⊙ local storage

→ It allows web applications to store data locally within the user's Browsers

→ Local storage is just like storage object

→ Data can be stored in the form of key-val pairs

→ value should always be a "string"

key	val
name	xyz
gender	male
city	vzg

→ To access & work with local storage, we have methods:

- setItem()
- getItem()
- clear()
- removeItem()

setItem()

syntax `localStorage.setItem("key", "value");`

>>JS
```
localStorage.setItem("name", "Rahul");
    "      "     ("gender", "male");
```

To check whether stored properly or not,

- go run JS
- right side o/p space click & go to inspect
- go to Application
- left side there will be storages, click on local storage
- then we can see website URL

getItem()

syntax `localStorage.getItem("key");`

>>
```
let name = localStorage.getItem("name");
let gender =    "       "    ("gender");
console.log(name)          o/p Rahul
  "   "  (gender)              male.
```

* If we want to access any key which we didn't, then it return "null" (eg. occupation)

* Values

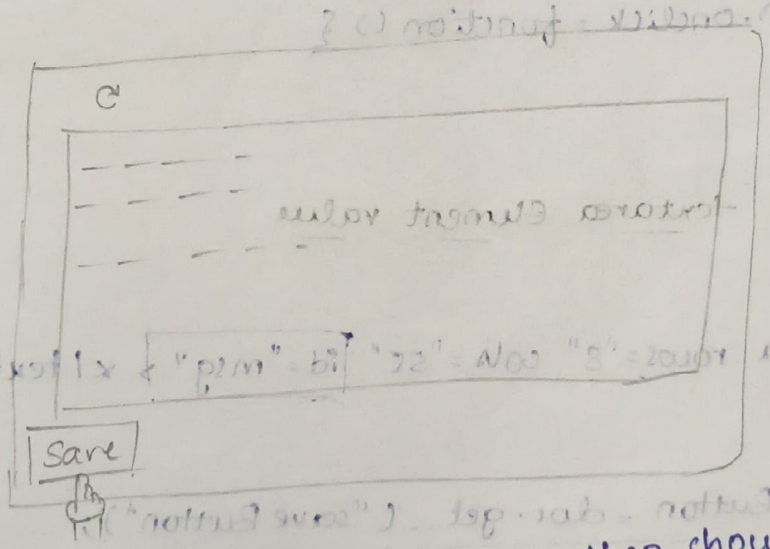
null

→ we use null in situation where we intentionally want a variable to not have a value yet.

>> Ex let selectedColor = null;
console.log (selectedColor);
console.log (typeof (selectedColor));

o/p null
object

Example



When we save text, wt all present till then should be saved.
When we refresh, the entire saved text should display.

* How can we provide [multiple text] as input?

⊛ textarea Element

Syntax >> HTML

```
<textarea rows="8" cols="50">
</textarea>
```

→ rows attr - specify no: of lines
→ cols attr - specify no: of characters per line.

① <textarea rows="1" cols="5"></textarea>

② <textarea rows="3" cols="2"></textarea>

How to store data in local storage on Button click

① Adding Button with event listner

>> HTML

```
<textarea rows="8" cols="55"></textarea>
<br />
<button class="btn btn-primary mt-1" id="saveButton">Save</button>
```

>> JS

```
let saveButton = doc.getElementById("saveButton");
saveButton.onclick = function () {
    :
};
```

② Accessing textarea Element value

>> HTML

```
<textarea rows="8" cols="55" id="msg"></textarea>
```

>> JS

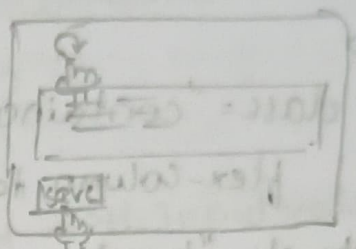
```
let saveButton = doc.get..("saveButton");
let textareaEl = doc.get..("msg")
saveButton.onclick = function() {
    let userEnteredText = textareaEl.value;
};
localStorage.setItem("userEnteredText", userEnteredText)
```

→ accessing

→ Save, goto console → application → Data visible → now refresh → still visible (in console) → web pg blank only.

→ Blank

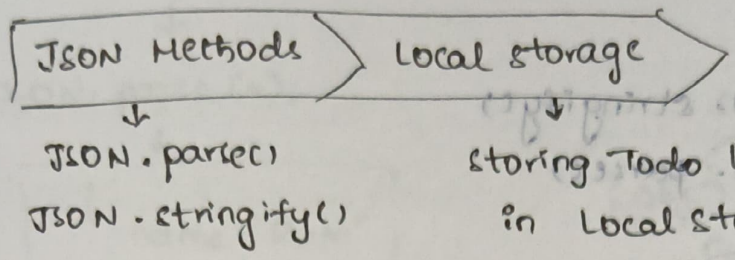
How to load Text msg automatically on Reload




```
let saveButton = doc.get_("saveButton")
let textareaEl =  ''      ("msg")
saveButton.onclick = function () {
    let userEnteredText = textareaEl.value;
    localstorage.setItem("userEnteredText", UserEntered
                                            Text);
}

let storedUserText = localstorage.getItem("userEnteredText")
if (storedUserText === null) {
    textareaEl.value = "";
}
else {
    textareaEl.value = storedUserText;
}
```

