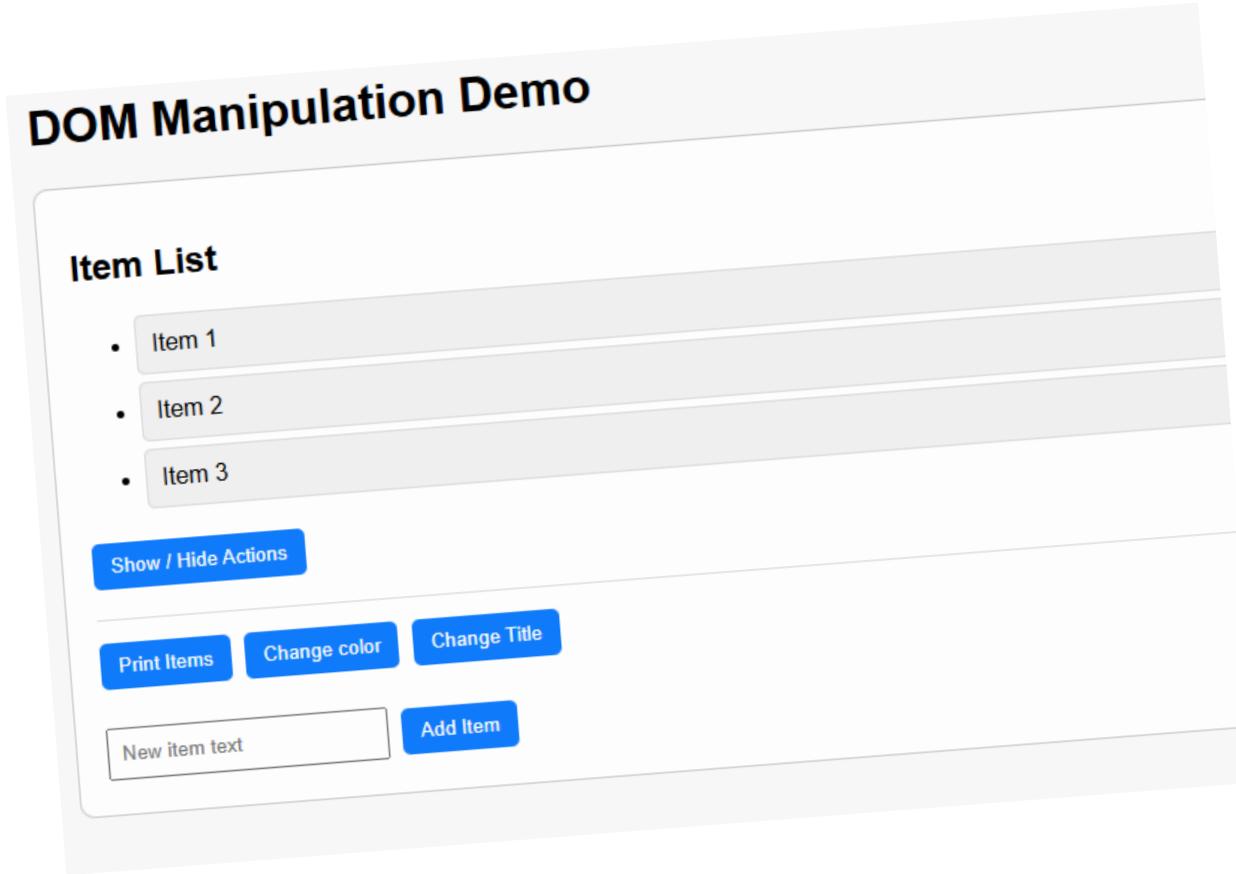


WEB DESIGN

W10 - DOM Manipulation





What will you learn today?



- ✓ Understand **what is the DOM**
- ✓ **Select DOM** elements
- ✓ Change DOM **element properties**
- ✓ Change DOM **element styles**
- ✓ Handle **DOM events**
- ✓ Get **DOM input value**
- ✓ Create a new DOM element

(as example using the ID of the element)

(as example the text content)

(as example the visibility, color)

(as example a click on button)

The power of JavaScript with HTML

JavaScript can be used on HTML pages to **update HTML elements dynamically**

```
<html>
  <body>
    <p id="text">Hello.</p>

    <script>
      const paragraph = document.getElementById("text");
      paragraph.textContent = "Hello from JavaScript!";
    </script>

  </body>
</html>
```

1 - Get the HTML element of id "text"

2 - Change its value dynamically

Hello from JavaScript!

