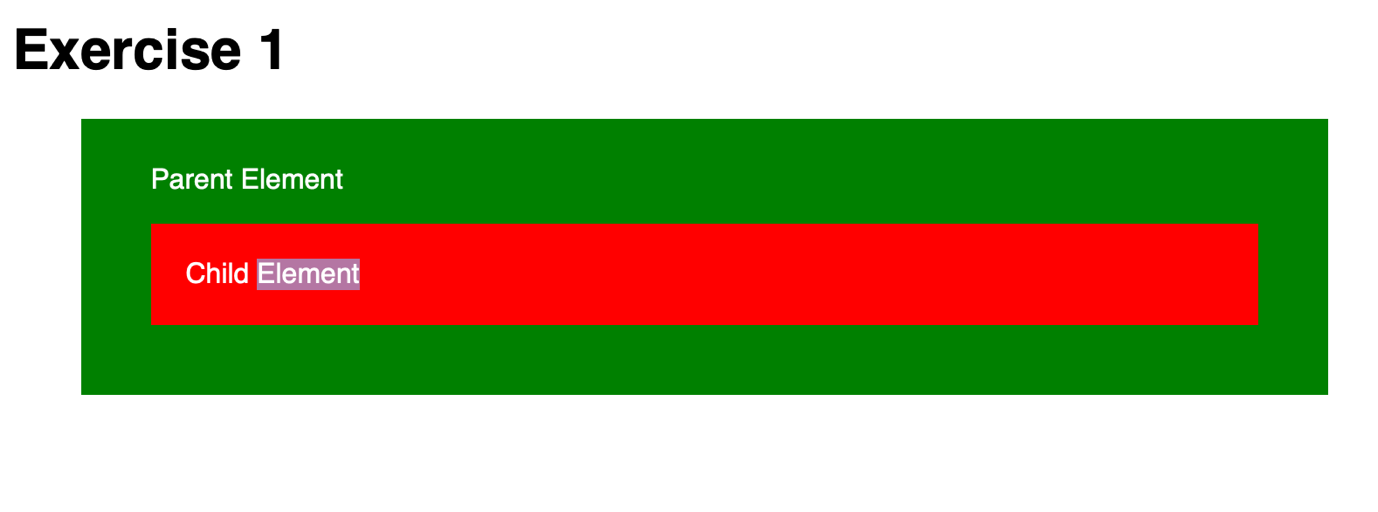
Lesson 4

JavaScript Events

Exercise 1



**Step 1:**

We will create an html file, Exercise1.html, with the following content

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title>Exercise 1</title>  <style type="text/css">  body {  font-family: sans-serif;  }  #parent {  margin: auto;  width: 80%;  padding: 10px 40px 40px 40px;  background-color: green;  color: white;  }  #child {  padding: 20px;  color: white;  background-color: red;  margin: 0;  }  </style>  </head>  <body>  <h1 id="header">Exercise 1</h1>    <div id="parent">    <p>Parent Element</p>    <div id="child">Child Element</div>  </div>    <script src="exercise1.js"></script>  </body>  </html> |

**Step 2:**

Now create a new file on the same folder called: exercise1.js

For this we would like to add three event Listeners for the following 3 elements:

* #child
* #parent
* And body

For each of this add a console log statement that consoles the name of each element

e.g console.log(“I am the child”);

|  |
| --- |
| /\* Syntax reminder:  myEl.addEventListener('click', function () {  // Code goes here...  });  \*/  // The elements to attach "click" handlers to  var docBody = document.body,  parentElement = document.getElementById('parent'),  childElement = document.getElementById('child');  // Add your listeners here... |

**Step 3:**

View the page and:

• click the *child* element...  
• ... then click the *parent* element...  
• ... then click the page heading: *Exercise 1*.

What do you see in the console?

**Step 4:**

Now amend your code so that you also console.log the event.target and event.currentTarget elements in each handler.

Test your page again. What do you see in the console?

**Step 5:**

Now amend the code so that the event does not bubble up from the *child* element (with stop-propagation).

Test your page again. Notice how clicking on the parent still causes a bubble?

**Exercise 1a:**

Add event handlers to #child and #parent that listen for the mouseleave event, which does not *bubble*

In each handler, console.log a message stating which handler it is, as you did in part A of the exercise.

