WEEK 5

2

TOPICS

- JSON
  - JSON and serialization
- Asynchronous operation
- What is an API
  - Postman
- Accessing an API using AJAX
- Accessing an API using Fetch

3

REVIEW

- Objects have p______, m______ and e_______
- Instance a Javascript object using the keyword____________
- You have an object defined as follows:

```
function Car()
{
    /* object stuff here */
    goLeft= function() {/*stuff here*/}
}
  car = new Car();
```

what is the correct way to access the goLeft method:
car->goLeft()   car.goLeft()   goLeft(car) other???

- In the code: `setTimeout (5000, abc);`   abc is a "_________ function"

4

REVIEW

```
function Rock(color)
{
    this.color = color;
    this.weight = 0;
    this.display =  showRock;
}

function showRock()
{
     s = this.weight;
     return s;
}

r  = new Rock('grey');
alert(r.display());   // what is displayed?
```

5

REVIEW

1. You have a function defined as: `calc = (a, b) => a % b;`

Show a call the function to get a result of 2

2. What is displayed:

```
a = new Array(3,4,5); a[3]=0; a.push(17);
console.log(a.join('-'))
```

3. What are three ways to associate an event handler with an event?

4. The array .map() method uses a function to change an array [true][false]

5. What is the difference between a one-dimensional array and an associative array?

6. To iterate through the set of indexes in an associative array, use ...

6

REVIEW

1. You have a form as:

```
<form method='get' action='dosomething.php'>
<!- form stuff here -->
<input type ='submit' value='submit' />
</form>
```

The best event to use to validate this form before proceeding to the "action" is ________

2. You have a form element defined as follows:

```
<select id='pick'>
<option>cats</option>
<option>dogs</option>
</select>
```

- How can you get the display text for the element that is currently selected?

7

REVIEW

```
<form>

Number: <input type="text" name="number">

<input type ='button' onclick='read()'>

</form>

<script>
function read()

{

    alert(document.getElementById("number"))

}
</script>
```

You want to display the number in the alert, but it is not showing correctly when you click the button. What is the MOST likely cause of the problem?

8

JSON

9

JSON

- Javascript Object Notation
- For data representation and transmission
- File extension is .json
- key - value pairs
- Text based

- Minimal and portable
- Often used between web app and server
- Based on conventions seen in many languages
- Code for parsing JSON is available in many languages

10

10

SIMPLE JSON OBJECT

```
{
    “first name”  : “Julie”,
    “last name”  : “Smith”,
    “course”      : “Web Programming”,
    “grade”       :  92
}
```

Notes:

- “key” : value
- Start and end with { }
- Keys in quotes
- Values are strings, numbers, booleans, and null
  - or an array or other object containing these types
- Commas between pairs
- Validators exist to check your JSON

  https://jsonlint.com/

11

11

TRY IT!

- Create a JSON that will be used by an animal adoption center to represent its animals.
- The following data should be included:
  - Name
  - Type (ex, dog, cat, rabbit)
  - Breed
  - Age
  - Gender
- Use jsonlint.com to check your file

12

12

A VALUE CAN BE AN ARRAY

- Use [ ] notation to indicate an array

```
{
    “first name”  : “Julie”,
    “last name”  : “Smith”,
    “course”       : “Web Programming”,
    “grades”       :  [88, 95, 91, 92]
}
```

13

13

TRY IT!

- Add the following to your pet object
  - Procedures that have been completed (i.e. , spay, vaccinations)- as an array

CONTAINED OBJECTS

- A JSON value can be another object
- Example: The student name field can be a sub-object

```
{
   “name” : {
                    “first”  : “Julie”,
                    “last”   : “Smith”
   }
   “course”       : “Web Programming”,
   “grades”       :  [88, 95, 91, 92]
}
```

SET OF JSON OBJECTS

- You can create an array of objects using the [ ] notation

```
[
   {“id” : 1,   “type” : “rose”},
   {“id” : 2,   “type” : “carnation”},
   {“id” : 3,   “type” : “sunflower”}
]
```