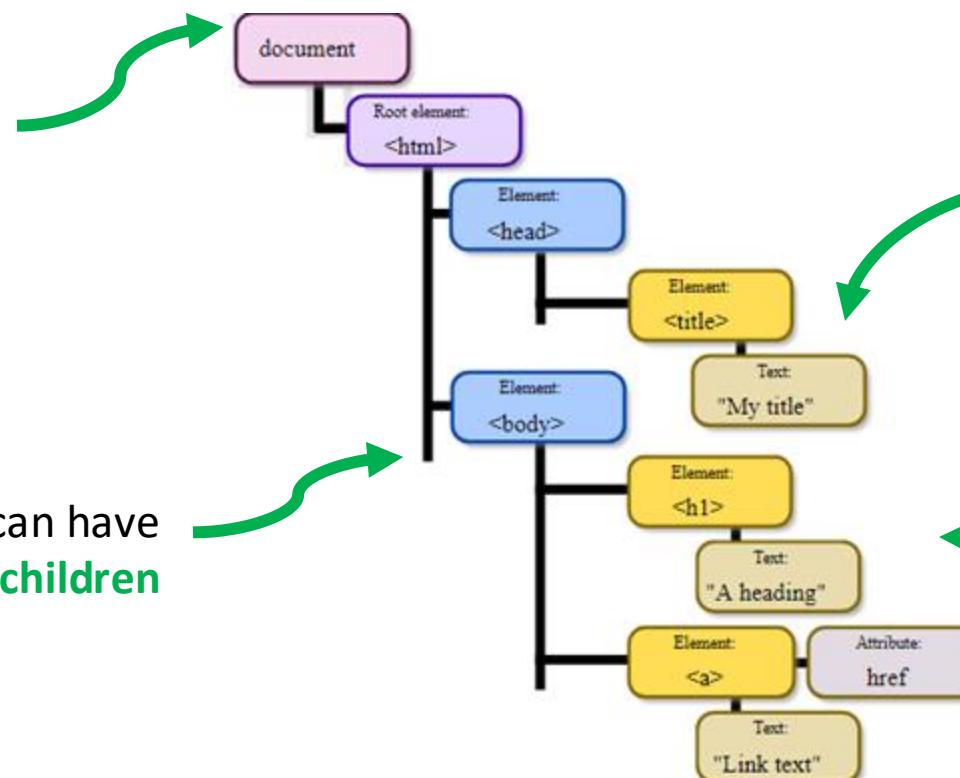
3 - UI is automatically updated

What is the DOM?

The DOM (Document Object Model) is a **tree structure** of an HTML page where each node is an **object** representing a part of this page

The **document** is the root **element** of the DOM tree

Elements can have **children**



Elements can have **properties**

The browser **creates the DOM** to render the page.



What can we do with the DOM ?

MORE INFOS

JS

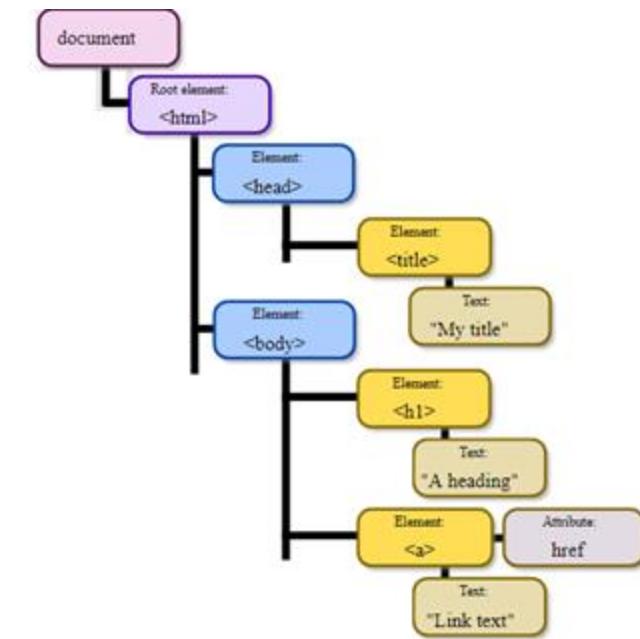
Using JS, you can manipulate **DOM elements**

DOM API

- Update elements attributes
- Update elements style
- Remove elements
- Add elements
- Listen to elements Events
- And much more...

