7. Callbacks & Schedulers

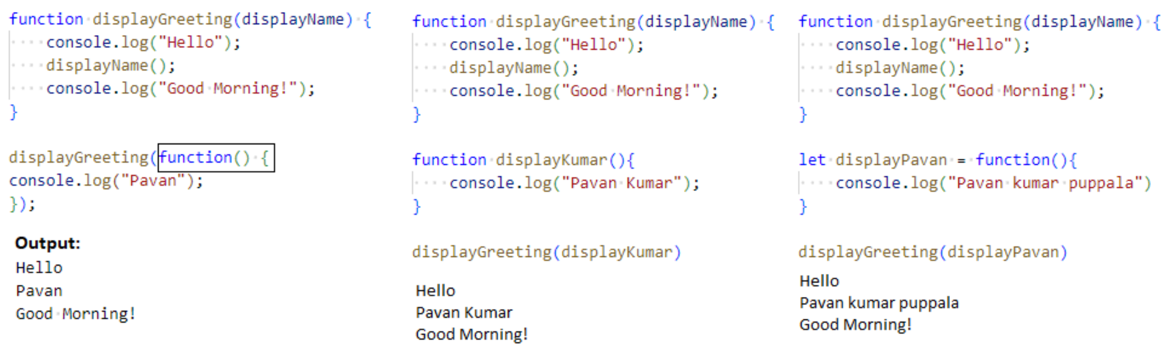
1. Callback function

- A Callback is a function that is passed as an argument to another function.
  - Passing a function as an argument
  - Passing a function name as an argument
  - Passing a function expression as an argument

In [7]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js46.png")
```

Out[7]:



```
function displayGreeting(displayName) {
    console.log("Hello");
    displayName();
    console.log("Good Morning!");
}

displayGreeting(function() {
console.log("Pavan");
});
```

Output:

```
Hello
Pavan
Good Morning!
```

```
function displayGreeting(displayName) {
    console.log("Hello");
    displayName();
    console.log("Good Morning!");
}

function displayKumar(){
    console.log("Pavan Kumar");
}

displayGreeting(displayKumar)
```

```
Hello
Pavan Kumar
Good Morning!
```

```
function displayGreeting(displayName) {
    console.log("Hello");
    displayName();
    console.log("Good Morning!");
}

let displayPavan = function(){
    console.log("Pavan kumar puppala")
}

displayGreeting(displayPavan)
```

```
Hello
Pavan kumar puppala
Good Morning!
```

Schedulers

- The Schedulers are used to schedule the execution of a callback function.
- There are different scheduler methods.
  - setInterval()
  - clearInterval()
  - setTimeout()
  - clearTimeout(), etc.

setInterval()

- The setInterval() method allows us to run a function at the specified interval of time repeatedly.
- Syntax: setInterval(function, delay);
- function - a callback function that is called repeatedly at the specified interval of time (delay).
- delay - time in milliseconds. (1 second = 1000 milliseconds)

- In the setInterval() method, the callback function repeatedly executes until the browser tab is closed or the scheduler is cancelled.
- When we call the setInterval() method, it returns a unique id. This unique Id is used to cancel the callback function execution.

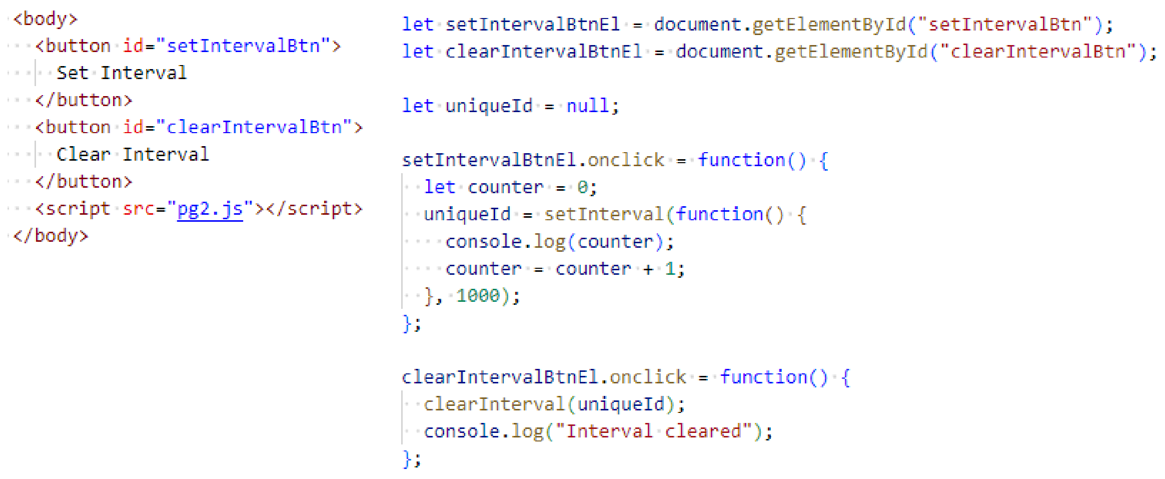
clearInterval()

- The clearInterval() method cancels a schedule previously set up by calling setInterval().
- To execute clearInterval() method, we need to pass the uniqueId returned by setInterval() as an argument.
- Syntax: clearInterval(uniqueId);

In [8]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js47.png")
```

Out[8]:



```
<body>
  <button id="setIntervalBtn">
    Set Interval
  </button>
  <button id="clearIntervalBtn">
    Clear Interval
  </button>
  <script src="pg2.js"></script>
</body>
```

```
let setIntervalBtnEl = document.getElementById("setIntervalBtn");
let clearIntervalBtnEl = document.getElementById("clearIntervalBtn");

let uniqueId = null;

setIntervalBtnEl.onclick = function() {
  let counter = 0;
  uniqueId = setInterval(function() {
    console.log(counter);
    counter = counter + 1;
  }, 1000);
};

clearIntervalBtnEl.onclick = function() {
  clearInterval(uniqueId);
  console.log("Interval cleared");
};
```

setTimeout()

- The setTimeout() method executes a function after the specified time.
- Syntax: setTimeout(function, delay);
- function - a callback function that is called after the specified time (delay).
- delay - time in milliseconds.

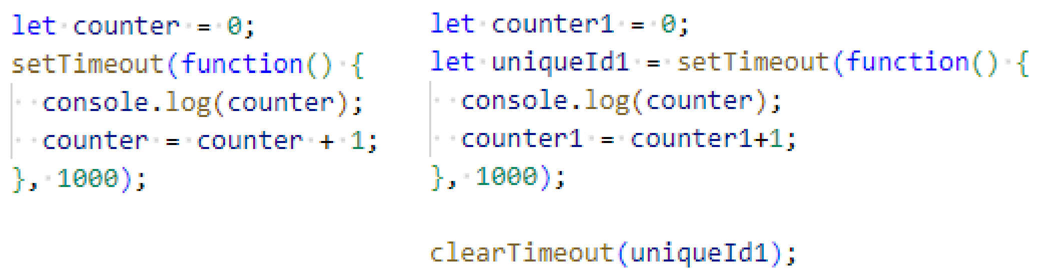
clearTimeout()

- We can cancel the setTimeout() before it executes the callback function using the clearTimeout() method.
- To execute clearTimeout(), we need to pass the uniqueId returned by setTimeout() as an argument.
- Syntax: clearTimeout(uniqueId);

In [9]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/js48.png")
```

Out[9]:



```javascript
let counter = 0;
setTimeout(function() {
  console.log(counter);
  counter = counter + 1;
}, 1000);
```

```javascript
let counter1 = 0;
let uniqueId1 = setTimeout(function() {
  console.log(counter);
  counter1 = counter1+1;
}, 1000);

clearTimeout(uniqueId1);
```

Event Listeners and More Events

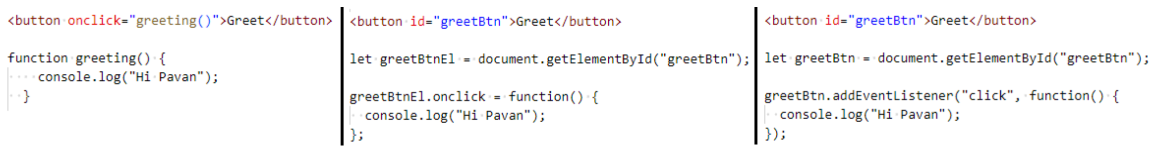
1. Event Listeners

- JavaScript offers three ways to add an Event Listener to a DOM element.
  - Inline event listeners
  - onevent listeners
  - addEventListener()

In [10]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/js49.png")
```

