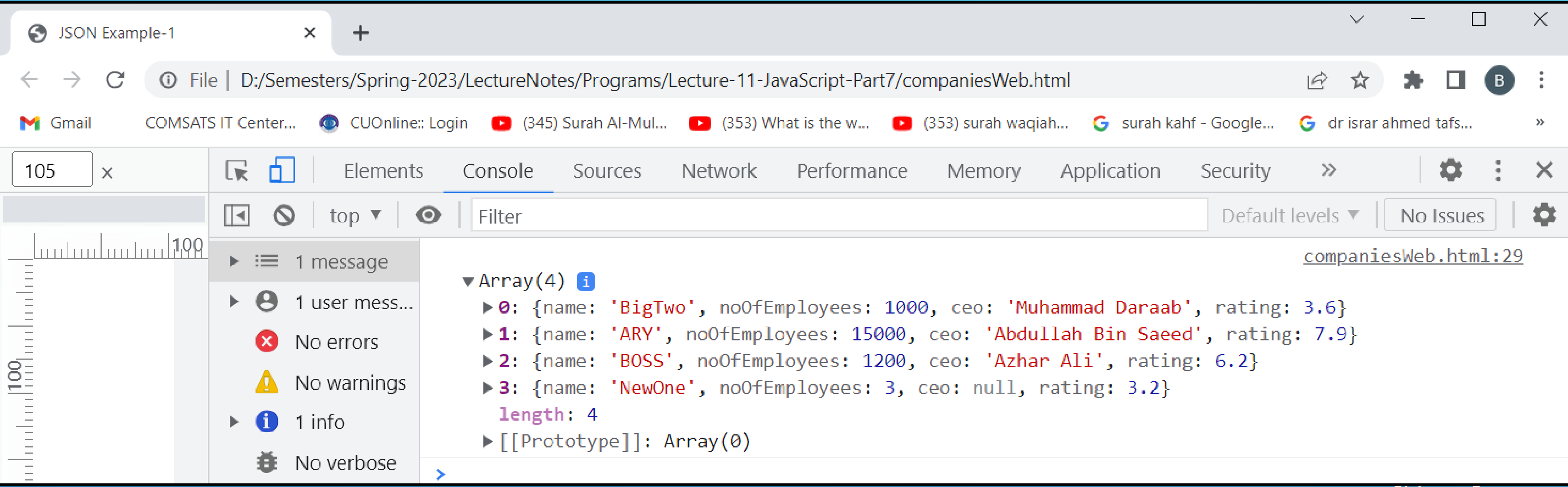
companiesWeb.html ●

D: > Semesters > Fall-2022 > Web Technologies > LectureNotes > Progr

```
<!DOCTYPE html>
<html>
    <head><title>JSON Example-1</title></head>
    <body>
     <script type="text/javascript">
        let companies=
        [
            {"name":"BigTwo",
                "noOfEmployees":1000,
                "ceo":"Muhammad Daraab",
                "rating":3.6
            },
            {"name":"ARY",
                "noOfEmployees":15000,
                "ceo":"Abdullah Bin Saeed",
                "rating":7.9
            },
            {"name":"BOSS",
                "noOfEmployees":1200,
                "ceo":"Azhar Ali",
                "rating":6.2
            },
            {"name":"NewOne",
            "noOfEmployees":3,
            "ceo":null,
            "rating":3.2
            }
]
console.log(companies);
    </script>
    </body>
</html>
```