16

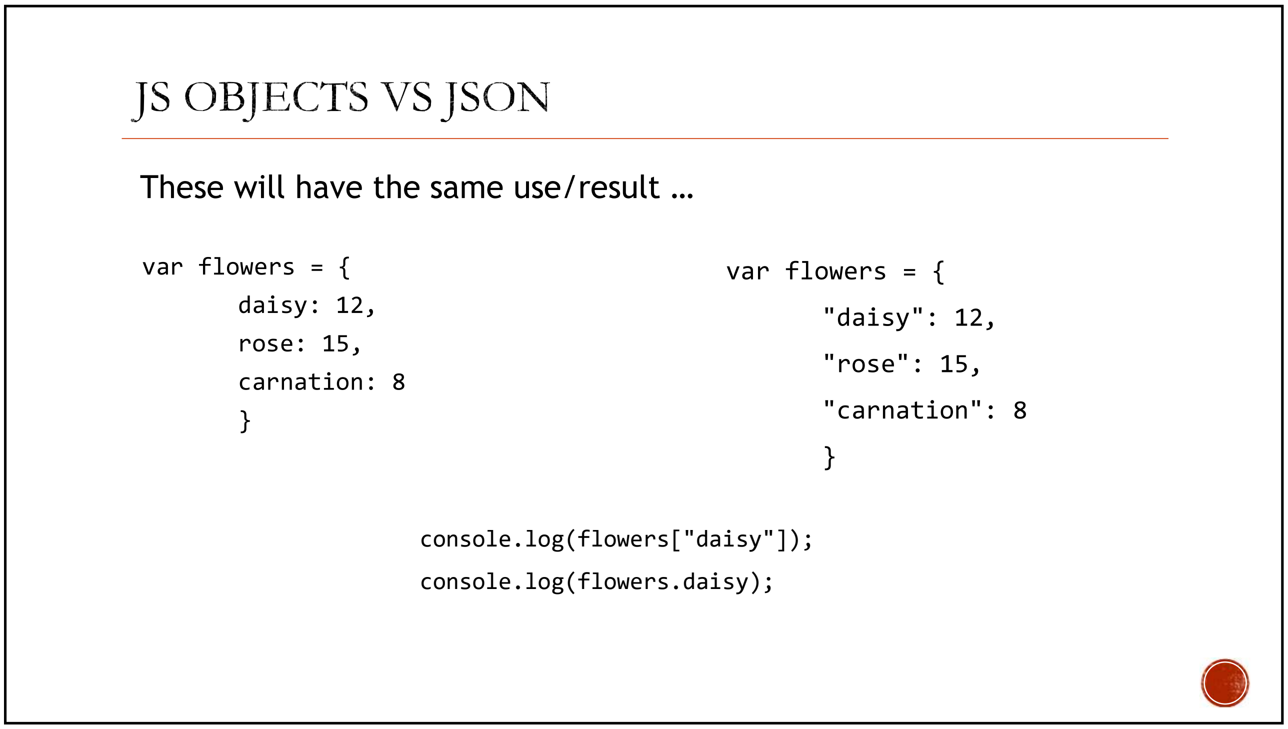
16

TRY IT!

- Add an “adopting family” field
- This should include
  - Name
  - Town
  - Number children in household
  - Other pets (true or false)
- Now create an array of three pets