Out[10]:



```html
<button onclick="greeting()">Greet</button>

function greeting() {
    console.log("Hi Pavan");
  }
```

```javascript
<button id="greetBtn">Greet</button>

let greetBtnEl = document.getElementById("greetBtn");

greetBtnEl.onclick = function() {
  console.log("Hi Pavan");
};
```

```javascript
<button id="greetBtn">Greet</button>

let greetBtn = document.getElementById("greetBtn");

greetBtn.addEventListener("click", function() {
  console.log("Hi Pavan");
});
```

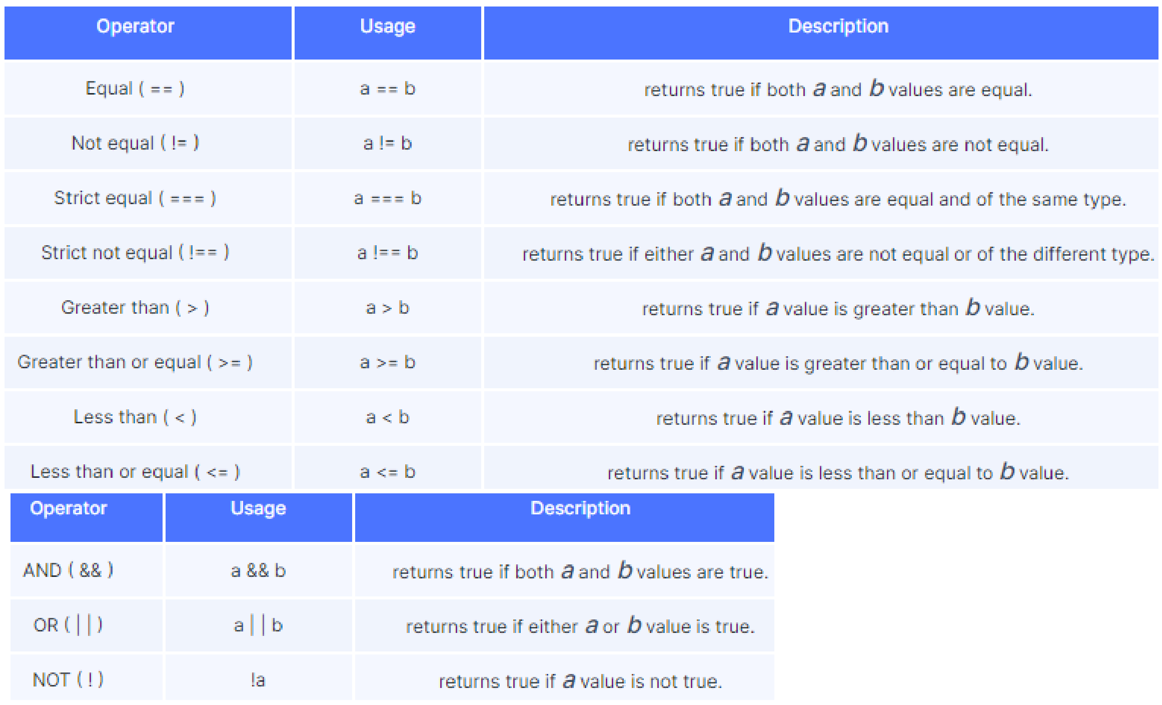
2. Operators

2.1 Comparison Operators

In [11]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js50.png")
```

Out[11]:



Operator	Usage	Description
Equal (==)	a == b	returns true if both *a* and *b* values are equal.
Not equal (!=)	a != b	returns true if both *a* and *b* values are not equal.
Strict equal (===)	a === b	returns true if both *a* and *b* values are equal and of the same type.
Strict not equal (!==)	a !== b	returns true if either *a* and *b* values are not equal or of the different type.
Greater than (>)	a > b	returns true if *a* value is greater than *b* value.
Greater than or equal (>=)	a >= b	returns true if *a* value is greater than or equal to *b* value.
Less than (<)	a < b	returns true if *a* value is less than *b* value.
Less than or equal (<=)	a <= b	returns true if *a* value is less than or equal to *b* value.

Operator	Usage	Description
AND (&&)	a && b	returns true if both *a* and *b* values are true.
OR (||)	a || b	returns true if either *a* or *b* value is true.
NOT (!)	!a	returns true if *a* value is not true.

3. More Events

- Events are the actions by which the user or browser interact with HTML elements.
- There are different types of events.
  - Keyboard Events
  - Mouse Events
  - Touch Events, and many more.

3.1 Keyboard Events

- Keyboard Event is the user interaction with the keyboard.
- The keyboard events are
  - keydown
  - keyup

3.1.1 Keydown event

- The keydown event occurs when a key on the keyboard is pressed.
- Syntax: element.addEventListener("keydown", function);

3.1.2 Keyup event

- The keyup event occurs when a key on the keyboard is released.
- Syntax: element.addEventListener("keyup", function);

3.2 Event Object

- Whenever an event happens, the browser creates an event object.
- It consists of information about the event that has happened.
- It consists of many properties and methods.
  - type
  - target
  - key
  - timeStamp
  - stopPropagation, and many more.

3.2.1 Properties & Methods

- For any event, event-specific properties and methods will be present.
- For Example, The keydown event has key property, whereas the onclick event doesn't have it.
- event.type : The event.type property contains the type of event occurred like click, keydown, etc.
- event.target : The event.target property contains the HTML element that triggered the event.
- event.key : The event.key property contains the value of the key pressed by the user.

Hypertext Transfer Protocol (HTTP)

1. Web Resources

- A Web Resource is any data that can be obtained via internet.
- A resource can be
  - HTML document
  - CSS document
  - JSON Data or Plain text
  - Image, Video, etc.

2. Uniform Resource Locator (URL)

- URL is a text string that specifies where a resource can be found on the internet.
- We can access web resources using the URL.
- Syntax: protocol://domainName/path?query-parameters
- In the URL http://www.flipkart.com/watches?type=digital&rating=4 (http://www.flipkart.com/watches?type=digital&rating=4),
  - http is a Protocol
  - www.flipkart.com is a Domain Name
  - /watches is a Path
  - type=digital&rating=4 is the Query Parameters

2.1 Protocol

- A protocol is a standard set of rules that allow electronic devices to communicate with each other.
- There are different types of protocols.
  - Hypertext Transfer Protocol (HTTP)
  - Hypertext Transfer Protocol Secure (HTTPS)
  - Web Sockets, etc.

2.1.1 HTTP

- The Hypertext Transfer Protocol (HTTP), is a protocol used to transfer resources over the web.
- Examples: Internet forums, Educational sites, etc.
- HTTP Request: Message sent by the client
- HTTP Response: Message sent by the server

2.1.2 HTTPS

- In Hypertext Transfer Protocol Secure (HTTPS), information is transferred in an encrypted format and provides secure communication.
- Examples: Banking Websites, Payment gateway, Login Pages, Emails and Corporate Sector Websites, etc.

2.2 Domain Name

- It indicates which Web server is being requested.

2.3 Path

- The path is to identify the resources on the server.
- Examples:
  - /watches in http://www.flipkart.com/watches (http://www.flipkart.com/watches)
  - /electronics/laptops/gaming in http://www.flipkart.com/electronics/laptops/gaming (http://www.flipkart.com/electronics/laptops/gaming)

2.4 Query Parameters

- Query parameters add some criteria to the request for the resource.
- Multiple query parameters can be added by using an & (ampersand) symbol.
- For example : http://www.flipkart.com/watches?type=digital&rating=4 (http://www.flipkart.com/watches?type=digital&rating=4)

3. HTTP

3.1 HTTP Requests

- HTTP requests are messages sent by the client to initiate an action on the server.
- HTTP request includes
  - Start Line
  - Headers
  - Body

3.1.1 Start Line

- A Start Line specifies a
  - URL
  - HTTP Method
  - HTTP Version

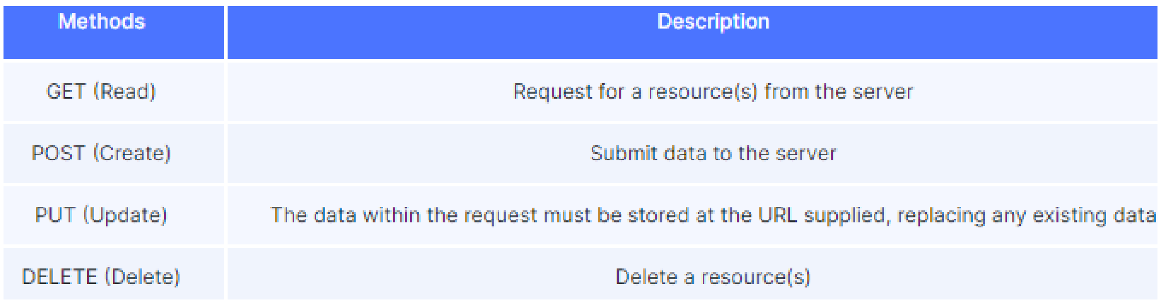
HTTP Methods

- The HTTP Request methods indicate the desired action to be performed for a given resource.

In [12]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js51.png")
```

Out[12]:



Methods	Description
GET (Read)	Request for a resource(s) from the server
POST (Create)	Submit data to the server
PUT (Update)	The data within the request must be stored at the URL supplied, replacing any existing data
DELETE (Delete)	Delete a resource(s)

3.1.2 Headers

- HTTP Headers let the client and the server to pass additional information with an HTTP request or response.