DOM

- ! But **DOM** is not part of the JS
- ✓ Browser uses the **DOM API** to interact with DOM



DOM API - Access to elements

<code>var element = document.getElementById(ID)</code>	Get the element that matches with ID
<code>document.getElementsByClassName(CLASS)</code>	Get all elements that match with the CLASS
<code>document.querySelectorAll(QUERY)</code>	Get all elements that match the QUERY
<code>element.children[INDEX]</code>	Get the child of given INDEX of the current element
<code>element.parentNode</code>	Get the parent node of given element

DOM API - Change elements attributes

<code>element.textContent = "hello"</code>	Get / Set the text of an element
<code>element.className = "container"</code>	Get / Set the class of an element
<code>element.style.display = node</code>	hide an element
<code>element.style.display = block</code>	show an element
<code>element.style.backgroundColor = "red";</code>	Change an element style

DOM API - Add/remove elements

	Create a new element of the given type
<code>document.createElement(type)</code>	Type (string) : any valid tag name for HTML , SVG elements html, div, span, p, h1, h2, ul, ol, li, Table, form, input, textarea etc.
<code>element.appendChild(newElement)</code>	Add an element to the current element
<code>element.remove()</code>	Remove the current element
<code>parentElement.removeChild(childElement)</code>	Remove the given child element from the parent

An example to understand !

```
<div id="container">
  <p id="text1">This is a paragraph</p>
  <p id="text2">This is another paragraph</p>
</div>

<script>
const newP = document.createElement("p");
newP.id = 'text3';
newP.textContent = "New paragraph";

const domContainer = document.getElementById("container");
domContainer.appendChild(newP);
</script>
```

We create a new element of type paragraph

We set the node textContent

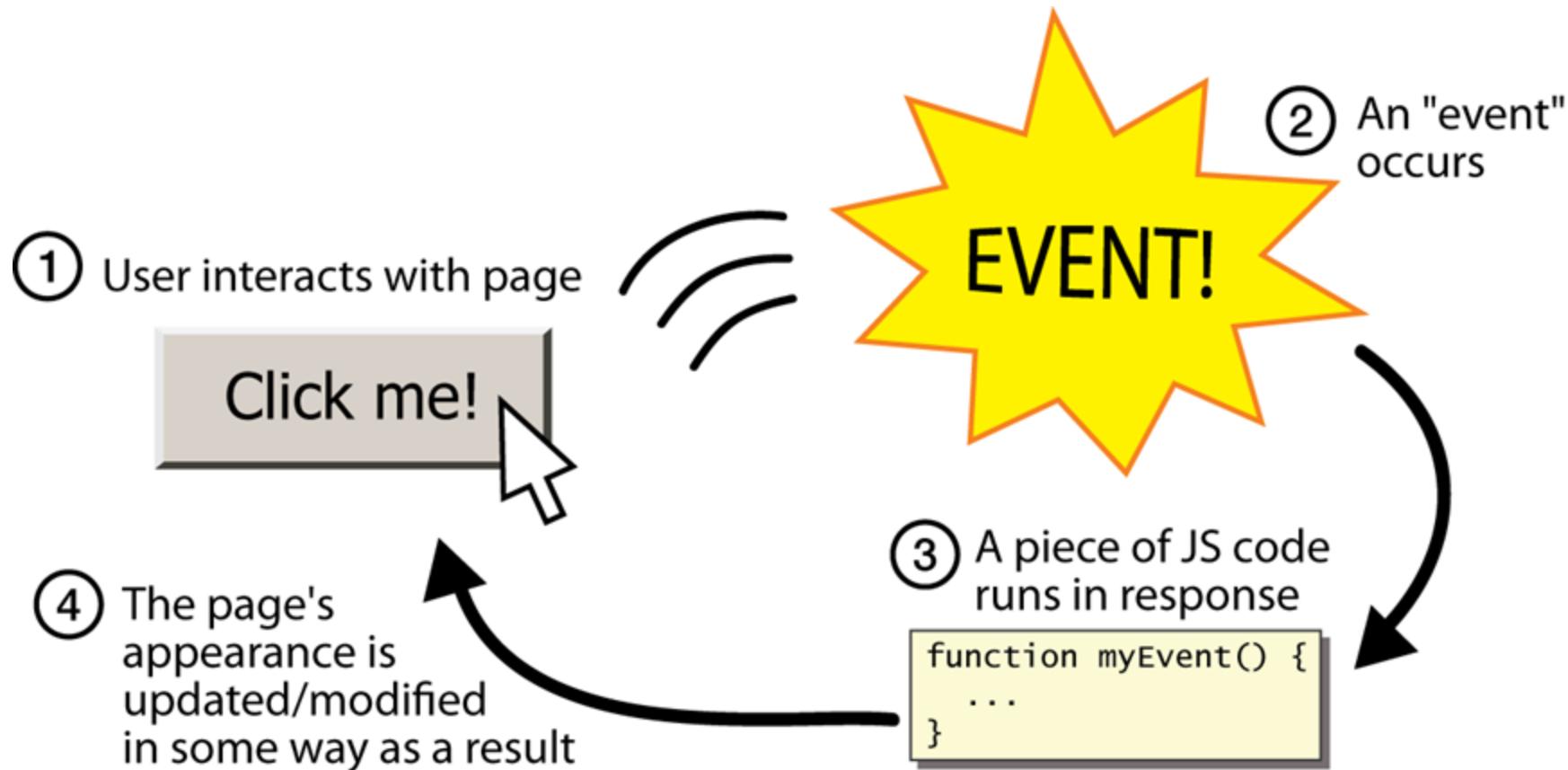
We get the DIV container , using getElementById