② Todos Application | Part-4



→ previously we learnt that in local storage "strings" as a values

how can we store other Types of val in local storage

⊛ Javascript object Notation (JSON)

JSON is data representation format used for

- storing data (client/server)
- exchanging data between client & servers

JSON supported Types
- number
- string
- Boolean
- Array
- object
- NULL

Javascript	JSON (identical)
10	10
"hello"	"hello"
true	true
[1,2,3]	[1,2,3]
null	null

JS object vs JSON object

```
{
  name: "Arun",
  age : 20,
  designation : "web Dev",
}
```

```
{
  "name" : "Arun",
  "age" : 20,
  "gender" : "Male"
}
```

↑ key double q (strings)

⊛ JSON Methods

- JavaScript provides JSON methods to convert Data into JSON format.
  - ↳ JSON.stringify()
  - ↳ JSON.parse()

* JSON.stringify()

→ can be array/obj etc

→ converts given value into JSON string.

syntax `JSON.stringify(val)` → val to be converted

Eg storing JS Objects → here val we want to store is obj - so we need to conv obj into string

```
let profile={
  name : "sai",
  age : 20,
  designation : "web Dev"
};
```

```
JSON.stringify(profile);
```

o/p `'{"name":"sai","age":20,"designation":"web Dev"}'`

↳ JSON string.

>>JS

```
let stringifiedProfile = JSON.stringify(profile);
console.log("");
console.log(typeof(""));   → string
```

* JSON.parse()

→ converts it parses JSON string & return as JS object

syntax `JSON.parse(string)` → string in JSON format

let x = '{"name" : "Ram", "age" : 20}'

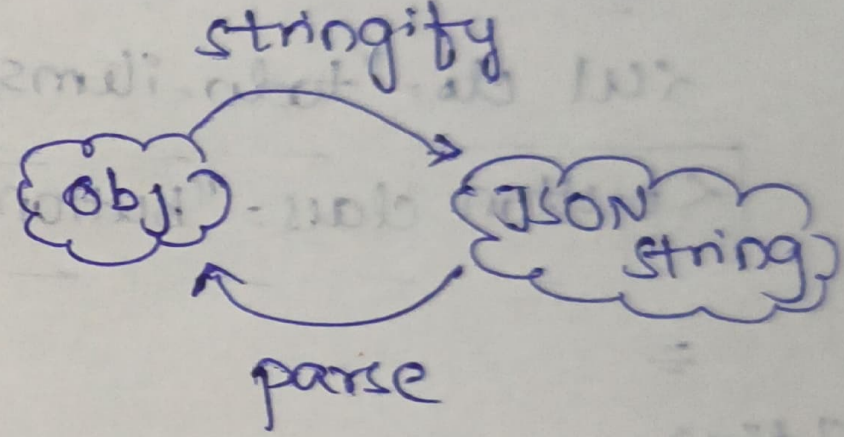
↓

JSON.parse(x);

↓

O/P

```
{
  name : "Ram"
  age : 20
}
```



» JS

```
let xy = JSON.parse(x);
console.log(xy); → object { name: "Ram", age: 20 }
    "   "  (typeof(xy)); → object
```

Not updated →

>2: object

How to remove corresponding Todo object from Todo list?

✱ Array Method: splice()

↳ push(): add element at end

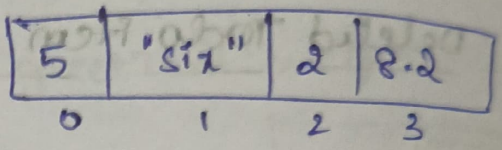
↳ pop(): remove end element

splice()

→ changes content of array.

→ using splice() we can:

- remove existing items
- replace existing items
- add new items.



Syntax

arr.splice(start, DeleteCount)

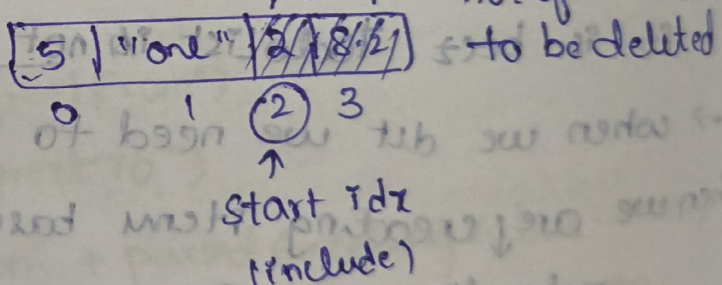
index

no: of items to be removed, strating from the given index.

Eg

```
let x = [5, "one", 2, 8.2]
x.splice(2,2)
console.log(x)
```

O/P [5, "one"]



✓ Removing existing items

→ splice() return array of deleted items.

```
>> let x = [5, "six", 2, 8.2]
   let dlt = x.splice(2,2);   ⟶ [2, 8.2]
   console.log(dlt)
```

dlt items.