Listing-11-4 companiesWeb.json

OUTPUT COMPANIESWEB.HTML

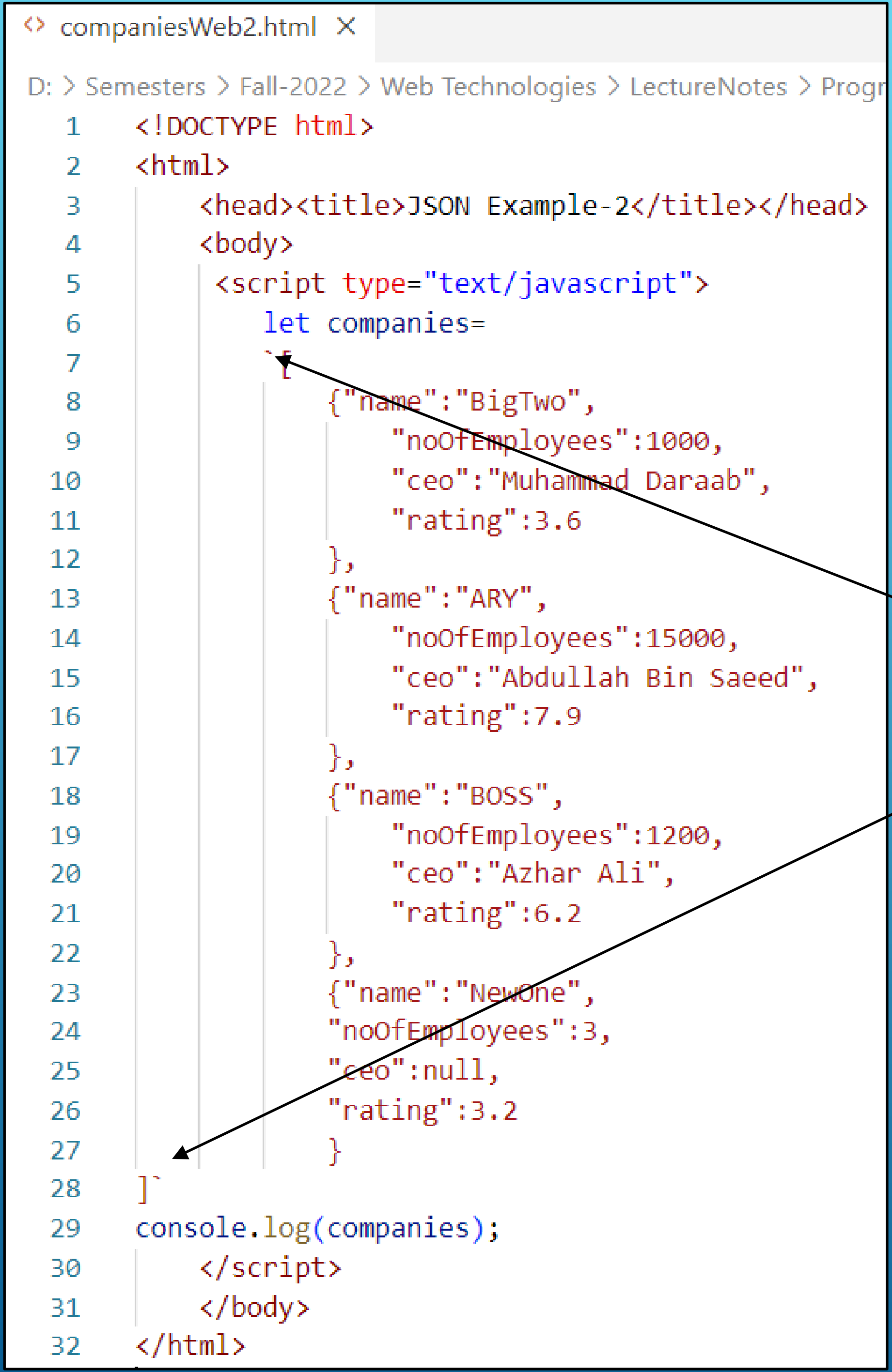


```
                                                                              companiesWeb.html:29
▼Array(4) i
  ▶0: {name: 'BigTwo', noOfEmployees: 1000, ceo: 'Muhammad Daraab', rating: 3.6}
  ▶1: {name: 'ARY', noOfEmployees: 15000, ceo: 'Abdullah Bin Saeed', rating: 7.9}
  ▶2: {name: 'BOSS', noOfEmployees: 1200, ceo: 'Azhar Ali', rating: 6.2}
  ▶3: {name: 'NewOne', noOfEmployees: 3, ceo: null, rating: 3.2}
   length: 4
  ▶[[Prototype]]: Array(0)
>
```

EXAMPLE PROGRAM-2 IN JSON

- Let put back tik(`)at start and end of array of objects of companies(it means you are making it constant) as shown in the modified listing of previous example companiesWeb.html and save as companiesWeb2,html, as shown in listing-11-5.