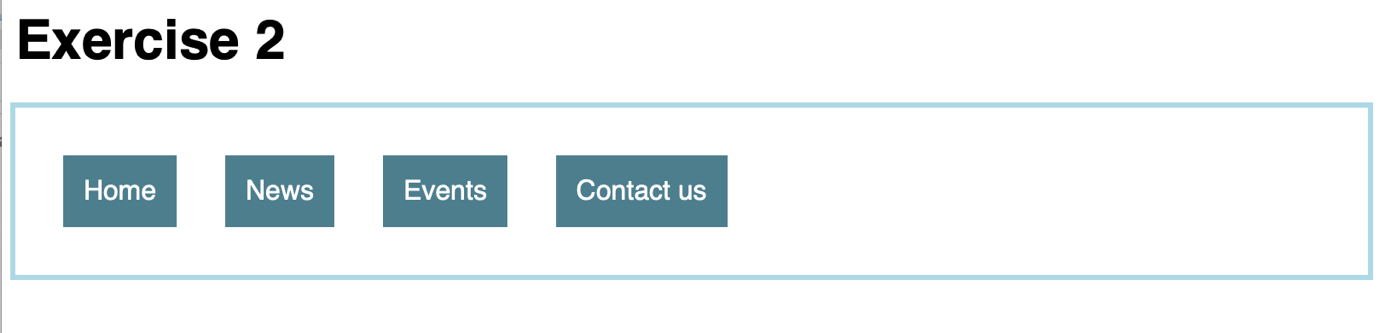
View the page and:

• place your mouse over the *child*...  
• ... then move your mouse out of *child* so it is over *parent*... • ... then move your mouse out of *parent*

What do you see in the console?

Exercise 2

At this exercise we have a menu with four links



Step 1

Create an html file called Exercise2.html, add the following code

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title>Exercise 2</title>  <style type="text/css">  body {  font-family: sans-serif;  }  .button {  margin: 0.75em;  padding: 0.75em;  display: inline-block;  background-color: #4D7E8E;  color: #fff;  cursor: pointer;  }  #button-wrapper {  padding: 1em;  outline: 3px solid #ADD8E6;  }  </style>  </head>  <body>  <div class="container">  <h1 id="header">Exercise 2</h1>  <div id="button-wrapper">  <p class="button" data-action="home">Home</p>  <p class="button" data-action="news">News</p>  <p class="button" data-action="events">Events</p>  <p class="button" data-action="contactus">Contact us</p>  </div>    </div>  <script src="exercise2.js"></script>  </body>  </html> |

Step 2:

Create a javascript file called exercise2.js, on this file you will need to add an event listener and write some code inside to find which element was selected.

|  |
| --- |
| // The element to attach the listener to  var buttonWrapper = document.getElementById('button-wrapper'); |

For this the currentTarget property wont help as it targets to the current event tartget.

Using event.target we can determine which button was clicked from inside our event handler and perform the necessary action.

We could potentially use this method

var attr = element.getAttribute(‘data-action’);

**Note**

You might have to do something more to avoid errors when the data-action element is not used.