17

17



JS OBJECTS VS JSON

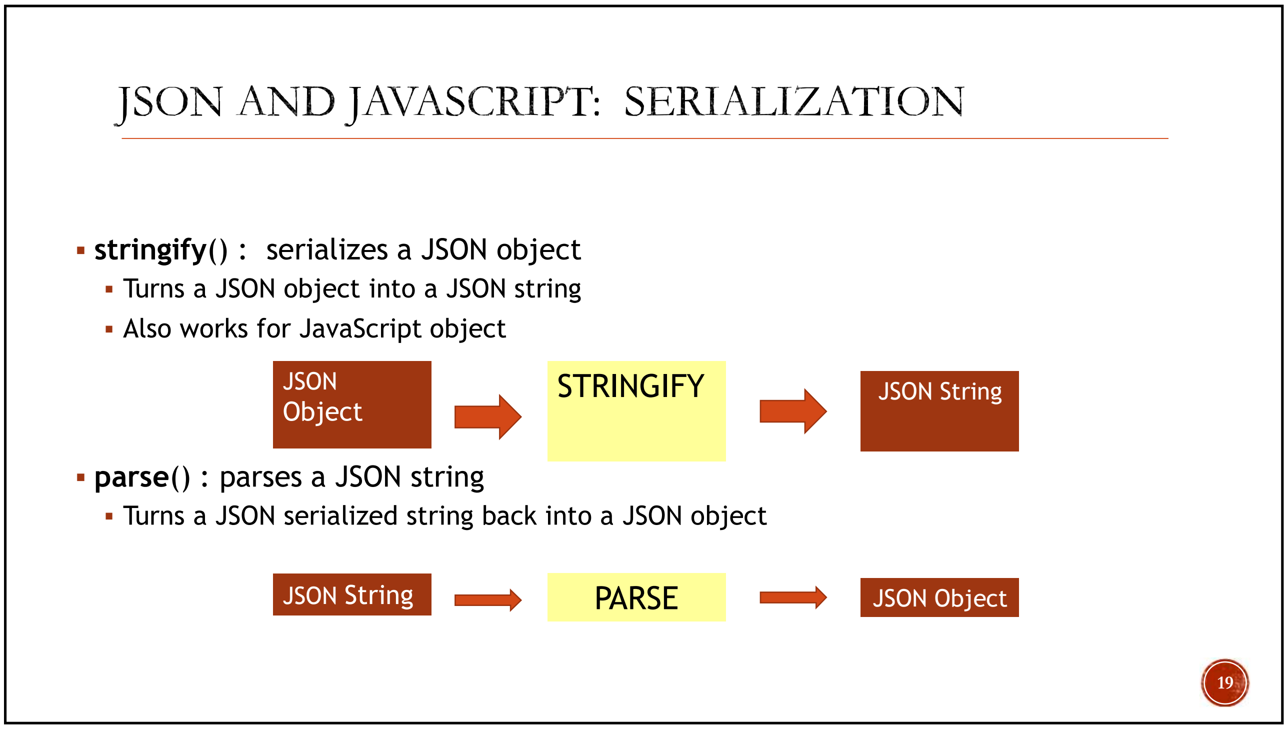
These will have the same use/result …

```
var flowers = {
      daisy: 12,
      rose: 15,
      carnation: 8
      }
```

```
var flowers = {
      "daisy": 12,
      "rose": 15,
      "carnation": 8
      }
```

```
console.log(flowers["daisy"]);
console.log(flowers.daisy);
```

18



JSON AND JAVASCRIPT: SERIALIZATION

- **stringify**() : serializes a JSON object
  - Turns a JSON object into a JSON string
  - Also works for JavaScript object

$$\begin{tikzcd}\text{JSON Object} \arrow[r] & \text{STRINGIFY} \arrow[r] & \text{JSON String}\end{tikzcd}$$

- **parse**() : parses a JSON string
  - Turns a JSON serialized string back into a JSON object

$$\begin{tikzcd}\text{JSON String} \arrow[r] & \text{PARSE} \arrow[r] & \text{JSON Object}\end{tikzcd}$$

19

19

STRINGIFY AND PARSE

```
student = {
             “name” : “Suzie”,
             “course” : “comp20”
          }

strStudent = JSON.stringify(student);  // serialize to a string

objStudent = JSON.parse(strStudent);  //restores to an object
```

USING STRINGIFY() WITH A JS OBJECT

```
var flowers = {
       daisy: 12,
       rose: 15,
       carnation: 8
}
```

Turn it into a JSON string

```
s = JSON.stringify(flowers)
document.write(s)
```

```
// output:
{"daisy":12,"rose":15,"carnation":8,"tulip":7}
```

And back to an object …

```
obj= JSON.parse(s)
document.write(obj['daisy’])
```

```
// output
12
```

toJSON

- toJSON method tells stringify how to serialize the object (ie what is in the string)

```
student = {
            “name” : “Suzie”,
            “course” : “comp20”,
            toJSON() { return this.name + “:” + this.course}
      }
```