COMPANIESWEB2.HTML



companiesWeb2.html ×

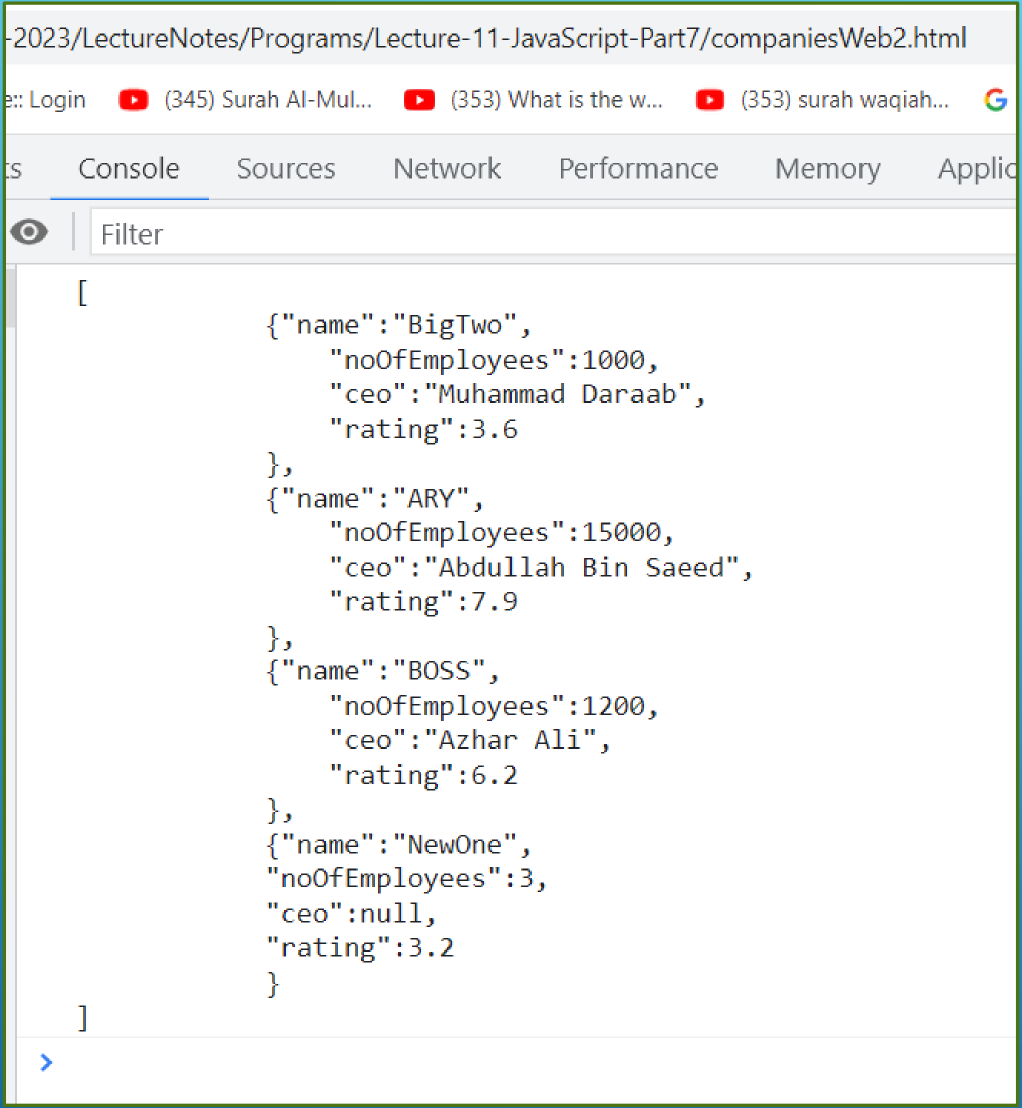
D: > Semesters > Fall-2022 > Web Technologies > LectureNotes > Progr

```
<!DOCTYPE html>
<html>
    <head><title>JSON Example-2</title></head>
    <body>
     <script type="text/javascript">
        let companies=
        `[
            {"name":"BigTwo",
                "noOfEmployees":1000,
                "ceo":"Muhammad Daraab",
                "rating":3.6
            },
            {"name":"ARY",
                "noOfEmployees":15000,
                "ceo":"Abdullah Bin Saeed",
                "rating":7.9
            },
            {"name":"BOSS",
                "noOfEmployees":1200,
                "ceo":"Azhar Ali",
                "rating":6.2
            },
            {"name":"NewOne",
            "noOfEmployees":3,
            "ceo":null,
            "rating":3.2
            }
]`
console.log(companies);
    </script>
    </body>
</html>
```