Exercise 3



Step 1

Copy the following code to Exercise3.html

<!DOCTYPE html >

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>TODO</title>

<link rel="stylesheet" href="style.css" />

</head>

<body>

<main>

<h1>TODO App</h1>

<section class="input-form">

<h2>Add a new TODO</h2>

<form action="">

<textarea id="user-input"></textarea>

<button id="btn">Save text</button>

</form>

</section>

<section class="output">

<h2>TODO List</h2>

<div id="saved-text">

</div>

</section>

</main>

<script src="exercise3.js"></script>

</body>

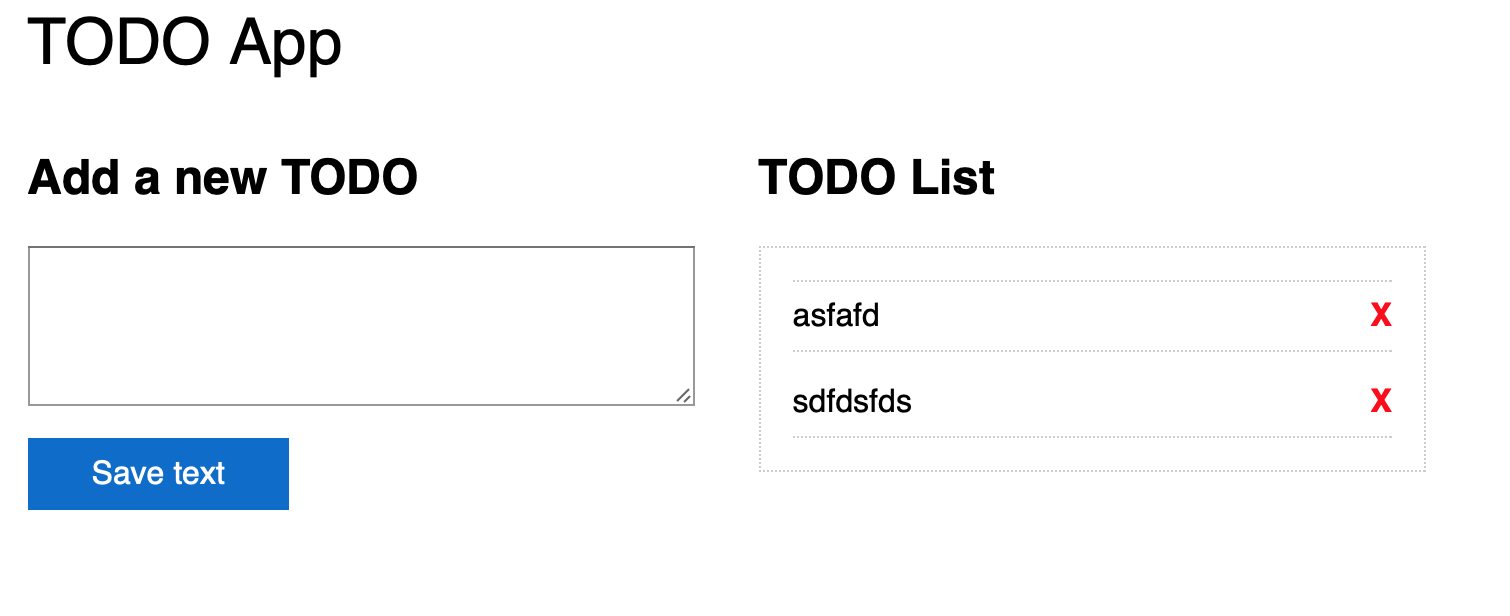
</html>

Step 2

Create an exercise3.js file

|  |
| --- |
| // Variables used by script.  // We need references to the textarea, the button and the div to place the results in  var userInput = document.getElementById('user-input'),  saveButton = document.getElementById('btn'),  resultEl = document.getElementById('saved-text');  // Helper function to help keep code clean...  // All it does is make a button element, which we then add to each of the p  // elements we add to the page  function makeMasterDeleteButton() {  var btn = document.createElement('a');  btn.textContent = 'X';  btn.classList.add('item-delete'); // give button the class "item-delete"  btn.setAttribute('href', '#');  return btn;  }  // Make sure we have valid elements to work with  if (userInput !== null && saveButton !== null && resultEl !== null) {  // Listen for clicks on form button  saveButton.addEventListener('click', function (event) {  var newNode, // Element we will create  itemDeleteButton, // Delete button for element  submitted = userInput.value; // Text from form field  // Check if user entered something  if (submitted !== '') {  // Make a p element and add the submitted text to it  newNode = document.createElement('p');  newNode.textContent = submitted;  // Make a delete button element and append it to p element  itemDeleteButton = makeMasterDeleteButton();  newNode.insertAdjacentElement('afterbegin', itemDeleteButton);  // add the p element to the page  resultEl.appendChild(newNode);  // Erase user's text from the textarea  userInput.value = '';  }  // Stop the browser's default behaviour. By default, when a "button" within a "form"  // element is clicked, the form gets submitted, resulting in a page reload (which we don't want)  event.preventDefault();  });  // Add event listener for item deletions here...  } |

The code can add a todo but what you will have to do is to write the code to delete.



**Your task:** Using event delegation, write an event listener that removes a paragraph from the “TODO list” when the corresponding delete button is clicked.

You will have to:

• Add the listener to a common ancestor of each button (i.e. the resultEl element)

• Inside the listener function:

* Make sure the thing that was clicked was actually a delete button. Note, the delete buttons all have the class: **item-delete**, so you can simply check the classList of the event.target object . E.g.: event.target.classList.contains('item-delete')
* If the thing that was clicked is a delete button, you should prevent the default behaviour (they are a elements)
* If the thing that was clicked is a delete button, you will need to delete its *parent* element (i.e. the *p* element). You can use a combination of parentNode and removeChild for this. To do this you will need a reference to the *p* element (event.target.parentNode) and to the div in which the *p* elements are contained (resultEl). E.g.:

resultEl.removeChild(event.target.parentNode)**;**

When done, you should be able to add items to the TODO list, and then remove them by clicking

the **X**. Clicking elsewhere on the TODO item or elsewhere in the list container will do nothing