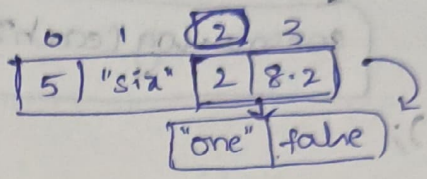
Adding new Items - splice()

syntax

```
arr.splice(start, dlt count, item1, item2 ...)
```

- start → idx (0)
- dlt count → no: of items to be removed, starting from given index.
- item1, item2 ... → items to be added, starting from given index.

eg
```
let x = [5, "six", 2, 8.2]
x.splice(2, 0, "one", false)
console.log(x)
```
o/p [5, "six", "one", false, 2, 8.2]



→ we can add items wherever we want at any idx
→ where as push() - only add end.

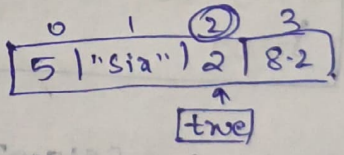
Replacing Existing items - splice()

syntax

```
arr.splice(start, dlt count, item1, item2 ...)
```

- dlt count → how many items we need to replace
- removing & add items

eg
```
let x = [5, "six", 2, 8.2];
x.splice(2, 1, true)
console.log(x)
```
o/p [5, "six", true, 8.2]



> x.splice(1, 2, "xyz", 100) → [5, "xyz", 100, 8.2]

* How to find index of item in array?

Array method : findIndex()

→ return index of first item that satisfies the provided testing function

→ if no item found returns '-1'

syntax

```
arr.findIndex(Testing function)
```

↳ func to execute on each val in array.

Some more Array Methods

3) includes() syntax `arr.includes(item)`

→ return true if provided item exist in array. If no item is found, it return false.

4) indexOf() syntax: `arr.indexOf(item)`

→ returns first index at which given item can be found in array. If no item is found, return -1.

5) lastIndexOf() syntax: `arr.lastIndexOf(item)`

→ returns last index at which given item can be found in array. If not found it return -1

6) find() syntax: `arr.find(Testing function)`

→ returns first items value that satisfies provided testing function. If no item is found, returns undefined

7) unshift() syntax: `arr.unshift(item1, item2 .... itemN)`

→ add one or more items to beginning of array & return the new array length.

8) shift() syntax: `arr.shift()`

→ remove first element from array & return that removed item

9) concat() syntax: `let newArr = arr1.concat(arr2);`

→ used to merge 2 arrays. It doesn't change existing arrays but instead return new array.

10) slice() syntax: `arr.slice(startIndex, endIndex)`

→ return portion b/w specified start & end index (end index not included) of an array into new array.

11) join() syntax `arr.join(separator)`

→ this method creates & return new string by concatenating all of items in array. separated by commas / specified separator string.

→ if array has only one item, then it will return without using specified separator.

→ here separator is a string used to separate each item of array. if omitted, the array items are separated with comma.

12) sort() syntax `arr.sort()`

→ sorts items of array & return sorted array.

→ Default sort order is asending.

removeItem()

→ this method removes specified storage object item based on key

syntax

```
localStorage.removeItem("key")
```

→ name of key to be removed

→ In this case, key is "todoList"

>>JS

```
localStorage.removeItem("todoList");
```

=

→ Entire todolist items will be removed.

* creating single Todo item
* creating multiple Todo items
* Taking user Input & creating Todos dynamically
* checking Todo
* Deleting Todo
* persisting Todo on Reload using local storage
* loops
    - for...of loop.
* local storage
    - setItem()
    - getItem()
    - removeItem()
* JSON methods
    - JSON.stringify(val)
    - JSON.parse(JSstring)

Module-7 Fetch & callbacks

① callbacks & schedulers

callback Function > schedulers

↓ Function as argument

↓ setInterval()
clearInterval()
setTimeOut()
clearTimeOut()

* passing function as Argument

» function displayGreeting (displayName) {
console.log ("Hello");
displayName(); ← execute funct which passed as arg.
console.log ("good morning");
}

↑ arg,

Hello
saru
Good morning

displayGreeting (function () {
console.log ("saru");
}); ← (method-1) (Directly pass funct as arg)

Define funct & provide funct name as arg

function displayRahul() {
console.log ("Rahul");
}
displayGreeting (displayRahul) ← method-2

displayName() is called then displayRahul() will be executed

Method 3 : funct expression with variable name

let displayRam = function() {
console.log ("Ram")
}
displayGreeting (displayRam);

⑧ Callback Function

↳ callback funct that is passed as argument to another function.

» function displayGreeting (displayName) {

≡

}

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

↳ callback function

⑨ schedulers

↳ used to schedule the execution of callback function

methods :-

1) setInterval()
2) clearInterval()
3) setTimeout()
4) cleartimeout()

---

① setInterval()

↳ allow to run a function repeatedly at specified interval of time.

Syntax `setInterval (function, delay)`

delay → Time in milliseconds

1 sec = 1,000 millisec

function → callback funct that is called repeatedly at specified interval of time

Eg.

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

op
0
1
2
3
...

- It executes repeatedly until:
  - Browser tab is closed
    (or)
  - scheduler is cancelled.

