**Building a To Do App:**

The submit event is an event that is fired on a form element when it is submitted. It occurs when a user clicks the submit button, presses the Enter key while focused on a form field, or triggers the submit () method of the form element using JavaScript.

let form\_element = document.getElementById('ourForm');

form\_element.addEventListener('submit', function () {

  alert('Form Submitted');

});

The **default behavior of a web browser** on **submitting a form** is that **web browser will try to send the data off somewhere.**

**Event Object:**

evet parameter is the event object of the submit.

event object is a standard object that is automatically created by the browser when an event occurs, such as a click, keypress, or submit event.

The event object contains information about the event that occurred, including the type of event, the target element that triggered the event, the coordinates of the mouse pointer (if applicable), the key that was pressed (if applicable), and other properties depending on the type of event.

Properties of the submit event.

1. **type**: The type of event, which is always "submit" for submit events.
2. **target**: The element that triggered the event, which is the form element in this case.
3. **currentTarget**: The element that the event listener is attached to, which is also the form element.
4. **timeStamp**: The time at which the event occurred, in milliseconds since the Unix epoch.
5. **submitter**: The button or input element that was used to submit the form, if applicable.
6. **defaultPrevented**: A boolean value indicating whether the **preventDefault()** method was called on the event object

**Event Object:**

let form\_element = document.getElementById('ourForm');

// addEventListener method is going to pass our function

// event parameter that is the event object of the submit event

//event Object  contains information about the event  that occurred,

// including the type of event, the target element that triggered the event,

// the coordinates of the mouse pointer (if applicable),

//the key that was pressed (if applicable), and other properties

// depending on the type of event.

form\_element.addEventListener('submit', (event) => {});

**Properties of the submit event Object:**

**Event.type**

let form\_element = document.getElementById('ourForm');

// evet parameter is the event object of the submit

// The event object contains information about the event that occurred, including the type of event, the target element that triggered the event, the coordinates of the mouse pointer (if applicable), the key that was pressed (if applicable), and other properties depending on the type of event.

form\_element.addEventListener('submit', function (event) {

  alert(event.type);

});



**Event.target**

let form\_element = document.getElementById('ourForm');

// evet parameter is the event object of the submit

// The event object contains information about the event that occurred, including the type of event, the target element that triggered the event, the coordinates of the mouse pointer (if applicable), the key that was pressed (if applicable), and other properties depending on the type of event.

form\_element.addEventListener('submit', function (event) {

  alert(event.target);

});



**Event.timeStamp**

let form\_element = document.getElementById('ourForm');

// evet parameter is the event object of the submit

// The event object contains information about the event that occurred, including the type of event, the target element that triggered the event, the coordinates of the mouse pointer (if applicable), the key that was pressed (if applicable), and other properties depending on the type of event.

form\_element.addEventListener('submit', function (event) {

  alert(event.timeStamp);

});



let form\_element = document.getElementById('ourForm');

// evet parameter is the event object of the submit

// The event object contains information about the event that occurred, including the type of event, the target element that triggered the event, the coordinates of the mouse pointer (if applicable), the key that was pressed (if applicable), and other properties depending on the type of event.

form\_element.addEventListener('submit', function (event) {

  alert(event.defaultPrevented);

});



**On key Press Event**

let inputelement = document.getElementById('input\_id');

inputelement.addEventListener('keypress', function (event) {

  //get the key value using event.key and assign it to a variable called keyValue

  alert(event.key);

});

**Prevent Input Field of Form from being empty:**

To validate if an input field is blank or not, you can use the **required** attribute in the HTML input tag. This attribute specifies that the input field must be filled out before submitting the form

using the **required** attribute in combination with the **title** attribute.

<input type="text" id="name" name="name" required title="Please enter your name">

**Code**

let form\_element = document.getElementById('ourForm');

// evet parameter is the event object of the submit

// The event object contains information about the event that occurred, including the type of event, the target element that triggered the event, the coordinates of the mouse pointer (if applicable), the key that was pressed (if applicable), and other properties depending on the type of event.

form\_element.addEventListener('submit', (event) => {

  // prevent the form from submitting by default.

  event.preventDefault();

  // checkValidity is built-in method of HTML form elements in

  //JavaScript that is used to check if the form data is valid

  //when the user submits the form.

  // check all the form fields for any validation errors, such as required fields that are empty or invalid input formats. If all fields are valid, then form\_element.checkValidity method will return true, and if any field is invalid, form\_element.checkValidity will return false.

  console.log(form\_element.reportValidity());

  if (form\_element.reportValidity()) {

    alert('Form Submitted');

  } else {

    alert('Please fill out all required fields.');

  }

});

event.preventDefault();

Above code prevents default behavior of web browser (refresh/reload and send data off to somewhere)

**Accessing the value typed inside the input:**

**Give a unique id to the input:**

 <input type="text" autocomplete="off" id="input\_id"

        Required title="Enter Task"/>

**Selecting the input and storing the Object Returned by the getElement by id Method into a variable:**

// getElementById method is going to return an object that representes html input element ,saving the object returned by getElementById method into a variable

let input\_Field = document.getElementById('input\_id');

**Accessing the Value typed in the input Field:**

  // input\_Field is an Object

  // input\_Field object has a property called value value

  console.log(input\_Field.value);

**Creating an Object representing ul:**

// Creating an Object, representing ul element

let task\_list = document.getElementById('our\_list');

**Create a new function:**

function new\_item() {}

**Calling newly created function:**

  // Calling a newly created function and passing in the value

  //  typed in the input field as an argument into the function

  new\_item(document.getElementById('input\_id')

.value);

// receiving value typed in the input field as a parameter

function new\_item(input\_value) {}

**Inserting the value entered in the input:**

function new\_item(input\_value) {

  // selecting task\_list object

  // Looking inside the task\_list object and calling (by ()) method  called

  // insertAdjacentHTML

  task\_list.insertAdjacentHTML();

}

insertAdjacentHTML()

Takes in two arguments.