Listing11-5-companiesWeb2.json

Note the back tick at start and end of array

OUTPUT COMPANIESWEB2.HTML



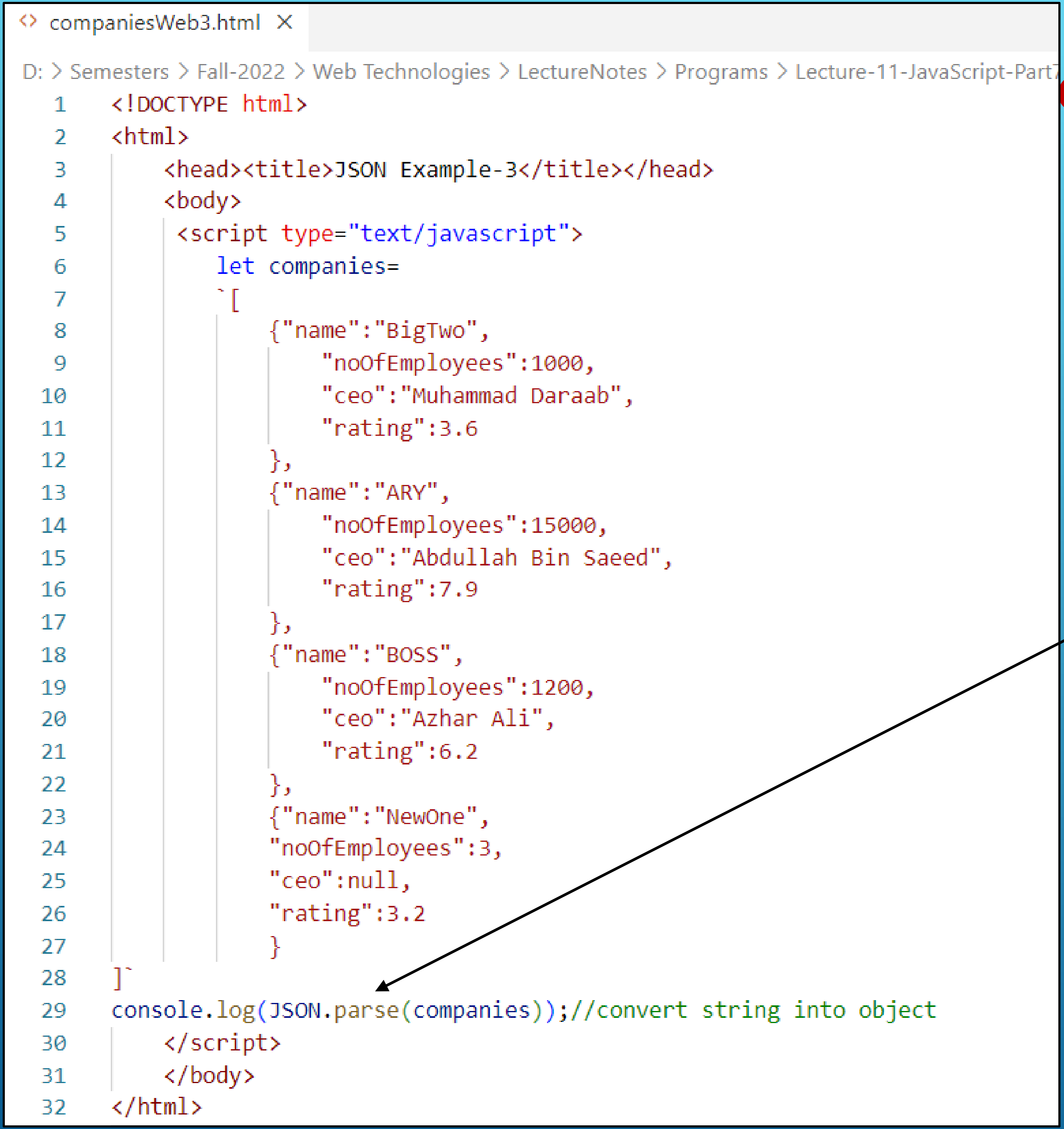
```
[
            {"name":"BigTwo",
                "noOfEmployees":1000,
                "ceo":"Muhammad Daraab",
                "rating":3.6
            },
            {"name":"ARY",
                "noOfEmployees":15000,
                "ceo":"Abdullah Bin Saeed",
                "rating":7.9
            },
            {"name":"BOSS",
                "noOfEmployees":1200,
                "ceo":"Azhar Ali",
                "rating":6.2
            },
            {"name":"NewOne",
            "noOfEmployees":3,
            "ceo":null,
            "rating":3.2
            }
]
>
```