3.1.3 Body

- We place the data in the Request body when we want to send data to the server.
- For example, form details filled by the user.
- HTTP Requests

3.2 HTTP Responses

- HTTP responses are messages sent by the server as an answer to the clients request.
- HTTP Response includes
  - Status Line
  - Headers
  - Body

3.2.1 Status Line

- A Status line specifies a
  - HTTP version
  - Status code
  - Status text

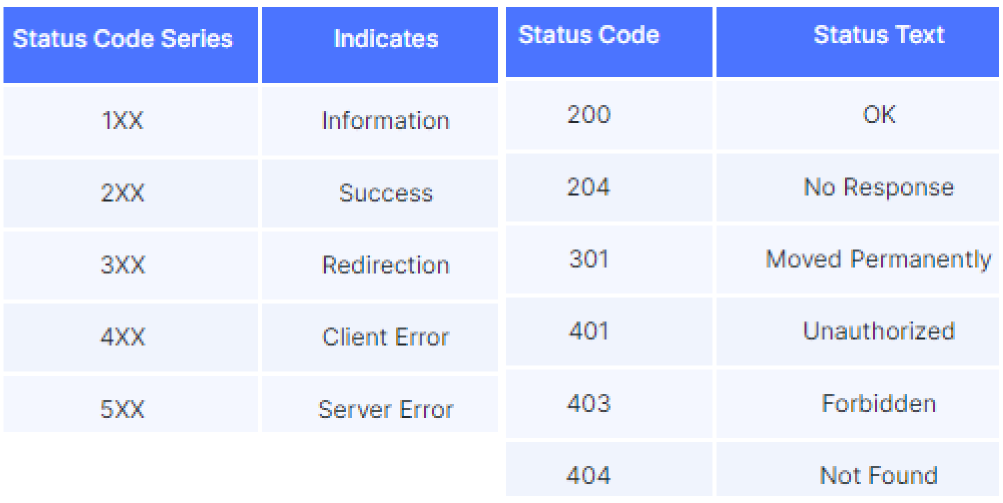
Status code

- Status codes Indicate whether an HTTP request has been successfully completed or not
- 200 (Success) - Indicates that the request has succeeded
- 201 (Created) - The request has succeeded and a new resource has been created as a result

In [13]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js52.png")
```

Out[13]:



Status Code Series	Indicates
1XX	Information
2XX	Success
3XX	Redirection
4XX	Client Error
5XX	Server Error

Status Code	Status Text
200	OK
204	No Response
301	Moved Permanently
401	Unauthorized
403	Forbidden
404	Not Found

3.2.2 Body

- Response Body contains the resource data that was requested by the client.

8. How to make the HTTP Requests using JS

1. Fetch

- The fetch() method is used to fetch resources across the Internet.
- Syntax: fetch(URL, OPTIONS)
- URL - URL of the resource
- OPTIONS - Request Configuration

1.1 Request Configuration

- We can configure the fetch request with many options like,
  - Request Method
  - Headers
  - Body
  - Credentials
  - Cache, etc.

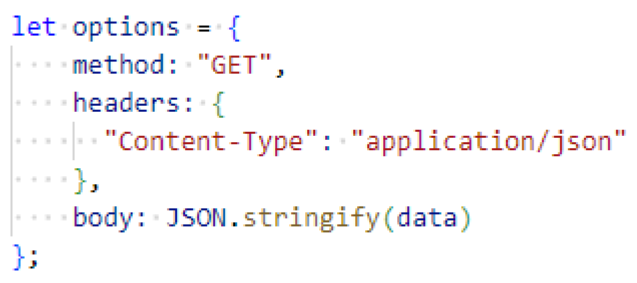
We can configure a request by passing an options object with required properties and their values.

- For example,

In [14]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js53.png")
```

Out[14]:



```
let options = {
    method: "GET",
    headers: {
      "Content-Type": "application/json"
    },
    body: JSON.stringify(data)
};
```

2. Making HTTP Requests using Fetch

- The method property value in the options object can be GET, POST, PUT, DELETE, etc. The default method is GET.

2.1 GET

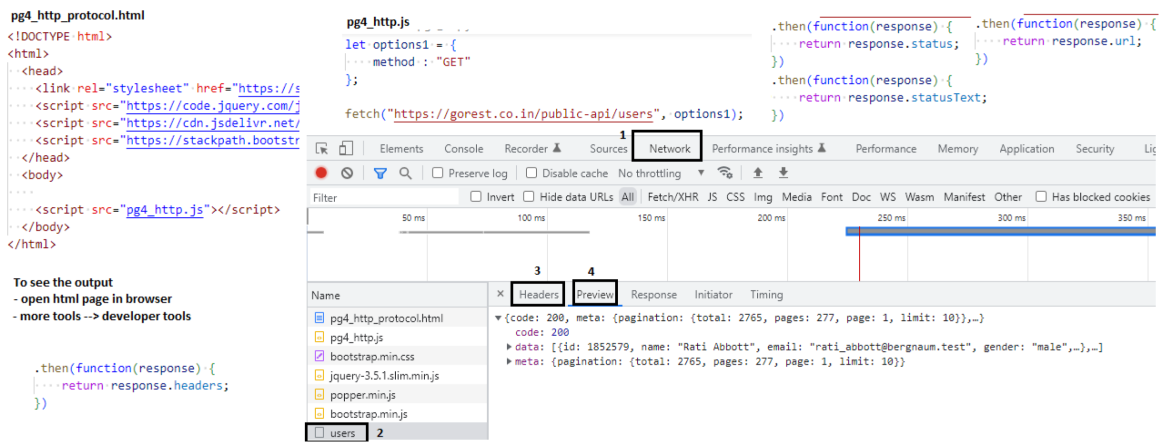
- The GET method can be used to retrieve (get) data from a specified resource.

For example,

In [20]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/js54.png")
```

Out[20]:



pg4_http_protocol.html

```html
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="https://s
    <script src="https://code.jquery.com/j
    <script src="https://cdn.jsdelivr.net/
    <script src="https://stackpath.bootstr
  </head>
  <body>

    <script src="pg4_http.js"></script>
  </body>
</html>
```

To see the output
- open html page in browser
- more tools --> developer tools

```javascript
.then(function(response) {
    return response.headers;
})
```

pg4_http.js

```javascript
let options1 = {
    method : "GET"
};

fetch("https://gorest.co.in/public-api/users", options1);
```

```javascript
.then(function(response) {
    return response.status;
})
.then(function(response) {
    return response.statusText;
})
```

```javascript
.then(function(response) {
    return response.url;
})
```

```
▼{code: 200, meta: {pagination: {total: 2765, pages: 277, page: 1, limit: 10}},…}
   code: 200
  ▶data: [{id: 1852579, name: "Rati Abbott", email: "rati_abbott@bergnaum.test", gender: "male",…},…]
  ▶meta: {pagination: {total: 2765, pages: 277, page: 1, limit: 10}}
```

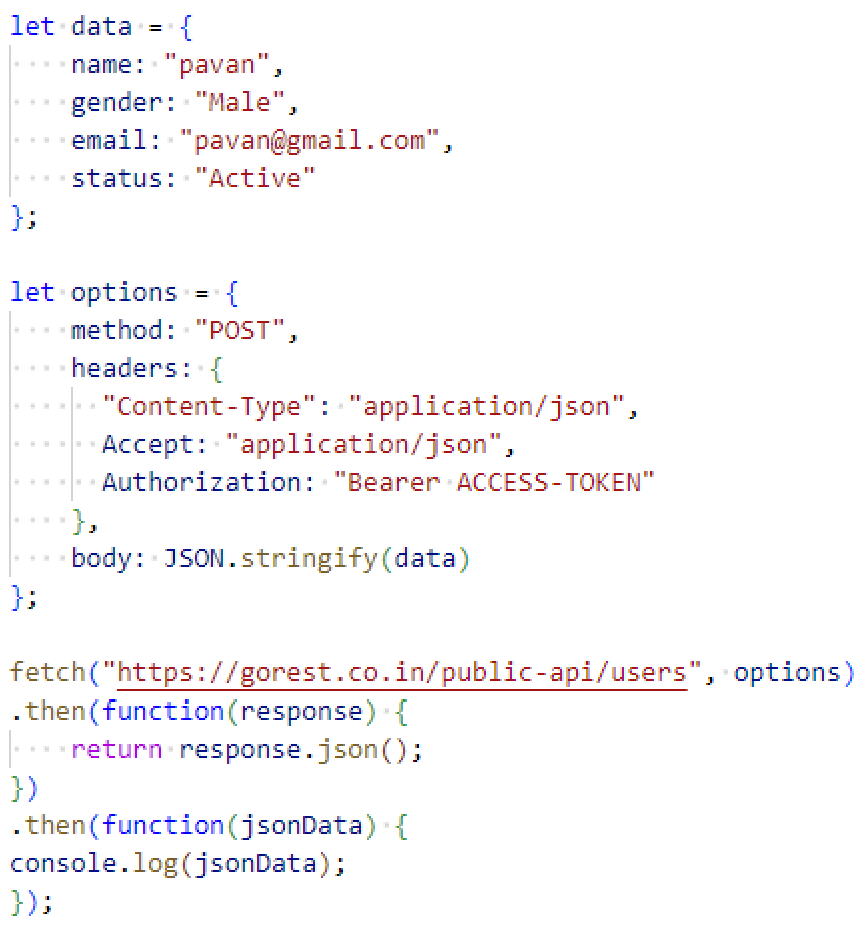
2.2 POST

- The POST method can be used to send data to the server.

In [16]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/js55.png")
```