22

22

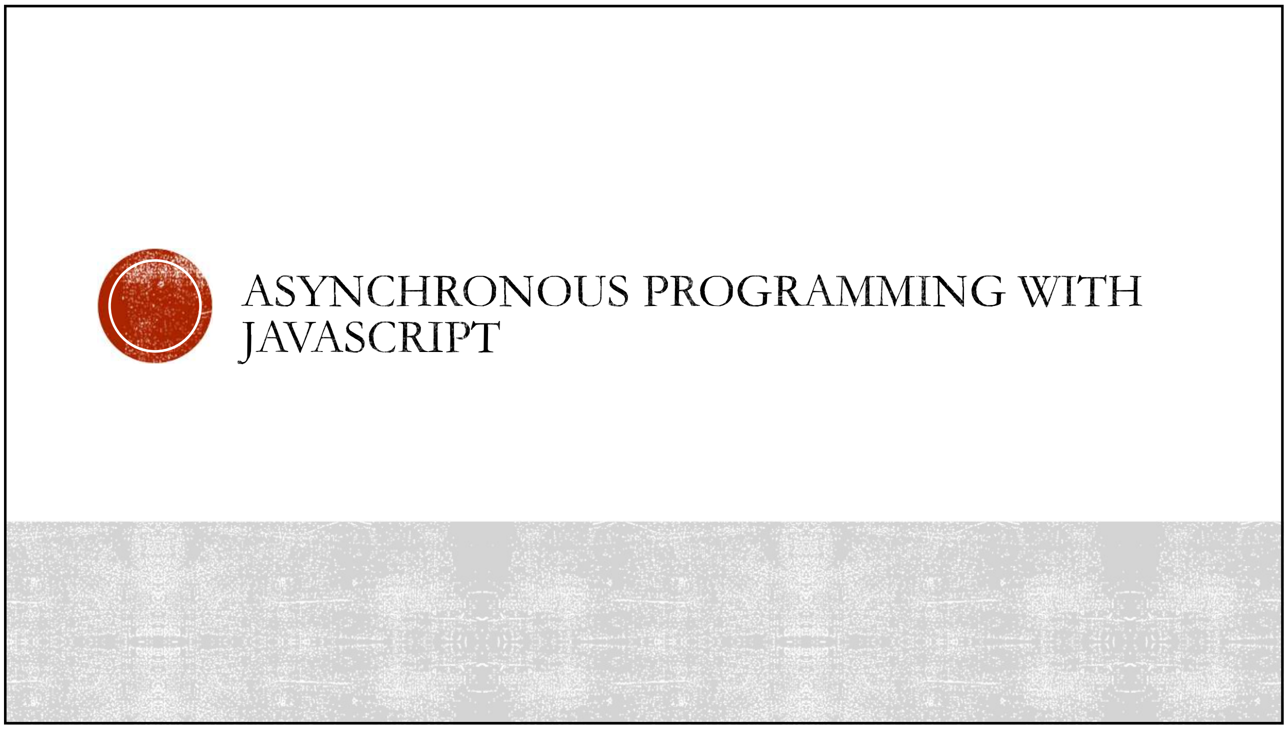
READING A JSON FILE USING $.GET

- The jQuery get method can read a json file asynchronously (more on that later)
- The JSON file is the target of the request
- A callback function handles processing the data
- A parameter passed to the callback represents the data as an object or a string
- The JSON file must be on a server
- You will need to include the jQuery library
- $.get(file, callback);

```
$.get( "https://online.com/file.json",
      function( data )
      {
        str = JSON.stringify(data);
        document.write(str);
      }
    )  //end get
```