② clearInterval()

↳ cancel schedule previously set up by calling setInterval()

↳ to execute clearInterval(), we need to pass uniqueId returned by setInterval()

Syntax

```
let uniqueId = setInterval (function, delay);
```

variable that saves uniqId returned from setInterval()

```
>> clearInterval (uniqueId)
```

Eg: setInterval clearInterval

```
>> html
   <body>
     <button id=setBtn> set Interval </button>
     <button id=clearBtn> clear Interval </button>
   </body>

>> JS
# Accessing Btns.
let setBtnEl = doc.get--("setBtn")
let clearBtnEl = doc.get--("clearBtn")
# adding event listners.
let uniqId = null;
let BtnEl.onclick = function() {
    let co = 0
uniqId = setInterval (function() {
    console.log (co)
    co = co+1;
}, 2000);
```

```
clearBtnEl.onclick = function () {
    clearInterval(uniqId)
    console.log("clear Interval called");
}
```

O/P

setInterval clear

1 (1)
0
1
2 (2)
3

clear Interval called

③ setTimeout()

↳ executes function after specified time.

syntax

```
setTimeout(function, delay)
```

delay → time in m/s.

function → Callback function that is called after specified time

* setInterval executes repeatedly after specified time
* setTimeout " only once after specified time.

④ clearTimeout()

↳ we can cancel setTimeout() before it executes " callback function using clearTimeout().

↳ to execute clearTimeout() we need to pass uniqueId returned by setTimeout()

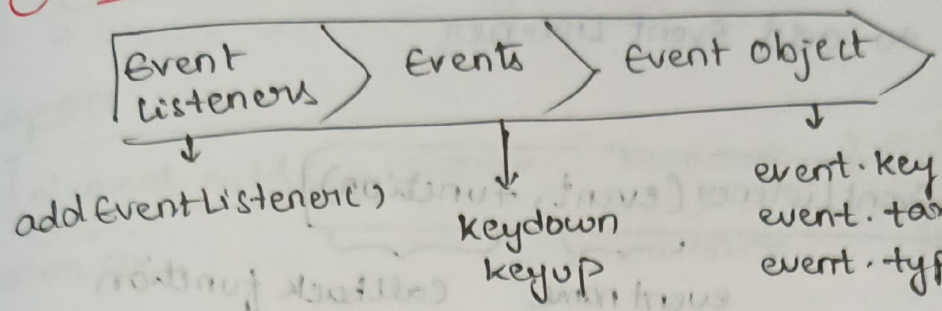
syntax

```
let uniqId = setTimeout(function, delay)
```

>>

```
clearTimeout(uniqId)
```

⑧ Event listeners & more Events



⑧ Event Listeners

↳ JS provide 3 ways to add event listeners to DOM element.

1) Inline event listeners
2) onevent Listeners
3) addEvent Listener()

① Inline event listeners

- onclicking the btn, we need to execute function.

>> HTML

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

E.L

>> JS

```
function greeting() {
    console.log("Hii Ram")
}
```

* we use HTML attr to add E.L

② onevent listeners

- we directly write E.L in Javascript
- to access btn element, we give "id" to Btn.

>> HTML

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

>> JS

```
let greetEl = doc.get...("greetBtn")
greetEl.onclick = function () {
    console.log = ("Hii ram")
}
```

addEventListener()

- modern way to add Event Listener

syntax

```
element.addEventListener(event, function)
```

HTML element — event name — callback function

eg:-

```
btnElement.addEventListener('click', function)
```

>> HTML

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

>> JS

```
let greetBtn = doc.get..("greetBtn")
greetBtn.addEventListener("click",
                          function() {
    console.log("morning!!")
}
```

(*) Events :-

↳ events are "actions" by which user or Browser interact with HTML elements

eg: click → interacting Btn element.

Event Types :-

1) Keyboard Events
2) mouse Events
3) Touch Events

---- etc

#1 Keyboard Events

↳ Keyboard events is the user interaction with keyboard

Keyboard Events:

↳ ① keydown

↳ ② keyup

keydown event

↳ occurs when key is pressed down

syntax

```
element.addEventListener('keydown', function)
```

HTML Ele — event name — callback function

Eg:

```
let inputEl = doc.createElement("input")
function printkeydown(){
    console.log("keypressed")
}
inputEl.addEventListner("keydown", printkeydown)
document.body.appendChild(inputEl);
```

since already funct define, we can directly access by funct name

O/P

saiju

keypressed
keypressed
keypressed
keypressed
keypressed

whenever key is pressed the function is executed.

How to know which key is pressed ?

Event object :-

↳ whenever event happens, browser creates an event object

↳ It consists of info abt event that has happened.

↳ It contain many properties & methods:

* type
* target
* key
* timestamp
* stoppropagation
- - etc

Properties & methods

↳ for every event, event-specific prop & methods will be present.

Eg: specific to that event.

Keydown event has "key" property whereas, onclick event doesn't have it.

Eg: # event.type

```
let ipEle = doc.createEl.. ("input")
function printkeydown(event) {
    console.log("key pressed");
  # console.log(event.type);
}
ipEle.addEventListener("keydown", printkeydown);
doc.body.appendChild(ipEle)
```

event obj is passed as arg to callback fnct.

← access prop & methods of event obj. o/p keydown

event.target

↳ provide HTML element on which event has occured.