Out[16]:



```javascript
let data = {
    name: "pavan",
    gender: "Male",
    email: "pavan@gmail.com",
    status: "Active"
};

let options = {
    method: "POST",
    headers: {
        "Content-Type": "application/json",
        Accept: "application/json",
        Authorization: "Bearer ACCESS-TOKEN"
    },
    body: JSON.stringify(data)
};

fetch("https://gorest.co.in/public-api/users", options)
.then(function(response) {
    return response.json();
})
.then(function(jsonData) {
console.log(jsonData);
});
```

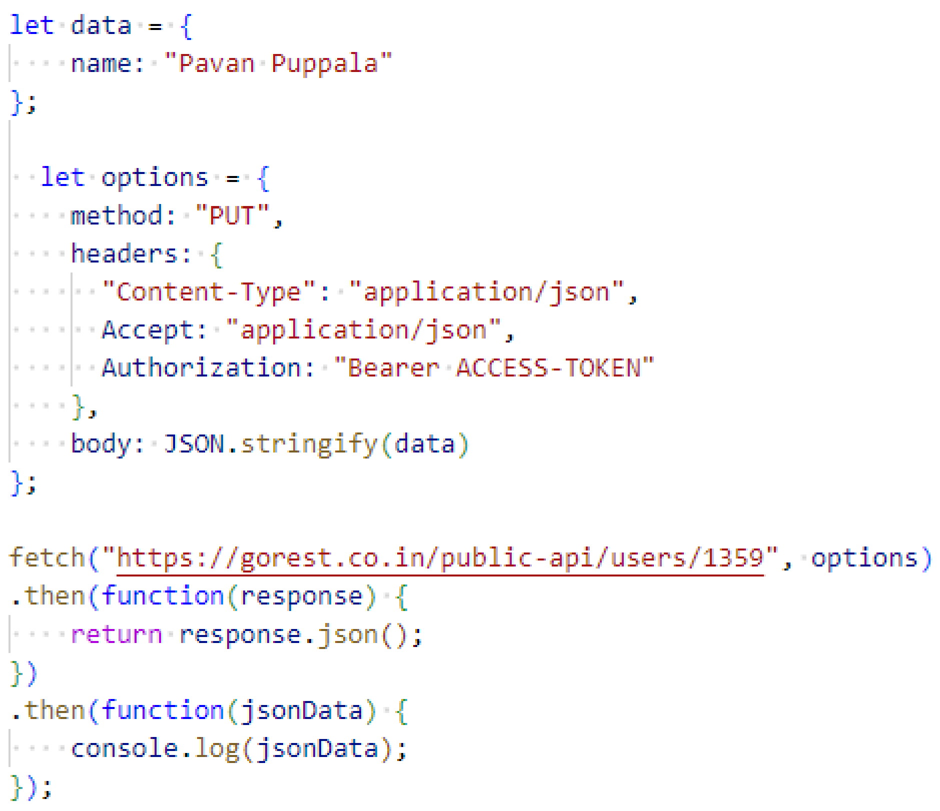
2.3 PUT

- The PUT method can be used to update the existing resource.

In [17]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js56.png")
```

Out[17]:



```
let data = {
    name: "Pavan Puppala"
};

  let options = {
    method: "PUT",
    headers: {
      "Content-Type": "application/json",
      Accept: "application/json",
      Authorization: "Bearer ACCESS-TOKEN"
    },
    body: JSON.stringify(data)
};

fetch("https://gorest.co.in/public-api/users/1359", options)
.then(function(response) {
    return response.json();
})
.then(function(jsonData) {
    console.log(jsonData);
});
```

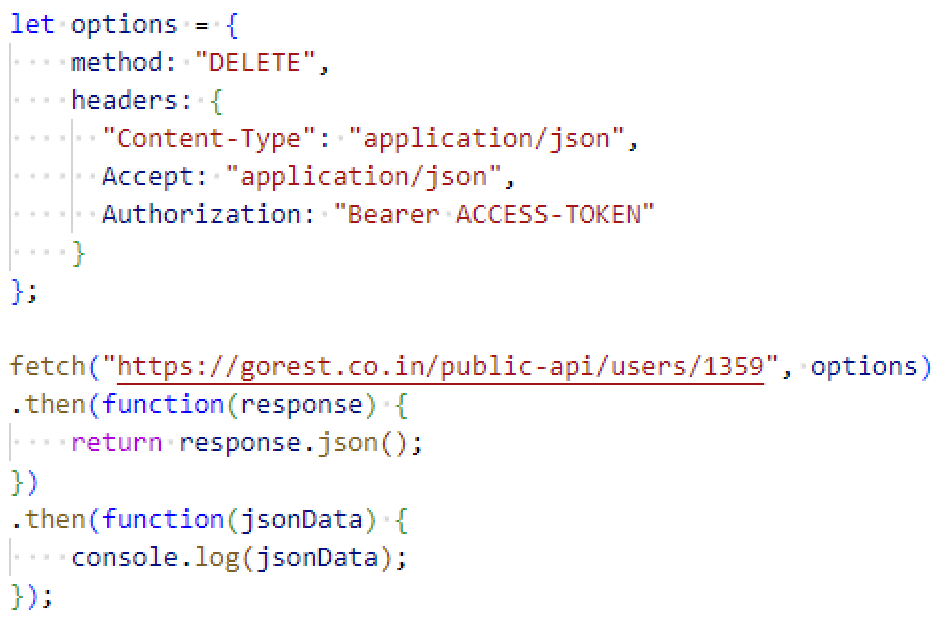
2.4 DELETE

- The DELETE method can be used to delete the specified resource.

In [18]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js57.png")
```

Out[18]:



```
let options = {
    method: "DELETE",
    headers: {
      "Content-Type": "application/json",
      Accept: "application/json",
      Authorization: "Bearer ACCESS-TOKEN"
    }
};

fetch("https://gorest.co.in/public-api/users/1359", options)
.then(function(response) {
    return response.json();
})
.then(function(jsonData) {
    console.log(jsonData);
});
```

3. HTTP Response Object Properties and Methods

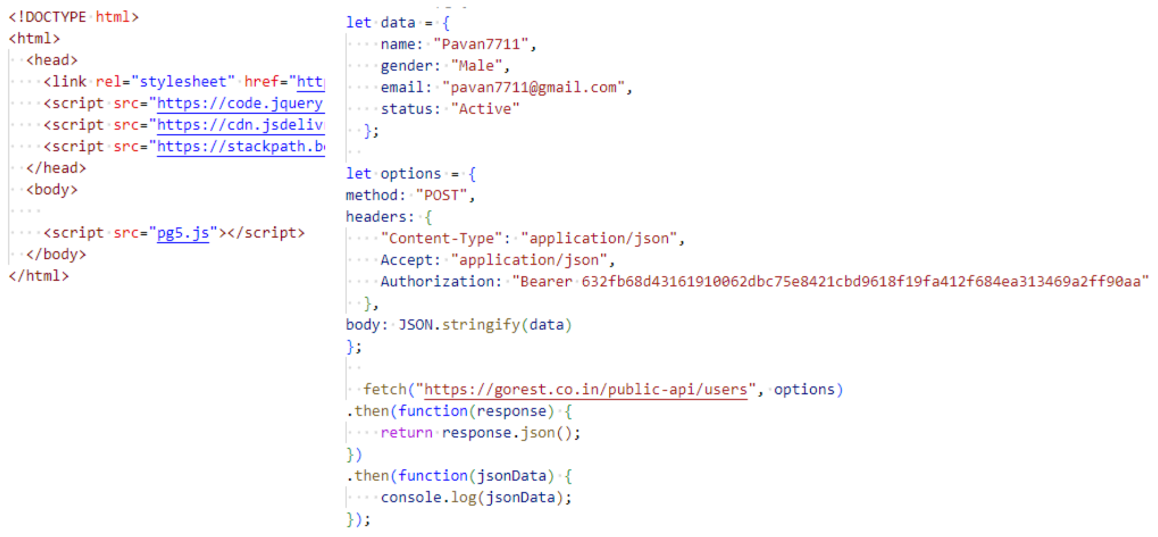
- Response Object provides multiple properties to give more information about the HTTP Response.
  - status (number) - HTTP status code
  - statusText (string) - Status text as reported by the server, e.g. "Unauthorized"
  - headers
  - url
  - text() - Getting text from response
  - json() - Parses the response as JSON