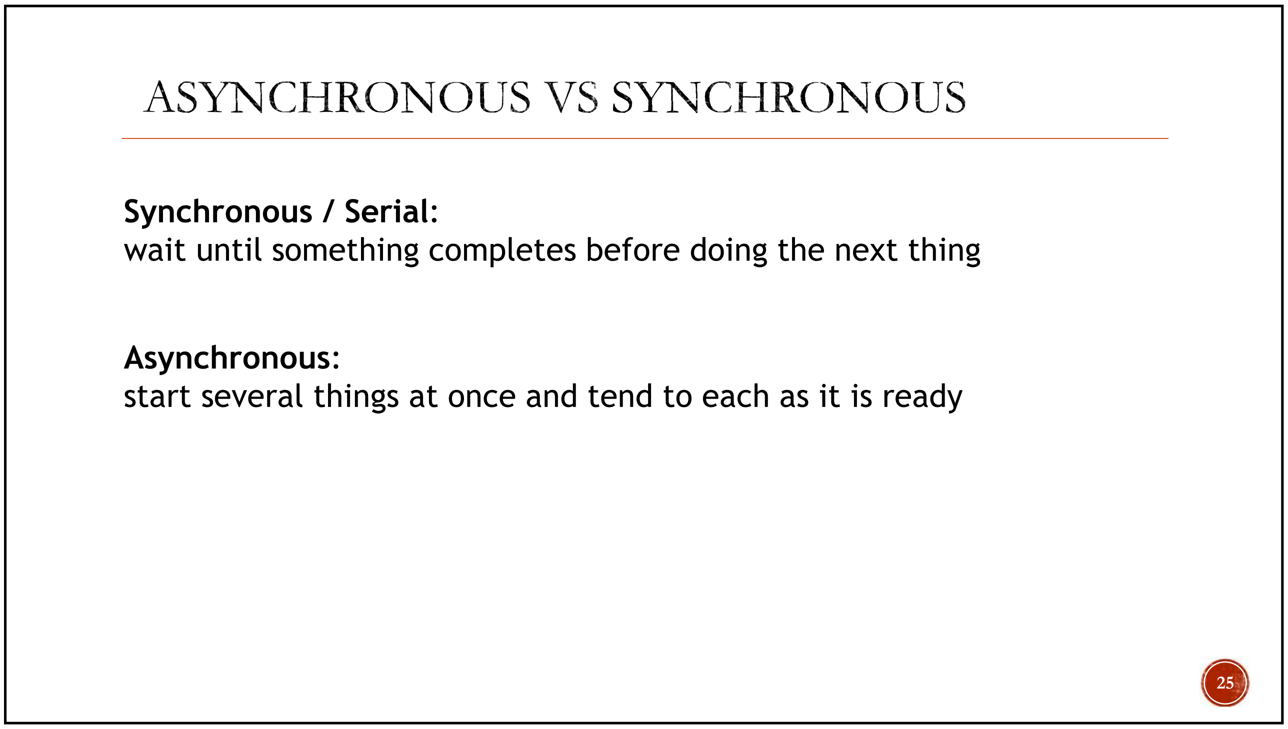
23

23



ASYNCHRONOUS PROGRAMMING WITH JAVASCRIPT

24



ASYNCHRONOUS VS SYNCHRONOUS

Synchronous / Serial:
wait until something completes before doing the next thing

Asynchronous:
start several things at once and tend to each as it is ready

25

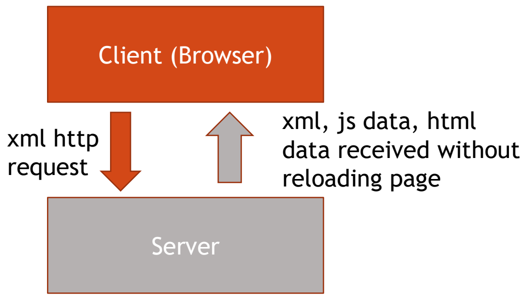
25

ASYNCHRONOUS OPERATIONS WITHIN JAVASCRIPT

Fetch data asynchronously from a web server without needing to refresh the page

This can be for an API or a data file such as a JSON

Data is sent in the background without needing to wait for a response



26

26

ACCESSING AN API

- An API or Application Programming Interface refers to specialized functionality that lives on a web server
- The API allows an organization to provide access to their data without compromising their data
- Many API sites require an API key to get access to their API
- Example: openweathermap.org
  - You will need an API key to access this site

Technologies

- AJAX
- fetch()
- $.get()
- REST

27

27

FINDING API'S

- API's may be offered to assist/attract a customer base
  - Example: UPS tracking, Flight info
- API's are also offered as a standalone service
- Several API's are available free or free with limited usage
- Adding an API to your application will significantly increase the functionality you are able to offer
- Rapidapi.com is a large repository for API's of all kinds – but many are not free.
- You can access an API using Javascript, PHP, Node.JS and more
- Technologies to access an API include AJAX, Promises, REST, cURL, and more