↳ Here it occurred on "input" element.

console.log(event.target); → o/p <input></input>

event.key

↳ this is event-specific prop

↳ specific to key-down, key-up events.

console.log(event.key); → Hi

o/p shift ← 1st shift

H ← 1st press key

i ← next

—x—

Keyup event

↳ keyup event occurs when key is "released"

Syntax

```
element.addEventListener("keyup", function)
```

HTML Elt — Event name — Callback

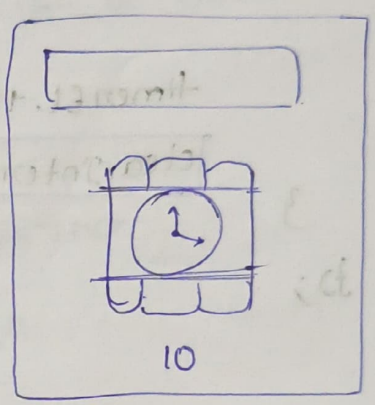
Summary

→ event listener
  * addEventListener()

→ Events
  * keydown
  * keyup

→ Event object
  * event.type
  * event.target
  * event.key

Before countdown reaches to "0" we must "defuse" then you'll display "You did it" otherwise it'll "Boom"



Example: Defuse Bomb Example

>> HTML

```
<body>
  <div cls = "timer-container">
    <input type="text" cls="user_input" id = "defuser" />
    <br />
    <img src="__" cls = "bomb-image">
    <p cls = "timer-display" id = "timer"> 10 </p>
  </div>
</body>
```

>> JS

```
let timerEle = doc.get...("timer")
let defuserEle = doc.get...("defuser")
let countdown = 10;

let intervalId = setInterval(function() {
    countdown = countdown - 1
    timerEle.textContent = countdown;
    if (countdown === 0) {
```

setTimeOut (2)

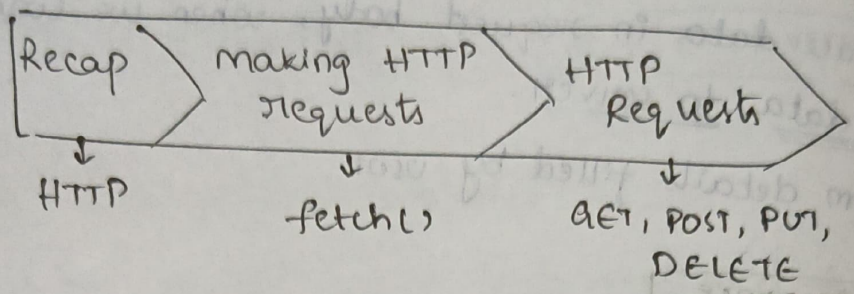
(1)

```
        timerEl.textContent = "Boom";
        clearInterval(intervalId);
    }
}, 1000);

defuserEl.addEventListener("keydown", function (event) {
    let bombdefuserText = defuserEl.value;
    if (event.key === "Enter" && bombdefuserText ===
                "defuse" && countdown !== 0) {
        timerEl.textContent = "You did it!";
        clearInterval(intervalId);
    }
});
```

Module-8 Fetch & callbacks 2

① HTTP Requests using JS



How to make HTTP req using JS?

⊛ fetch():-

↳ used to fetch resources across internet

Syntax

fetch(URL, OPTIONS)

URL of resource (loc)

request configuration

Request configuration

↳ we can configure fetch req with many opt like

- request method
- Headers
- Body
- credentials
- cache

Sample:

```
>>JS    let options = {
           method : "GET", -> # HTTP method.
           headers : {
              'content-Type' : 'application/json',
           },
           body : JSON.stringify (data),
           ....
       };
```

→ HTTP Method

method: "GET" → method property val can be GET, POST, PUT, DELETE.

key → By default - 'GET'

making HTTP GET req using fetch()

```
let options = {          → object
    method: "GET"
};
fetch("https://gorest.co.in/public-api/users", options);
                                                req config.
```

→ After running,

right click → more tools → Developer tools → (Top) Network → (left side) users → (right) headers (URL, req method, status code 200 ...etc are visible) → Preview (response from server)

Accessing HTTP response

Syntax

```
fetch(URL, OPTIONS)
  .then(function(response) {
      return response.status;
  })
  .then(function(status) {
      console.log(status)
  });
```

* http response obj is sent as arg to call back function
* from there we've to return response.

* HTTP response object

Properties & methods

↳ response object provide multiple properties to give more information abt HTTP response

* status
* statusText
* headers
* url
* text()
* json()
} methods.

→ status (number) – HTTP status code

→ statusText (string) – status text as reported by server
eg: "unauthorized"

→ headers – some more details

→ url

→ text() – Getting text from response

→ json() – Parses response as JSON

eg
```
let options = {
    method: "GET"
}
fetch ("https://gorest.co.in/public-api/users", options)
.then ( function (response) {
    return response.headers;
})
.then ( function (headers) {
    console.log (headers)
});
```
change this status / statusText / headers / url

* → response.text() → body of response conv to strings & we'll get ~~text~~ in JSON format.

```
.then (function (data) {
    console.log (data)
})
```

* to parse string ~~in~~ into JSON. ~~obj~~

```
.then (function (data) {
    let parseData = JSON.parse (data);
    console.log (parseData)
});
```

Accessing response in JSON

↳ we'll use .json()

```
.then (function (response) {
    return response.json()
.then (function (jsonData) {
    console.log (jsonData);
});
```

→ directly conv response body in json formate

* Till now, reading resource from server which is "GET"

—x—

→ let say, we want to create new resource, we'll use POST.