For example,

In [21]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js58.png")
```

Out[21]:



```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="htt
    <script src="https://code.jquery
    <script src="https://cdn.jsdeliv
    <script src="https://stackpath.b
  </head>
  <body>

    <script src="pg5.js"></script>
  </body>
</html>
```

```
let data = {
    name: "Pavan7711",
    gender: "Male",
    email: "pavan7711@gmail.com",
    status: "Active"
  };

let options = {
method: "POST",
headers: {
    "Content-Type": "application/json",
    Accept: "application/json",
    Authorization: "Bearer 632fb68d43161910062dbc75e8421cbd9618f19fa412f684ea313469a2ff90aa"
  },
body: JSON.stringify(data)
};

  fetch("https://gorest.co.in/public-api/users", options)
.then(function(response) {
    return response.json();
})
.then(function(jsonData) {
    console.log(jsonData);
});
```

Wikipedia Search

1. HTML Input Element

1.1 Search

- The HTML input element with the type search is designed for the user to enter the search queries.

2. Bootstrap Components

2.1 Spinner

- The Spinners can be used to show the loading state of the page.

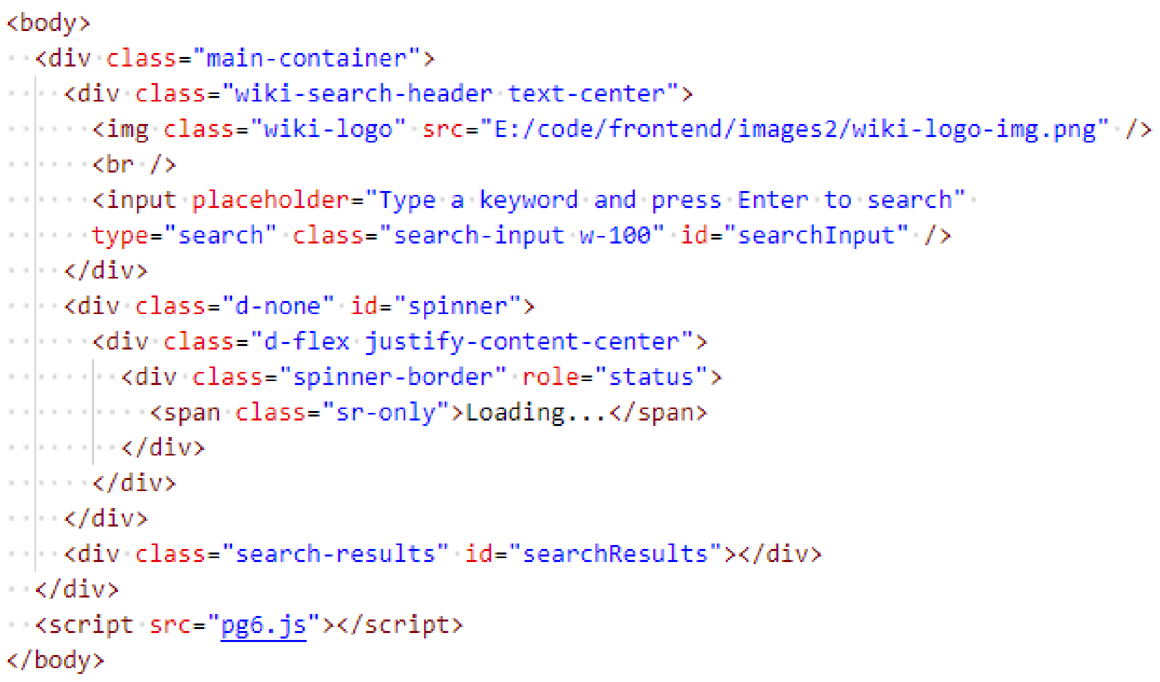
Steps:

- Add EventListener and get the search text
- make http request and get the search results
- display search results

In [1]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/js59.png")
```

Out[1]:

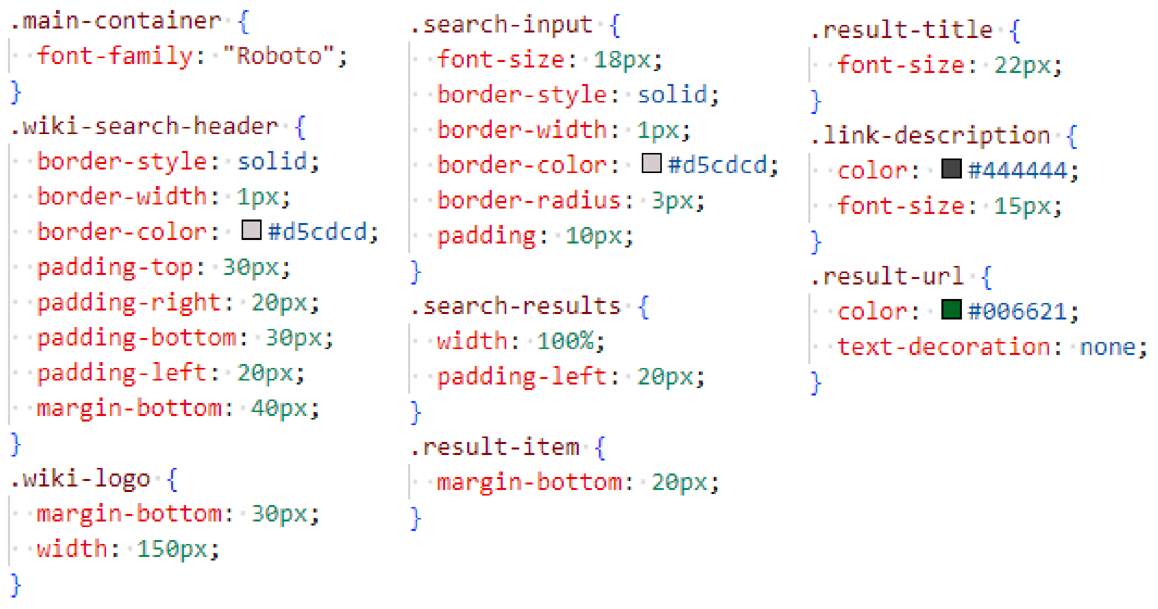


```html
<body>
  <div class="main-container">
    <div class="wiki-search-header text-center">
      <img class="wiki-logo" src="E:/code/frontend/images2/wiki-logo-img.png" />
      <br />
      <input placeholder="Type a keyword and press Enter to search"
      type="search" class="search-input w-100" id="searchInput" />
    </div>
    <div class="d-none" id="spinner">
      <div class="d-flex justify-content-center">
        <div class="spinner-border" role="status">
          <span class="sr-only">Loading...</span>
        </div>
      </div>
    </div>
    <div class="search-results" id="searchResults"></div>
  </div>
  <script src="pg6.js"></script>
</body>
```