28

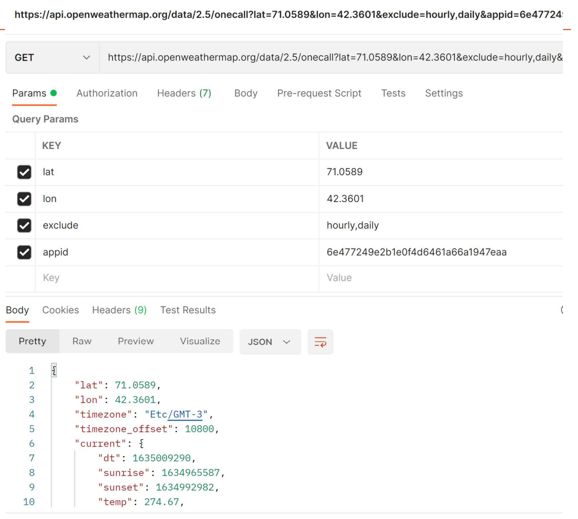
AUTHENTICATION AND API'S

- One or more levels of authentication may be required to access an API. In both cases the user needs to create an account with the API organization.
- API Key
  - The key is assigned after the user creates the account.
  - The key is easy to implement as it is just one additional name/value pair to be added to the URL or post data
- oAuth
  - oAuth (Open Authorization) allows a user to authorize an app or service by using another.
  - oAuth uses access tokens that contains information about the user and the resource the token is intended for.
  - oAuth2.0 was expanded to allow programmatic access to an API through a multi—step process.
    - Credentials are sent to the API which returns the access token
    - The token is then used in any subsequent access to the API

29

POSTMAN.COM

- Postman is a web application that helps to test an API call – with NO coding needed.
- A postman collection is a json object that details the correct parameters for an API call.
- Example: http://www.zippopotam.us/



https://api.openweathermap.org/data/2.5/onecall?lat=71.0589&lon=42.3601&exclude=hourly,daily&appid=6e47724

https://api.openweathermap.org/data/2.5/onecall?lat=71.0589&lon=42.3601&exclude=hourly,daily&

	KEY	VALUE
✓	lat	71.0589
✓	lon	42.3601
✓	exclude	hourly,daily
✓	appid	6e477249e2b1e0f4d6461a66a1947eaa
	Key	Value

```
{
    "lat": 71.0589,
    "lon": 42.3601,
    "timezone": "Etc/GMT-3",
    "timezone_offset": 10800,
    "current": {
        "dt": 1635009290,
        "sunrise": 1634965587,
        "sunset": 1634992982,
        "temp": 274.67,
```

NOTE: CROSS-ORIGIN REQUEST SHARING (CORS)

- Security policy that applies when your browser fetches assets for a web page
  - Fonts
  - Images
  - Scripts
- Security policies minimize the risks associated with code that can hack a browser
  - Downloading malicious code
  - “Hijacking” the browser
  - Adding undesirable plugins

FYI: SECURITY POLICIES

- Same origin
  - “Documents” must have the same origin
  - A page hosted on a server can only interact with other documents that are also on that server
  - Even a different protocol (http vs. https) will be deemed as a different origin
- CORS
  - Cross origin requests
  - Server will specify what can gain access and how they gain access
  - Accomplished with http headers: Access-Control-Allow-Origin
  - Headers can be set up on the server or in an .htaccess file:

```
<Files "*.json">
  Header set Access-Control-Allow-Origin "*"
</Files>
```

32

32

AJAX: ASYNCHRONOUS JAVASCRIPT AND XML

- **XMLHttpRequest** can send and receive data from web server.
- **readystate** has a value between 0 to 4 to indicate the status of request.
- **onreadystatechange** an event for the XMLHttpRequest object that is triggered when there is a change in the readystate value
- **open()/send()** methods of the XMLHttpRequest object - send the request
- Data can be represented using JSON or XML
- *In this course, we will focus on JSON*

33

33

READY STATE AND STATUS

Ready State Values

0 : Unsent
nothing happened yet - open() not called

1 : Opened
send() not yet called

2 : Headers Received
send() and open() called

3 : Loading
Data is being received

4 : Done
Operation completed

Status

200	Success
201	Resource was created
204	Request is successful, but data not received.
404	Page Not Found

Ready State =>4 and Status => 200
indicates successful completion of the request

34

PUTTING IT TOGETHER

```
function requestData() {
  var reqObj = new XMLHttpRequest();
  if (! reqObj)
      {alert(“Error: Can’t create HTTP Request object"); return;}
  data = “id:101”;
  reqObj.onreadystatechange = getMyData();
  reqObj.open("POST", "https://asyncrequest.com", true);
  reqObj.send(data);
}
```

35

OPEN() AND SEND()

```
req = new XMLHttpRequest();
req.open("post","https://asyncrequest.com",true);
```