First argument is where in the element we want to add the new content(end of li).

So, write “beforeend”

Second argument is the content or html element we want to add to this element.

So

**First Create Template Literal and assign it to a variable:**

  // Create a template Literal and assing it to variable

  let our\_Task = ` <li id="tasks\_to\_do">Buy Carrots</li>

      <button type="submit">Delete</button>`;

**Instead of Buy Carrots add dynamic task which was input:**

function new\_item(document.getElementById('input\_id')

.value) {

  // Create a template Literal and assing it to variable

  let our\_Task = ` <li id="tasks\_to\_do">${ document.getElementById('input\_id')

.value }</li>

      <button type="submit">Delete</button>`;

  // selecting task\_list object

  // Looking inside the task\_list object and calling (by ()) method  called

  // insertAdjacentHTML

  task\_list.insertAdjacentHTML('beforeend', our\_Task);

}

Whereas **task\_list** is an Object that **represents** the **ul element**:

let task\_list = document.getElementById('our\_list');

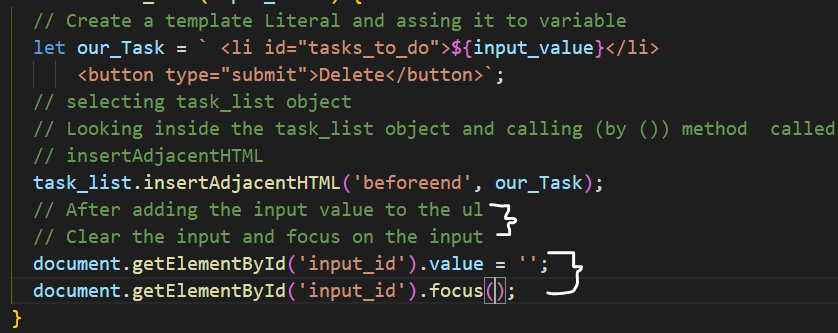
On pressing submit button, input must be focused,

  document.getElementById('input\_id').value = '';

  document.getElementById('input\_id').focus();

// new\_item function will be called everytime we hit the Task button

function new\_item(input\_value) {



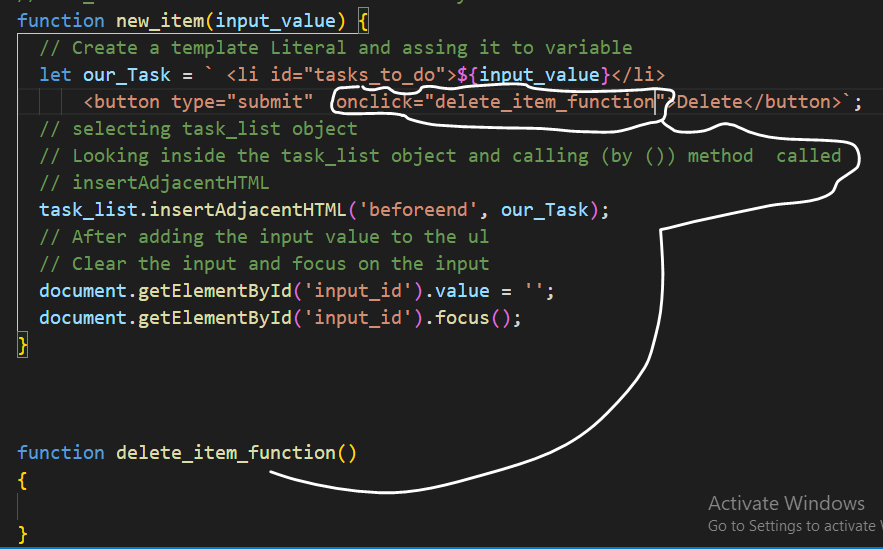
**Create a brand-new function:**

function delete\_item\_function()

{

}

**Call this function by using the template literal created on pressing submit button:**



When we click on delete button, we need to know which item we need to delete

**Browser Consider each HTML element as an Object:**

 <button type="submit" id="delete\_button" onclick="nameoffunction">Delete</button>

nameoffunction is a function that is being called by button object, hence this key word inside the nameoffunction will point towards the specific button object.

Hence, element that’s going to be deleted, it’s object is going to be passed inside the function.

  // on clicking Delete button delete\_item\_function function is called

  // Each Delete button has unique object

  //In <button type="submit"onclick="delete\_item\_function(this);">

  // button object is calling the delete\_item\_function function on clicking

 //this key word is going to point towards the button object

//this key word is going to be passes to our function

 let our\_Task = ` <li id="tasks\_to\_do">${input\_value}<button type="submit"

  onclick="delete\_item\_function(this);">Delete</button></li>`;

function delete\_item\_function(delete\_button\_object) {

  // delete\_button\_object is going to point towards the delete button that go clicked

  //delete\_button\_object.parentElement is going to select parentElement that is

  // containing child element button

  // remove() method is used to remove the element

  delete\_button\_object.parentElement.remove();

}