In [3]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/js61.png")
```

Out[3]:

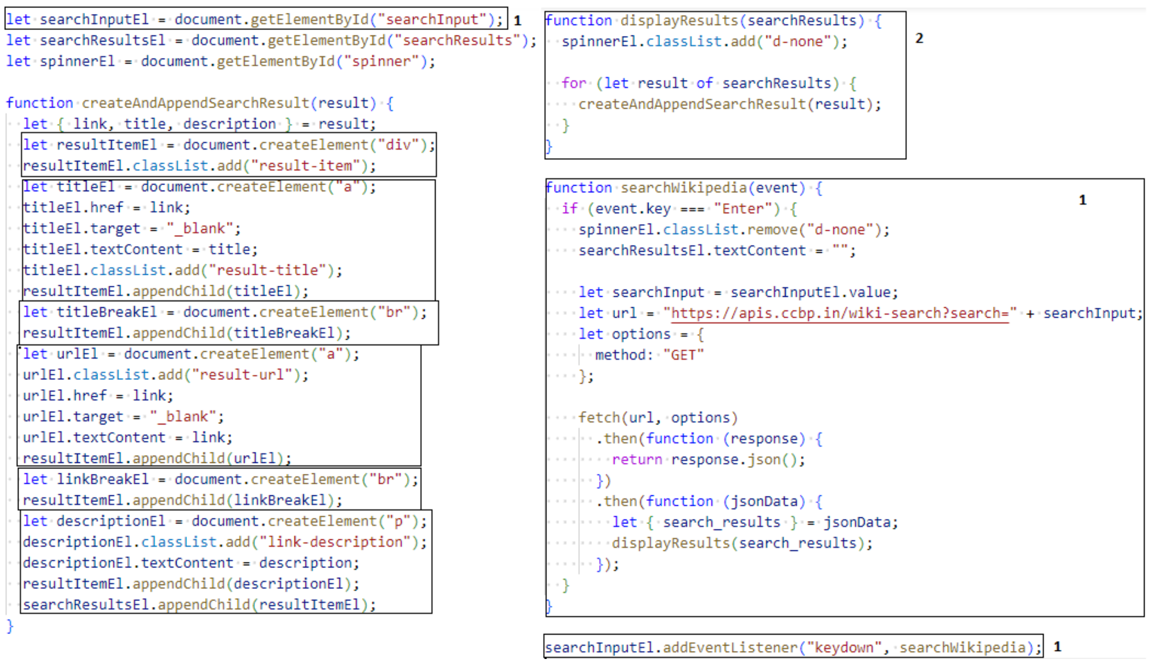


```css
.main-container {
  font-family: "Roboto";
}
.wiki-search-header {
  border-style: solid;
  border-width: 1px;
  border-color: #d5cdcd;
  padding-top: 30px;
  padding-right: 20px;
  padding-bottom: 30px;
  padding-left: 20px;
  margin-bottom: 40px;
}
.wiki-logo {
  margin-bottom: 30px;
  width: 150px;
}
.search-input {
  font-size: 18px;
  border-style: solid;
  border-width: 1px;
  border-color: #d5cdcd;
  border-radius: 3px;
  padding: 10px;
}
.search-results {
  width: 100%;
  padding-left: 20px;
}
.result-item {
  margin-bottom: 20px;
}
.result-title {
  font-size: 22px;
}
.link-description {
  color: #444444;
  font-size: 15px;
}
.result-url {
  color: #006621;
  text-decoration: none;
}
```

In [2]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js60.png")
```

Out[2]:



```
let searchInputEl = document.getElementById("searchInput"); 1
let searchResultsEl = document.getElementById("searchResults");
let spinnerEl = document.getElementById("spinner");

function createAndAppendSearchResult(result) {
  let { link, title, description } = result;
  let resultItemEl = document.createElement("div");
  resultItemEl.classList.add("result-item");
  let titleEl = document.createElement("a");
  titleEl.href = link;
  titleEl.target = "_blank";
  titleEl.textContent = title;
  titleEl.classList.add("result-title");
  resultItemEl.appendChild(titleEl);
  let titleBreakEl = document.createElement("br");
  resultItemEl.appendChild(titleBreakEl);
  let urlEl = document.createElement("a");
  urlEl.classList.add("result-url");
  urlEl.href = link;
  urlEl.target = "_blank";
  urlEl.textContent = link;
  resultItemEl.appendChild(urlEl);
  let linkBreakEl = document.createElement("br");
  resultItemEl.appendChild(linkBreakEl);
  let descriptionEl = document.createElement("p");
  descriptionEl.classList.add("link-description");
  descriptionEl.textContent = description;
  resultItemEl.appendChild(descriptionEl);
  searchResultsEl.appendChild(resultItemEl);
}
```

```
function displayResults(searchResults) {        2
  spinnerEl.classList.add("d-none");

  for (let result of searchResults) {
    createAndAppendSearchResult(result);
  }
}

function searchWikipedia(event) {        1
  if (event.key === "Enter") {
    spinnerEl.classList.remove("d-none");
    searchResultsEl.textContent = "";

    let searchInput = searchInputEl.value;
    let url = "https://apis.ccbp.in/wiki-search?search=" + searchInput;
    let options = {
      method: "GET"
    };

    fetch(url, options)
      .then(function (response) {
        return response.json();
      })
      .then(function (jsonData) {
        let { search_results } = jsonData;
        displayResults(search_results);
      });
  }
}

searchInputEl.addEventListener("keydown", searchWikipedia); 1
```

9. Forms

1. HTML Forms

- The HTML Forms can be used to collect data from the user.
- Forms are of different kinds:
  - Login/Sign in Form
  - Registration Form
  - Contact Us Form, etc.

1.1 HTML Form Element

- The HTML form element can be used to create HTML forms. It is a container that can contain different types of Input elements like Text Fields, Checkboxes, etc.

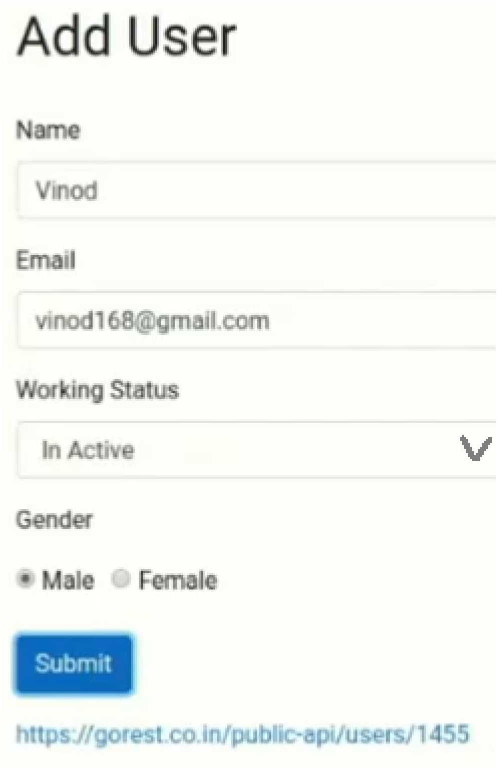
Note:

- Whenever we click a button or press Enter key while editing any input field in the form, the submit event will be triggered.

In [2]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js62.png")
```

Out[2]:



Add User

Name

Vinod

Email

vinod168@gmail.com

Working Status

In Active

Gender

● Male ○ Female

https://gorest.co.in/public-api/users/1455

2. Event Object Methods

- whenever an event happens, the browser creates an Even Object
- it consists information about the event that has happened

2.1 preventDefault

- The preventDefault() method prevents the occurrence of default action.
- Here in the form, it prevents the default behaviour of the submit event.

let myFormEl = document.getElementById("myForm");

myFormEl.addEventListener("submit", function(event) { event.preventDefault(); });

3. Event Types

- There are different types of events.
  - Keyboard Events
  - Mouse Events
  - Touch Events
  - Form Events, etc.

3.1 Form Events

- A Form Event is an event that can occur within a form.
- Some of the form events are:
  - blur
  - focus
  - change, etc.

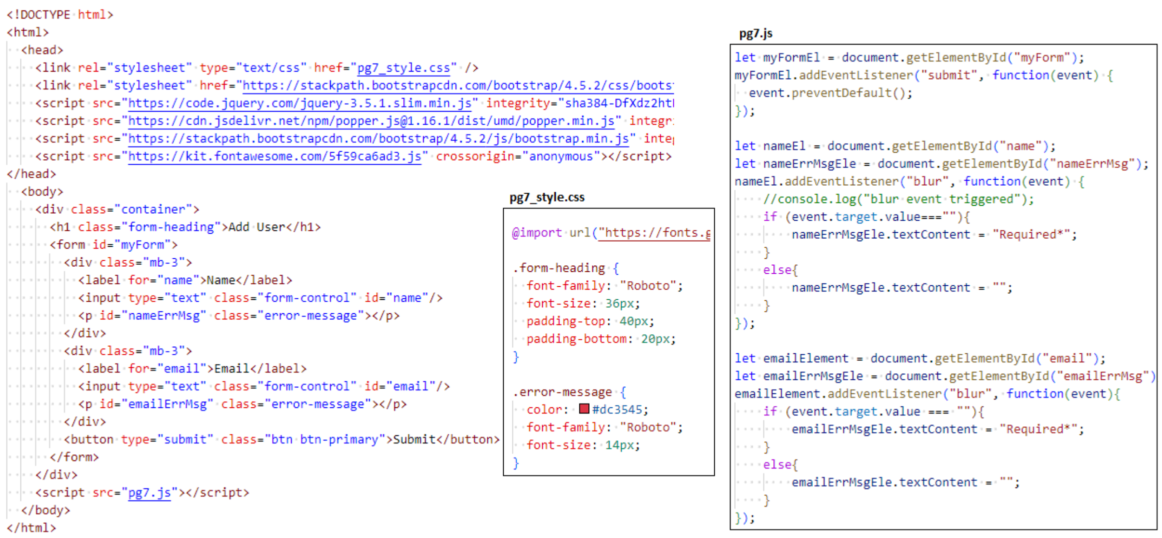
3.1.1 Blur Event

- The blur event happens when an HTML element has lost focus.

In [4]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/js63.png")
```