As we run the program companiesWeb2.html, now you can see that whole array of objects are displayed as it is in console window output companiesWeb2.html.

EXAMPLE PROGRAM-3 IN JSON

- Let modify the companiesWeb2.html and save as companiesWeb3.html, In modified code we convert the constant javascript string into javascript object by using parse method in console.log() as shown in listing 11-6.

COMPANIESWEB3.HTML



companiesWeb3.html

D: > Semesters > Fall-2022 > Web Technologies > LectureNotes > Programs > Lecture-11-JavaScript-Part

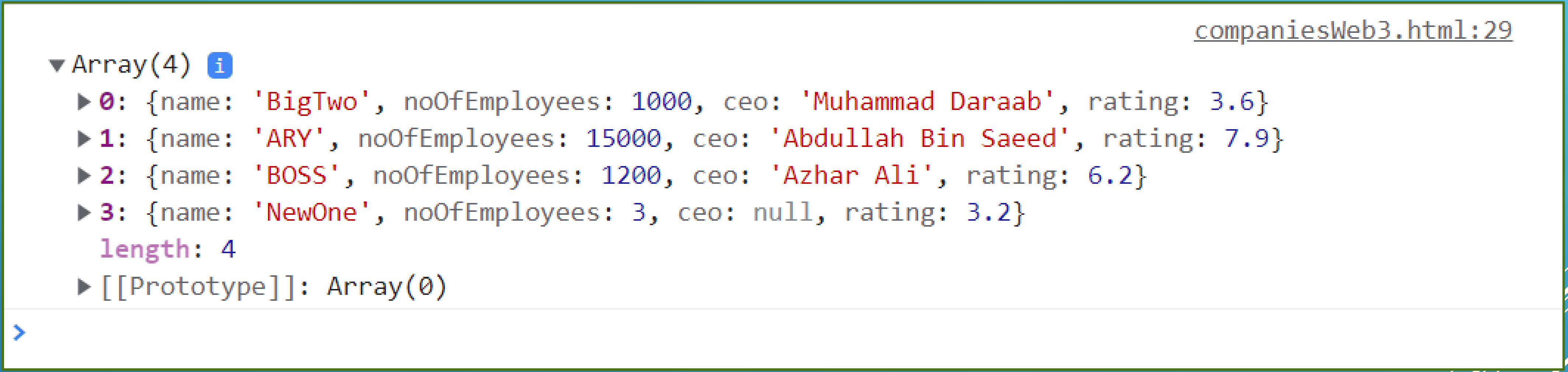
```
<!DOCTYPE html>
<html>
    <head><title>JSON Example-3</title></head>
    <body>
     <script type="text/javascript">
        let companies=
        `[
            {"name":"BigTwo",
                "noOfEmployees":1000,
                "ceo":"Muhammad Daraab",
                "rating":3.6
            },
            {"name":"ARY",
                "noOfEmployees":15000,
                "ceo":"Abdullah Bin Saeed",
                "rating":7.9
            },
            {"name":"BOSS",
                "noOfEmployees":1200,
                "ceo":"Azhar Ali",
                "rating":6.2
            },
            {"name":"NewOne",
            "noOfEmployees":3,
            "ceo":null,
            "rating":3.2
            }
]`
console.log(JSON.parse(companies));//convert string into object
    </script>
    </body>
</html>
```