POST Request

syntax

```
let options = {
    method: "POST"
};
```

request configuration

↳ we need to mention "headers"

* content-type: says that, new data we're sending is in JSON format

"content-Type": "application/json",

* accept: client telling server that which ever respons you send, its better if you send in JSON format .. it specifies this

Accept: "application/json"

* Authorization: permissions will be specified.

authorization: "Bearer url"

↳ gorest.co.in → login → API Access TOKEN

specifies that you are req, you have permission

* we've to create resource where do we get?

creating new user (creating data)

>> JS

```
let data = {
    name: "Ram",
    gender: "male",
    email: "ram@gmail.com"
    status: "Active",
};
```

Adding that to HTTP req: (Body)

```
body: JSON.stringify(data)
```

→

```
let options = {
    method: "POST",
    headers: {
        "content-Type": "application/json",
        Accept: "application/json",
        Authentication: "Bearer ur"
    },
    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);
});
```

O/P Object { code: 201, meta: null, data: object }

code: 201
meta: null
data: object
id: createdat:
name: updatedat:
gender:
email:
status:

* If suppose, email already exist, we get error: 422 means, error on client side.

— x —

PUT request

↳ updata particular resource

```
>>  let options = {
        method: "PUT"
    };
```

* header, accept, authorization are same

updating user data

```
>>  let data = {
        name: "Ram charan"
    };
```

→ change

specify which user we want to change (Id)

```
fetch("https://gorest.co.in/public-api/users/1359", options)
```

* id displayed in console itself.

O/P > data: object
id: 1452
name: "Ram charan"

* if random id given — will get 404 error

DELETE Request

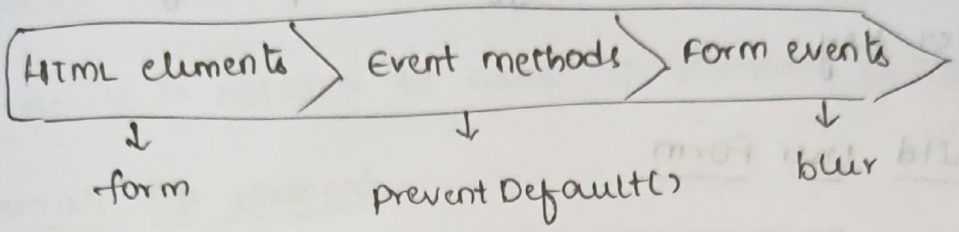
```
>>  let options = {
        method: "DELETE"
    };
```

* header, accept, authorization are same
* In URL, method resource we want to delete (id)

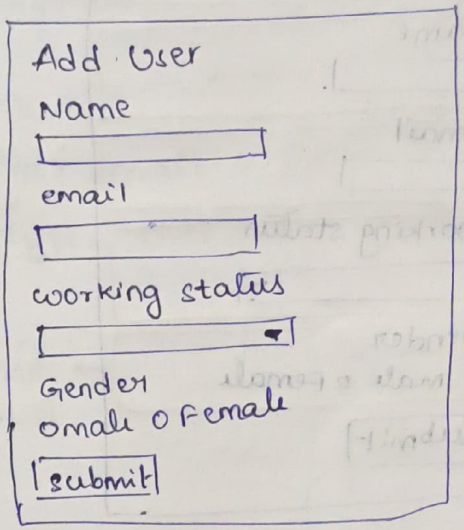
O/P 204 successfully deleted

Module-9 Forms

① Forms



* User Form



* HTML Forms

↳ used to collect data from user.

↳ different kinds:

* login/sign in form
* registration form
* contact us form ... etc

form element

↳ HTML Form element used to create html forms.

↳ It is container contains diff types of i/p elements like * Text fields
* checkboxes

Syn

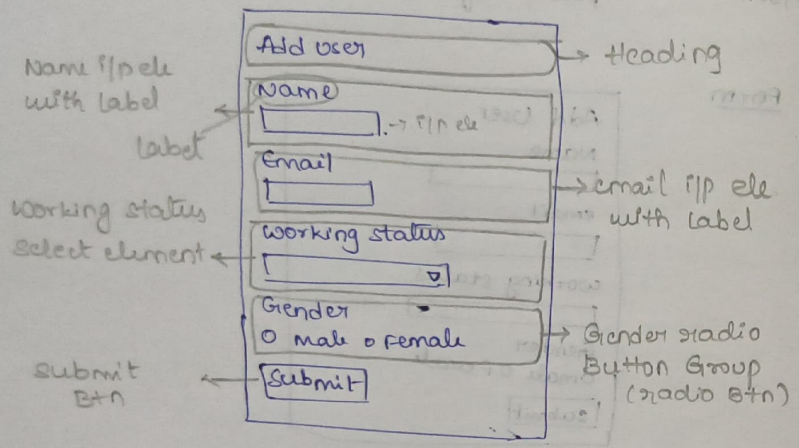
```
<form>
  ... form content
</form>
```

↳ form element has special event

* submit event

↳ submit event will be triggered on pressing Enter key.