- Parameters:
  - post or get
  - Address of processing file on server (relative path)
  - Boolean: is this to be sent asynchronously (normally, true)
- req.send(“id:101");
  - Uses a JSON string

36

36

EXAMPLE, CONTINUED

```
function getMyData()
{
      if(this.readyState==4 && this.status==200)
      {
        var data=this.responseText;
        var info=JSON.parse(data);  //convert to object if needed
        for(i in info ){
           document.write(i + “:"+ info[i]);
        }
}
```

37

37

ASYNCHRONOUS CALLS USING A PROMISE

- A promise is a placeholder for the result of an asynchronous operation.
- Promises will often be used with API's
- They can resolve successfully or unsuccessfully.

```
new Promise (resolves_callback, rejects_callback) => {
            // api call  here
            // uses the resolves and rejects callback functions on success or failure
 }
```

38

FETCH

- The function fetch() uses Promises
- Uses for fetch() include accessing an API endpoint or reading a JSON file
- Example:

```
 res = fetch("https://abc.com/location.json")
                .then (res => res.text())
                .then (data => console.log(data))
                .catch (error => console.log(error))
```

Reference: https://medium.com/@armando_amador/how-to-make-http-requests-using-fetch-api-and-promises-b0ca7370a444

39

39

EXAMPLE – USING FETCH WITH THE WEATHER API

```
res = fetch("https://api.openweathermap.org/data/2.5/weather?lat=42.519539&lon=-
70.896713&appid=xxx&units=imperial")
        .then (res => res.text())
        .then (data =>
          {
                data = JSON.parse(data)
                data = data.current.temp;
            console.log("The current temperature is " + data + " degrees")
          })
        .catch (error => console.log(error))
```

Note: test the URL in postman.com first to ensure your request and response is what you intended.

40

EXAMPLE – CREATE AN API

```
<?php
        //contrary.php
        $word = $_GET["w"];
        if ($word == "black")
            $resp=  "White";
        else if ($word == "white")
            $resp= "Black";
        else if ($word == "up")
            $resp= "Down";
        else if ($word == "down")
            $resp= "Up";
        else if ($word == "yes")
            $resp= "No";
        else if ($word == "no")
            $resp= "Yes";
        else $resp= "nothing to say!";

   echo "{\"word\":\"$word\",\"response\":\"$resp\"}";
?>
```

41

EXAMPLE: ACCESS AN API USING AJAX

```
<body>

    <h1>Contrary</h1>
    <form>Pick a word: <input type =
'text' name = 'word'>
        <br><input type = "button"
            value = "Get Response"
            onclick="loadData()">
    </form>
    <div id="cData">&nbsp;</div>
</body>
```

```
<script>
        function loadData() {

            request = new XMLHttpRequest();
            theWord = document.forms[0].word.value;
            theURL= "https://examples.secretcheese.com/ajax/contrary.php?w="+theWord
            request.open("GET", theURL , true);
            request.onreadystatechange = function()
           {

                if (request.readyState == 4 && request.status == 200)
                {
                    theData = request.responseText;
                    resp = JSON.parse(theData)
                    document.getElementById("cData").innerHTML = "<br>The response is: " + theData
                                + "<br>response word is: "  + resp['response'];

                } //end if completed
                else if (request.readyState == 4 && request.status != 200)
                {
                   document.getElementById("cData").innerHTML += "<br>Request failed!";
                }
            }  // end event
            request.send();
        }  // end load data
 </script>
```

42

EXAMPLE: ACCESS AN API USING FETCH

```
<body>

    <h1>Contrary</h1>
    <form>Pick a word:
        <input type = 'text'
               name = 'word'>
        <br><input type = "button"
            value = "Get Response"
            onclick="loadData()">
    </form>
    <div id="cData">&nbsp;</div>
</body>
```

```
<script>
function loadData()
{
     theWord = document.getElementById('word').value;
     url = "https://examples.secretcheese.com/ajax/contrary.php?w=" + theWord
     fetch(url)
        .then(res => res.text())
        .then (data =>
           {
             resp = JSON.parse(data)
             document.getElementById("cData").innerHTML =
                     "<br>The response is: " + resp["response"];
           })  //end then
         .catch (error => console.log(error))
 }  // end load data
</script>
```

43