Listing-11-6-companiesWeb3.json

JSON.parse()

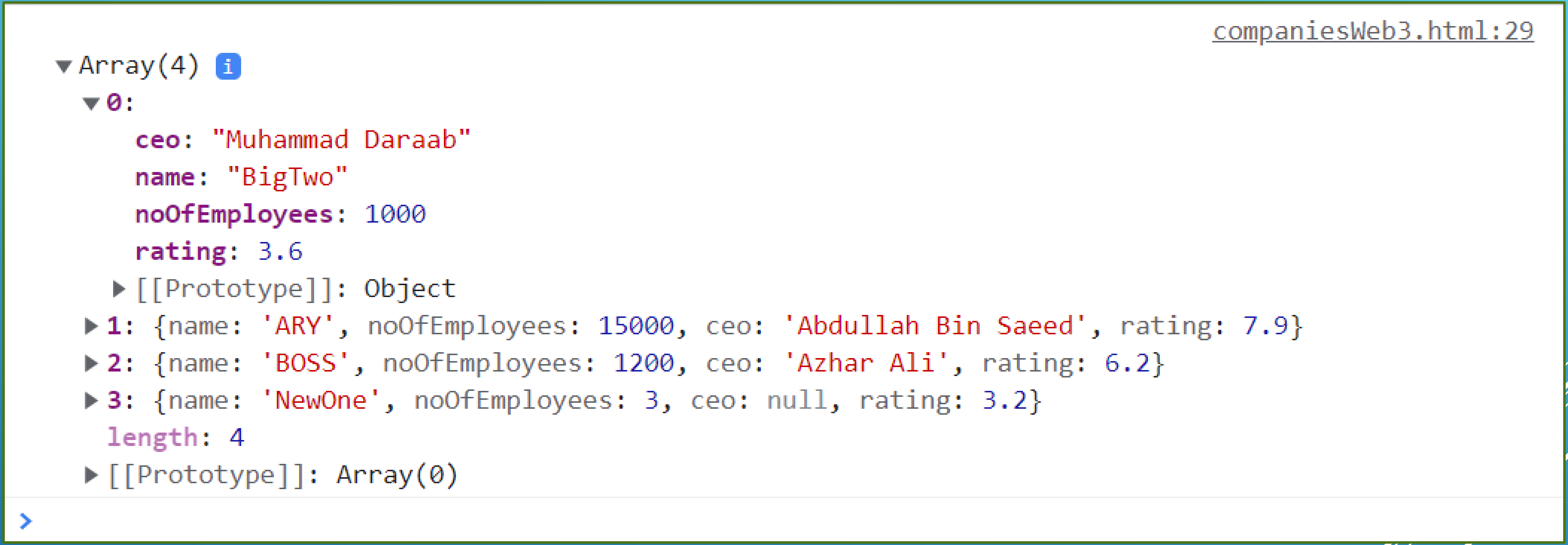
OUTPUT COMPANIESWEB3.HTML

Run the program companiesWeb3.html, now you can see that whole array of 4 objects are displayed in console window .



```
                                                              companiesWeb3.html:29
▼Array(4) i
  ▶0: {name: 'BigTwo', noOfEmployees: 1000, ceo: 'Muhammad Daraab', rating: 3.6}
  ▶1: {name: 'ARY', noOfEmployees: 15000, ceo: 'Abdullah Bin Saeed', rating: 7.9}
  ▶2: {name: 'BOSS', noOfEmployees: 1200, ceo: 'Azhar Ali', rating: 6.2}
  ▶3: {name: 'NewOne', noOfEmployees: 3, ceo: null, rating: 3.2}
   length: 4
  ▶[[Prototype]]: Array(0)
>
```