Lets Build User Form



↳ form element will hold all the i/p elements inside it

>> HTML Adding form element

```
<body>
  <div cls="container">
    <h1 cls="form-heading">Add User</h1>
    <form id="myform">
    </form>
  </div>
</body>
```

* Name i/p element with label

>> HTML

```
<form id="myform">
  <div cls="mb-3">
    <label for="name">Name</label>
    <input type="text" cls="form-control" id="name" />
```

↳ Bootstrap cls

```
  </div>
</form>
```

↳ As it is related to name i/p element. So for = id

* email i/p element

>> HTML

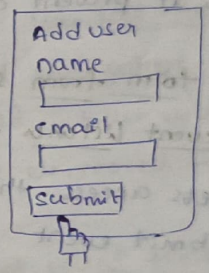
```
<form id="myform">
  <div cls="mb-3">
    <label for="email">Email</label>
    <input type="text" cls="form-control" id="email" />
  </div>
</form>
```

* Submit Button

>> HTML

```
<div cls="mb-3">
</div>
<div cls="mb-3">
</div>
<button type="submit" cls="btn btn-primary">submit</button>
```

event to trigger



↳ when we click on submit btn, o/p display as,

It may have been moved / delitid

Default Behaviour

↳ whenever we click Button or press Enter key while editing any i/p field of a form "submit event" will be triggered.

* preventDefault() method

Event obj

↳ whenever event happens, browsers create Event object

↳ contain info abt event.

↳ event obj have many prop & methods:

→ * preventDefault()
  * type
  * target
  * key
  * timestamp --- etc

↳ preventDefault() :- prevent occurance of default action normally.

↳ in form, it prevent default behaviour of submit event

Preventing form from submitting

Adding event listener to form

↳ In JS, lets access that form. for that form we will have submit event

>>JS

```
let myformEl = doc.get--("myForm");
myformEl.addEventListener("submit", function(event) {
    event.preventDefault()
});
```

* How to add field level validations

Name
[Ram]
Required *

email
[ ]
Required *

Event types

* keyboard events
* mouse events
* touch events
* form events --- etc

* Form Events

↳ an event that can occur within a form.

Form events:

- blur
- focus
- change --- etc

Blur Event

↳ event happens when element has lost focus

↳ say typing rahul & clicked somewhere else in form & moved to email field - blur event will occur Instead of focusing on one ele, moved to another. (lost focus)

showing required msg on not filled fields

* Adding Blur event listener

↳ access name i/p ele & add blur e.v.l

>>JS

```
let nameEl = doc.get--("name");
=
nameEl.addEventListener("blur", function(event) {
    console.log("blur event triggered");
});
```

* adding paragraph element

```
<div ...
  <label ...
  <input type="text" cls="form-control" id="name" />
  <p cls="errorMessage" id="nameError"></p>
```

Changing text content

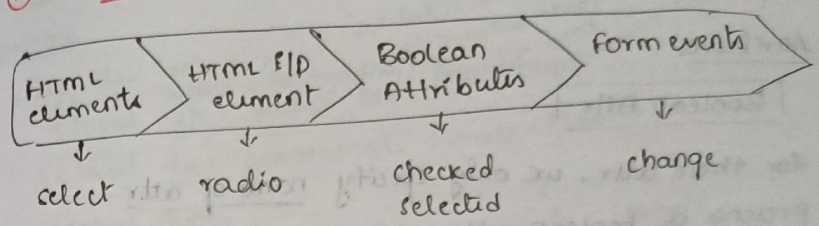
>> JS
```
let nameErrMsgEl = doc.get_("nameErrMsg")

nameEl.addEventListener("blur", function(event){
    if(event.target.value === ""){
        nameErrMsgEl.textContent = "Required *";
    }
    else{
        nameErrMsgEl.textContent = "*";
    }
});
```

for not filling email

>> JS
```
let emailEl = doc.get("email");
=
emailEl.addEventListener("blur", function(event){
    same as above
});
```

② Forms' part-2



* adding working status drop-down

↳ select element

↳ used to create drop-down list

Syn
```
<select>
    opt 1
    opt 2
    =
</select>
```

option element-

↳ each option is defined by option element

Syn `<option value='value1"> Label Text 1 </option>`

↳ text content (label Text 1) – used as label (values)

↳ every opt should contain value attribute

↳ It specifies value of given i/p element

* adding working status drop-down

>> HTML
```
<form id="myform">
=
# adding label
# adding options
    <div cls="mb-3">
        <label for="status"> working status </label>
        <select id="status" cls="form-control">
            <option value="Active"> Active </option>
            <option value='Inactive"> Inactive </option>
        </select>
    </div>
```

How to get particular option selected by default in drop-down?

Attributes

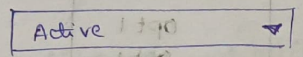
* **Boolean Attr**

↳ for these attr, we only specify name of attr

↳ Presence of boolean attr, represent true value, & absence of " represent false value

* selected Boolean Attr.
* checked
* disabled
* readonly
* default

The selected Attribute



↳ specifies that an opt should be pre-selected when page loads

* Selecting opt by default

>> HTML

```
<option value="Active" selected> Active </option>
<option value="Inactive"> Inactive </option>
```

* add radio buttons to user form

↳ opt are limited - radio
many opt - dropdown.