Out[4]:



```html
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="pg7_style.css" />
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/boots
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htI
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integr
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" inte
    <script src="https://kit.fontawesome.com/5f59ca6ad3.js" crossorigin="anonymous"></script>
</head>
  <body>
    <div class="container">
      <h1 class="form-heading">Add User</h1>
      <form id="myForm">
        <div class="mb-3">
          <label for="name">Name</label>
          <input type="text" class="form-control" id="name"/>
          <p id="nameErrMsg" class="error-message"></p>
        </div>
        <div class="mb-3">
          <label for="email">Email</label>
          <input type="text" class="form-control" id="email"/>
          <p id="emailErrMsg" class="error-message"></p>
        </div>
        <button type="submit" class="btn btn-primary">Submit</button>
      </form>
    </div>
    <script src="pg7.js"></script>
  </body>
</html>
```

pg7_style.css

```css
@import url("https://fonts.g

.form-heading {
  font-family: "Roboto";
  font-size: 36px;
  padding-top: 40px;
  padding-bottom: 20px;
}

.error-message {
  color: #dc3545;
  font-family: "Roboto";
  font-size: 14px;
}
```

pg7.js

```javascript
let myFormEl = document.getElementById("myForm");
myFormEl.addEventListener("submit", function(event) {
  event.preventDefault();
});

let nameEl = document.getElementById("name");
let nameErrMsgEle = document.getElementById("nameErrMsg");
nameEl.addEventListener("blur", function(event) {
    //console.log("blur event triggered");
    if (event.target.value===""){
        nameErrMsgEle.textContent = "Required*";
    }
    else{
        nameErrMsgEle.textContent = "";
    }
});

let emailElement = document.getElementById("email");
let emailErrMsgEle = document.getElementById("emailErrMsg")
emailElement.addEventListener("blur", function(event){
    if (event.target.value === ""){
        emailErrMsgEle.textContent = "Required*";
    }
    else{
        emailErrMsgEle.textContent = "";
    }
});
```

1. HTML Select Element

- The HTML select element is used to create a drop-down list.

1.1 HTML Option Element

- The HTML option element is used to create the menu option of a drop-down list.
- The text content of the HTML option element is used as a label.

1.1.1 The value Attribute

- Every HTML option element should contain the HTML value attribute.

2. HTML Input Element

2.1 Radio

- The HTML input radio element is used to select one option among a list of given options.

2.1.1 HTML name attribute

- The HTML name Attribute specifies the name for an HTML Element.

2.1.2 Radio Group

- All the radio buttons with same name collectively called as a radio group.
- We can select only one radio button within a radio group.

3. Boolean Attributes

- For the HTML Boolean attributes, we only specify the name of the HTML attribute.
- The presence of a boolean attribute represents the true value, and the absence represents the false value.
  - selected
  - checked
  - disabled
  - readonly
  - default

3.1 HTML selected attribute

- The selected attribute specifies that an option should be pre-selected when the page loads.

3.2 HTML checked attribute

- The checked attribute specifies that an input element should be pre-selected (checked) when the page loads.

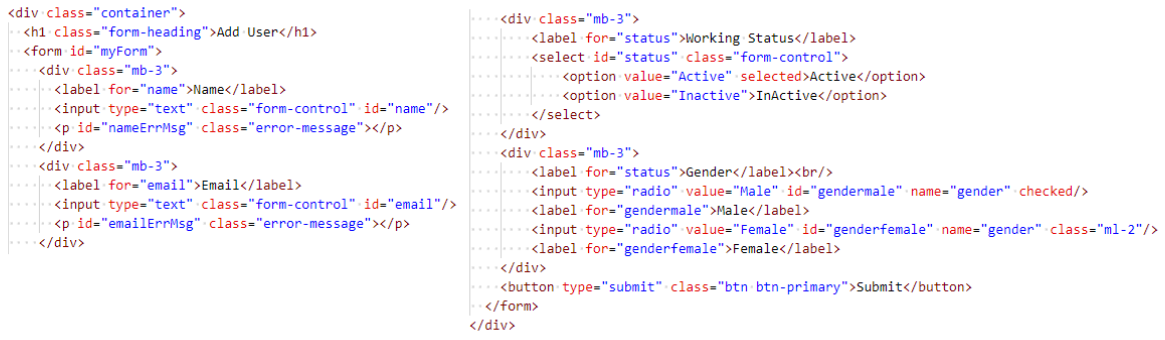
steps:

- create a form in html file with two fields(name, email)
- To overcome the event use preventDefault() in js file
- Now add the field level validations
- create drop down (selected)
- create radio button (checked)
- maintain all the form data in the object
- send the form data using Post method on submit event
- form level validations

In [6]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js64.png")
```

Out[6]:



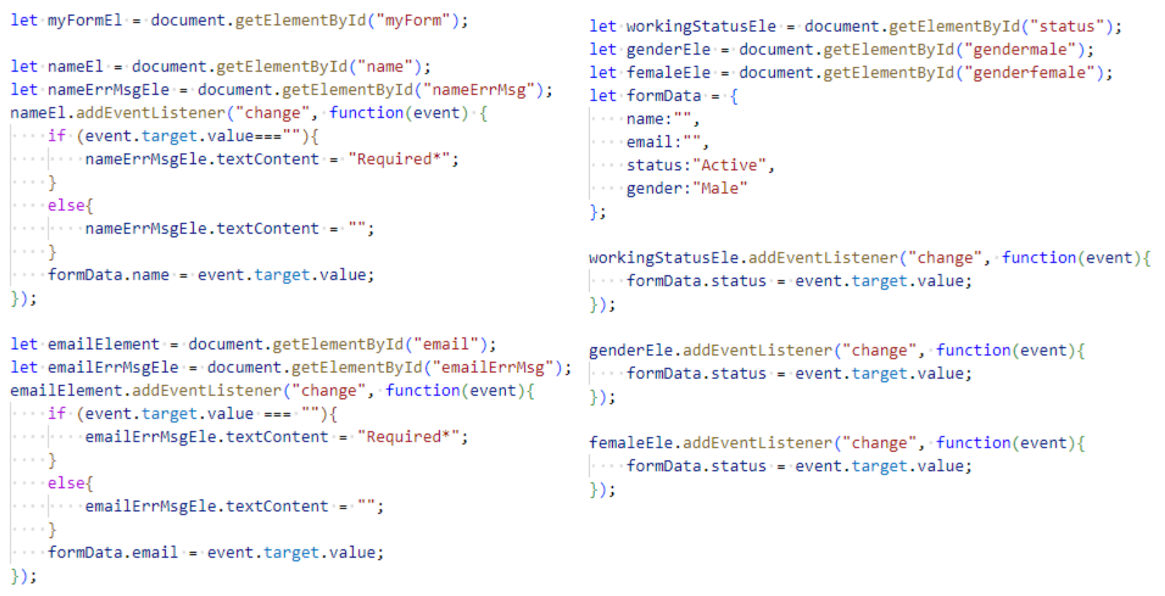
```
<div class="container">
  <h1 class="form-heading">Add User</h1>
  <form id="myForm">
    <div class="mb-3">
      <label for="name">Name</label>
      <input type="text" class="form-control" id="name"/>
      <p id="nameErrMsg" class="error-message"></p>
    </div>
    <div class="mb-3">
      <label for="email">Email</label>
      <input type="text" class="form-control" id="email"/>
      <p id="emailErrMsg" class="error-message"></p>
    </div>
```

```
    <div class="mb-3">
        <label for="status">Working Status</label>
        <select id="status" class="form-control">
            <option value="Active" selected>Active</option>
            <option value="Inactive">InActive</option>
        </select>
    </div>
    <div class="mb-3">
        <label for="status">Gender</label><br/>
        <input type="radio" value="Male" id="gendermale" name="gender" checked/>
        <label for="gendermale">Male</label>
        <input type="radio" value="Female" id="genderfemale" name="gender" class="ml-2"/>
        <label for="genderfemale">Female</label>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
  </form>
</div>
```

In [7]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js65.png")
```

Out[7]:



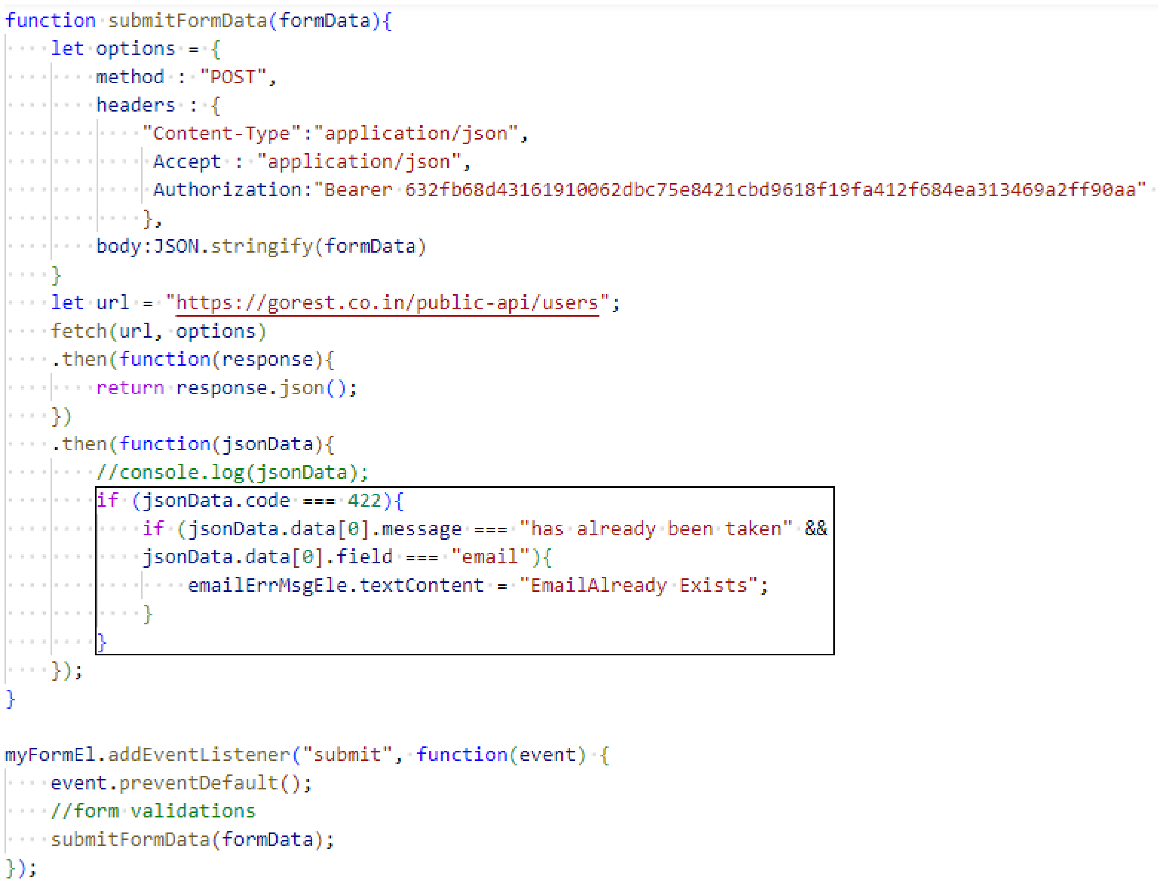
```
let myFormEl = document.getElementById("myForm");

let nameEl = document.getElementById("name");
let nameErrMsgEle = document.getElementById("nameErrMsg");
nameEl.addEventListener("change", function(event) {
    if (event.target.value===""){
        nameErrMsgEle.textContent = "Required*";
    }
    else{
        nameErrMsgEle.textContent = "";
    }
    formData.name = event.target.value;
});

let emailElement = document.getElementById("email");
let emailErrMsgEle = document.getElementById("emailErrMsg");
emailElement.addEventListener("change", function(event){
    if (event.target.value === ""){
        emailErrMsgEle.textContent = "Required*";
    }
    else{
        emailErrMsgEle.textContent = "";
    }
    formData.email = event.target.value;
});
```

```
let workingStatusEle = document.getElementById("status");
let genderEle = document.getElementById("gendermale");
let femaleEle = document.getElementById("genderfemale");
let formData = {
    name:"",
    email:"",
    status:"Active",
    gender:"Male"
};

workingStatusEle.addEventListener("change", function(event){
    formData.status = event.target.value;
});

genderEle.addEventListener("change", function(event){
    formData.status = event.target.value;
});

femaleEle.addEventListener("change", function(event){
    formData.status = event.target.value;
});
```

In [9]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/js66.png")
```

Out[9]:

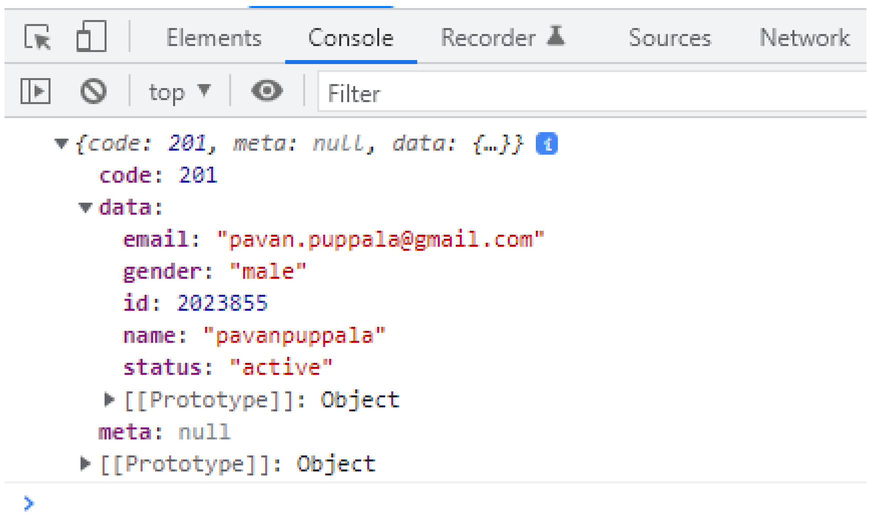


```javascript
function submitFormData(formData){
    let options = {
        method : "POST",
        headers : {
            "Content-Type":"application/json",
            Accept : "application/json",
            Authorization:"Bearer 632fb68d43161910062dbc75e8421cbd9618f19fa412f684ea313469a2ff90aa"
        },
        body:JSON.stringify(formData)
    }
    let url = "https://gorest.co.in/public-api/users";
    fetch(url, options)
    .then(function(response){
        return response.json();
    })
    .then(function(jsonData){
        //console.log(jsonData);
        if (jsonData.code === 422){
            if (jsonData.data[0].message === "has already been taken" &&
            jsonData.data[0].field === "email"){
                emailErrMsgEle.textContent = "EmailAlready Exists";
            }
        }
    });
}

myFormEl.addEventListener("submit", function(event) {
    event.preventDefault();
    //form validations
    submitFormData(formData);
});
```

In [10]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/js67.png")
```

Out[10]:



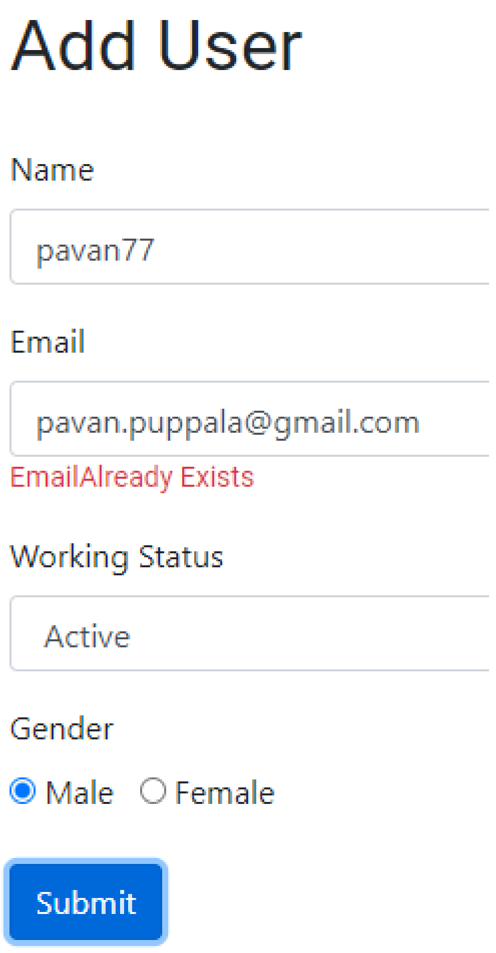
```
▼{code: 201, meta: null, data: {…}}
   code: 201
  ▼data:
     email: "pavan.puppala@gmail.com"
     gender: "male"
     id: 2023855
     name: "pavanpuppala"
     status: "active"
    ▶[[Prototype]]: Object
   meta: null
  ▶[[Prototype]]: Object
>
```

What if user enter email which is already exists

In [11]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js68.png")
```

Out[11]:



Add User

Name

pavan77

Email

pavan.puppala@gmail.com

EmailAlready Exists

Working Status

Active

Gender

[✓ Male] Female