OUTPUT COMPANIESWEB3.HTML

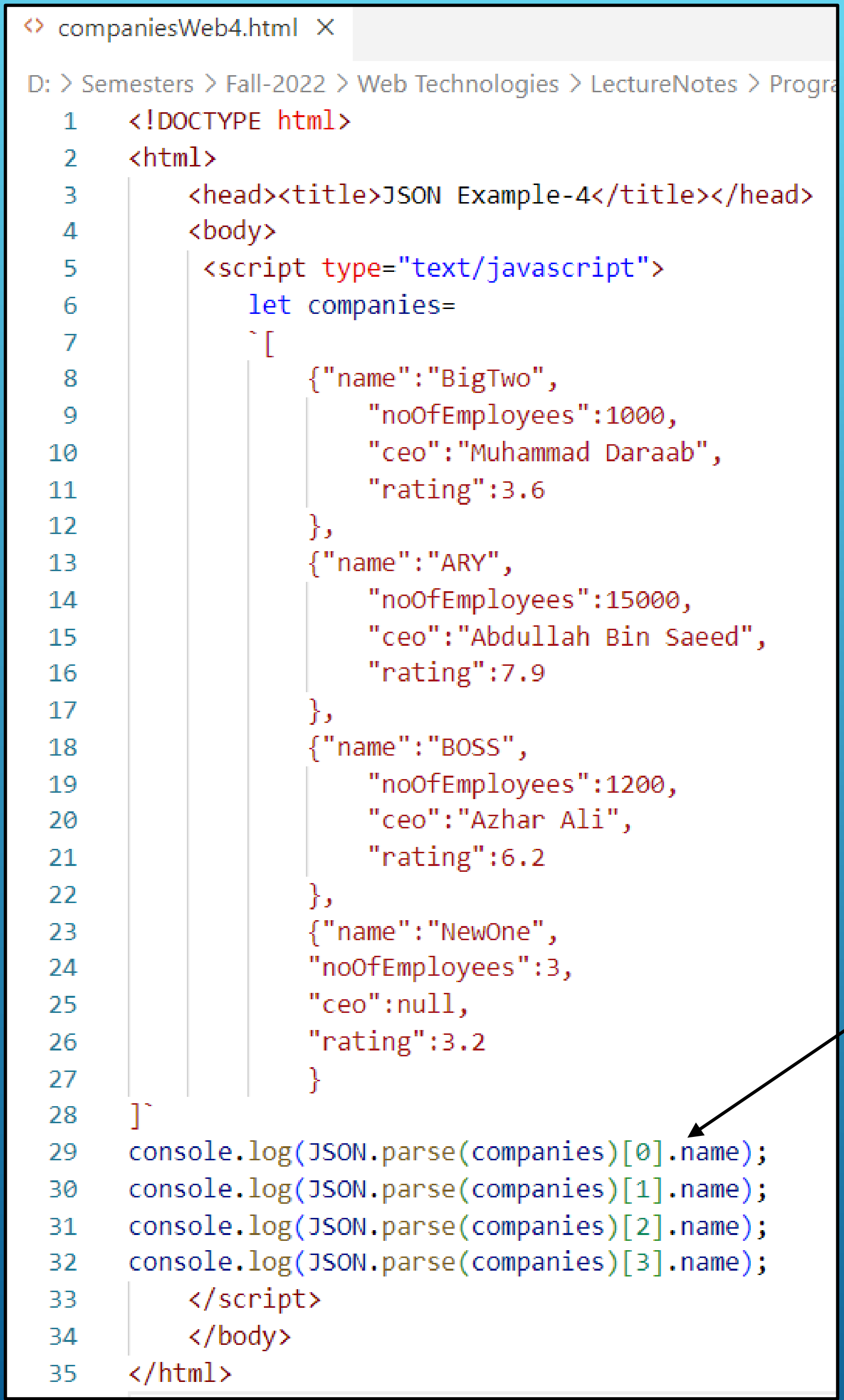


```
                                                                  companiesWeb3.html:29
▼Array(4) i
  ▼0:
      ceo: "Muhammad Daraab"
      name: "BigTwo"
      noOfEmployees: 1000
      rating: 3.6
    ▶[[Prototype]]: Object
  ▶1: {name: 'ARY', noOfEmployees: 15000, ceo: 'Abdullah Bin Saeed', rating: 7.9}
  ▶2: {name: 'BOSS', noOfEmployees: 1200, ceo: 'Azhar Ali', rating: 6.2}
  ▶3: {name: 'NewOne', noOfEmployees: 3, ceo: null, rating: 3.2}
   length: 4
  ▶[[Prototype]]: Array(0)
>
```

EXAMPLE-4 PROGRAM IN JSON

- Let modify the companiesWeb3.html and save as companiesWeb4,html, In modified code we are accessing individual data members in each object, for example let we print the names of each company, listing-11-7.