* **radio**

↳ radio btn is used to select one opt among all.

Syn `<input type="radio" value="value1" />`

>> HTML

```
<div cls="mb-3">
  <input type="radio" id="gendermale" value="male"/>
  <input " " id="genderfemale" value="female"
                                cls="ml-2"/>
</div>
```

o/p ○ ○

* adding labels to radio

>> HTML

```
<div cls="mb-3">
  <h1 cls="gender-field-heading"> Gender </h1>
  <input type="radio" id= - - - - -
  <label for="gendermale"> male </label>
  <input type= - - - - - -
  <label for="genderfemale"> Female </label>
</div>
```

(label a r/p ki related cheppadaniki for used.)

o/p ○ male ○ female.

↳ Here both radio btns will get selected if we click But only one should be selected.

To solve this,

How to specify only one opt should be selected

* **name attribute**

↳ this attr specifies name for HTML element (to group)

Syn `<input type="radio" value="male" name="gender" />`

↳ all radio btn with same name collectively called as radio group

↳ within a group, only 1 radio btn will be selected at a time.

>> HTML

```
<input type="radio" value="male" id="gendermale"
                                 name="gender" />
=
<input type="radio" value="female" id="genderfemale">
                    cls="ml-2"     name="gender" />
```

o/p ○ male ◉ female

how to get particular radio Button selected by default

(male) female

* checked boolean attr

↳ specifies that i/p element should be pre-selected (checked) when pg loads.
  * dropdown - selected B.A
  * radio btn - checked B.A

* checking radio btn by default

>> HTML

```
<input type="radio" value="male" id="genderMale"
       name="gender" checked />
   =
<input type="radio" value="female" id="genderFemale"
       cls="ml-2" name="gender" />
```

---

* Lets make **post request** on submitting form Data

-> post request on submitting

steps:

↳ maintain all form data in object
↳ send form data using post req method on submit event

① maintain data in obj

Initial form data object:-

```
>> JS.  let formdata = {
        name:"",
        email:" ",
        status: "Active",   -> val in HTML & status should be same
        gender: "male",     } default
     };
```

* form data should updat accordingly.

* How to updat working status ?

form events:

↳ event that occur within form.
events:-
  * blur ✓
  * submit
  * change ✓

change event

↳ It occurs when value of element has been changed
(val of dropdown will change - this event occur)

* updating working status

```
>> JS  let workingstatusEl = doc.get...("status");
        =
        let formData = {
            =
        };
        =
        workingstatusEl.addEventListener("change", function(event){
            formData.status = event.target.value;
        });
```

* updating gender status

```
>> let genderMaleEl = doc.get("genderMale")
   let genderFemaleEl = doc.get("genderFemale")
   =
   genderMaleEl.addEventListener("change",function(event){
       formData.status = event.target.value;
   })
   genderFemaleEl.add..Listene("change",function(event){
       formData.status = event.target.value;
   });
```

* when enter is pressed "change event" triggered & "submit event" will also be triggered.

* Similar to blur, when user press enter, or move to next field, form data will approp. updati with txt

blur → change

>>JS

```
nameEl.addEL("change", function(event){
    ≡
} else {
    —
}
formData.name = event.target.value
```

>>JS Similar for email El (both changes)

* focus change - blur
* on enter press - change.

② submit Data

↳ when user click enter/submit, submit event is triggered we have to make post req.

↳ Best practice - "Form level validations"

>> Submitting form data

JS

```
let myformEl = doc.get..("myForm")
≡
myformEl.addEL("submit", function(event){
    event.preventDefault();
    submitFormData(formData)        -> Form level validations
});
```

* post request

>>JS

```
≡
let formData = {
    ≡
};
function submitFormData(formData){
```

* req configuration

```
let options = {
    method: "POST"
    headers: {
        "Content-Type": "application/json",
        Accept: "application/json",
        Authorization: "Bearer 00f3f8----"
    };
    body: JSON.stringify(formData)
};
```

* url

```
let url = "https://gorest.co.in/public-api/users";
```

* Accessing http response

```
fetch(url, options)
.then(function(response){
    return response.json();
})
.then(function(jsonData){
    console.log(jsonData);
});
}
≡
```

O/P (successful)

```
object {code: 201, meta: null, data: object}
code: 201
meta: null
>data: object
    id: 1400
    name: —
    email —
    gender: —
    creati_at: —
    updatid_at: —
```

* with the help of ID, we can specifically access info abt particular user with ID, placing in URL

user specific url

>> https://gorest.co.in/public-api/users/1400

O/P

```
{"code":200,"meta":null,"data":{"id":1400,
"name":"rahul"
}
```

— x —

* what if user enter already existing email

> if existing mail enters, will get error as 4xx means client side error

→ "msg": "has already been taken"

checking response code

>>JS

```
fetch(url, options)
.then(function(response){
=
})
.then(function(jsonData){
if(jsonData.code===422){
```

checking msg in response

```
if(jsonData.data[0].message==="has already been taken"){
```

Display err msg

```
emailErrMsgEl.textContent = "Email Already Exists";
}
}
});
```