COMPANIESWEB4.HTML



companiesWeb4.html ×

D: > Semesters > Fall-2022 > Web Technologies > LectureNotes > Progra

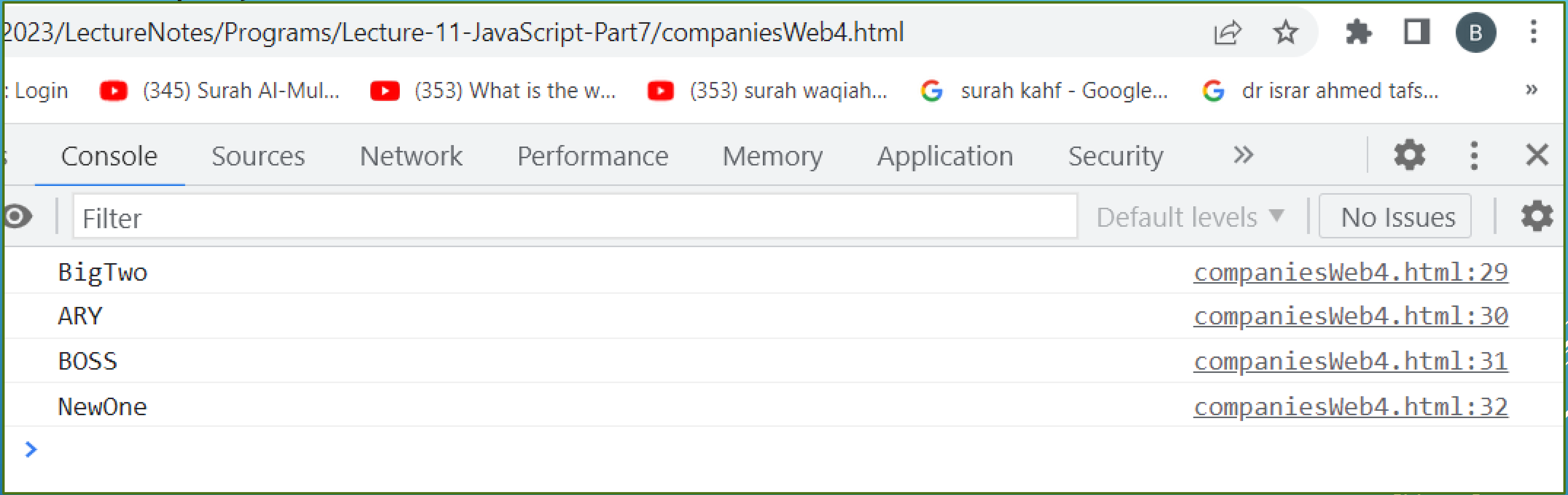
```
<!DOCTYPE html>
<html>
    <head><title>JSON Example-4</title></head>
    <body>
     <script type="text/javascript">
        let companies=
        `[
            {"name":"BigTwo",
                "noOfEmployees":1000,
                "ceo":"Muhammad Daraab",
                "rating":3.6
            },
            {"name":"ARY",
                "noOfEmployees":15000,
                "ceo":"Abdullah Bin Saeed",
                "rating":7.9
            },
            {"name":"BOSS",
                "noOfEmployees":1200,
                "ceo":"Azhar Ali",
                "rating":6.2
            },
            {"name":"NewOne",
            "noOfEmployees":3,
            "ceo":null,
            "rating":3.2
            }
]`
console.log(JSON.parse(companies)[0].name);
console.log(JSON.parse(companies)[1].name);
console.log(JSON.parse(companies)[2].name);
console.log(JSON.parse(companies)[3].name);
    </script>
    </body>
</html>
```

Listing-11-7
companiesWeb4.json

Accessing name of each company in console window

OUTPUT COMPANIESWEB4.HTML

Run the program companiesWeb4.html, now you can see the name of each company in console window .



```
BigTwo                                                  companiesWeb4.html:29
ARY                                                     companiesWeb4.html:30
BOSS                                                    companiesWeb4.html:31
NewOne                                                  companiesWeb4.html:32
>
```