We append the new paragraph to the container DIV

```
<div id="container">
  <p id="text1">This is a paragraph</p>
  <p id="text2">This is another paragraph</p>
  <p id="text3">New paragraph</p>
</div>
```

As a result, a new paragraph is added to the DIV container

DOM API - Listen to an event



DOM API - Listen to an event

<code>element.addEventListener(name, function)</code>	Register an Event Handler , for the event name to the element , and the function to execute when the event happens.
<code>element.removeEventListener(name, function)</code>	Remove the Event Handler function for the event name from the element .

DOM API - Get Event information

event.target	a reference to the object related to the event
event.clientX	The X coordinate at which the event occurred
event.clientY	The Y coordinate at which the event occurred

Let's explore !

- ✓ Open the [DEMO-CODE](#)
- ✓ Take some time to **explore the code** for each **actions**

The screenshot shows a code editor with three tabs: HTML, CSS, and JavaScript. The HTML tab contains a simple UI structure with buttons for printing items, changing colors, and changing titles. The CSS tab is empty. The JavaScript tab contains event listeners for these buttons. A red arrow points from the 'Print Items' button in the browser screenshot to the line of code in the JavaScript panel that handles its click event.

```
HTML
16  <!-- ACTIONS PANEL -->
17  <div id="actions">
18  <button id="printItems">Print Items</button>
19  <button id="changeColor">Change color</button>
20  <button id="changeTitle">Change Title</button>
21
22  <br /><br />
23
24  <!-- ADD ITEM WITH INPUT -->
```

```
JavaScript
39  function onToggleActions() {
40    actionsDiv.style.display === "none" ? "block" : "none";
41  }
42
43  // ----- EVENT BINDING -----
44
45  document
46    .getElementById("printItems")
47    .addEventListener("click", onPrintItems);
48
49  document
50    .getElementById("changeColor")
51    .addEventListener("click", onChangeColor);
52
53  document
54    .getElementById("changeTitle")
55    .addEventListener("click", onChangeTitle);
56
57  document
58    .getElementById("addItem")
59    .addEventListener("click", onAddItem);
60
61  document
62    .getElementById("toggleActions")
63    .addEventListener("click", onToggleActions);
```

ACTIONS EXPLORE

- **Print** the list on console
- **Show / Hide** an element
- **Change the color** of an item
- **Change the title**
- **Add** an item

Case 1

When clicking the “Print Item” button, the list of items is displayed on console

The diagram illustrates a user interface for managing an item list. On the left, a light gray rounded rectangle contains a heading "Item List" and a bulleted list: "Item 1", "Item 2", and "Item 3". Below the list are five blue rectangular buttons with white text: "Print Item", "Change color", "Change Title", "Add Item", and "Remove Last Item". A red arrow points from the "Print Item" button towards a black rectangular box on the right. This box displays the text "Item 1", "Item 2", and "Item 3" in white, corresponding to the items listed in the interface.

Item List

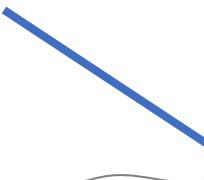
- Item 1
- Item 2
- Item 3

Print Item Change color Change Title Add Item Remove Last Item

Item 1
Item 2
Item 3

Display items lists upon button click

1 - select the button from the DOM, using the ID



```
const printButton = document.getElementById("printItems");
printButton.addEventListener("click", onPrintItems);

function onPrintItems() {
  const list = document.getElementById("itemList");
  const items = list.children;

  for (const item of items) {
    console.log(item.textContent);
  }
}
```

Display items lists upon button click

```
const printButton = document.getElementById("printItems");
printButton.addEventListener("click", onPrintItems);

function onPrintItems() {
  const list = document.getElementById("itemList");
  const items = list.children;

  for (const item of items) {
    console.log(item.textContent);
  }
}
```

2 - Tell the browser
“When this button is
clicked, run this
function:

Display items lists upon button click

```
const printButton = document.getElementById("printItems");
printButton.addEventListener("click", onPrintItems);

function onPrintItems() {
  const list = document.getElementById("itemList");
  const items = list.children;

  for (const item of items) {
    console.log(item.textContent);
  }
}
```

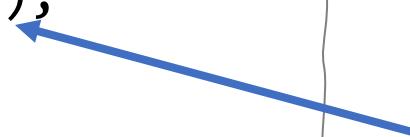
3 - onPrintItems runs
only when the **click**
happens

Display items lists upon button click

```
const printButton = document.getElementById("printItems");
printButton.addEventListener("click", onPrintItems);

function onPrintItems() {
  const list = document.getElementById("itemList");
  const items = list.children;

  for (const item of items) {
    console.log(item.textContent);
  }
}
```



4 - We access the `` element containing the items.

Display items lists upon button click

```
const printButton = document.getElementById("printItems");
printButton.addEventListener("click", onPrintItems);

function onPrintItems() {
  const list = document.getElementById("itemList");
  const items = list.children; ←
    for (const item of items) {
      console.log(item.textContent);
    }
}
```

5 - children gives us all
 elements inside
the list.

Display items lists upon button click

```
const printButton = document.getElementById("printItems");
printButton.addEventListener("click", onPrintItems);

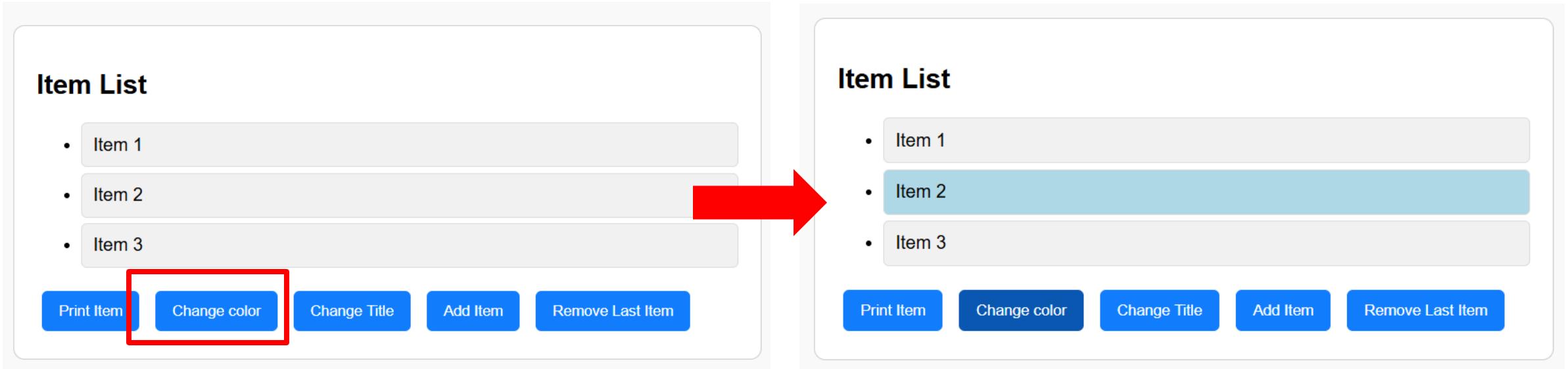
function onPrintItems() {
  const list = document.getElementById("itemList");
  const items = list.children;

  for (const item of items) {
    console.log(item.textContent);
  }
}
```

6 We print the LI
Text content

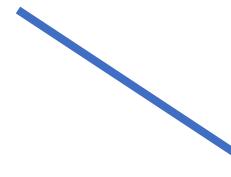
Case 2

When clicking the "Change color" button, Change the background color of the **second ** to light blue



Change a color upon click

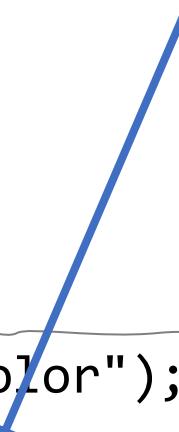
1 - select the button from the DOM, using the ID



```
const changeColorButton = document.getElementById("changeColor");
changeColorButton.addEventListener("click", onChangeColor),
```

```
function onChangeColor() {
  const list = document.getElementById("itemList");
  const secondItem = list.children[1];
  secondItem.style.backgroundColor = "lightblue";
}
```

2 - Tell the browser "When this button is clicked, run this function:



Select the list and get the 2nd element

- ✓ style is an object representing inline CSS.
- ✓ Changing a property immediately updates the UI.

Case 3

When clicking the "Change title" button, change the text of the <h2> to "**Updated Items List**".

The diagram illustrates a user interface transition between two states: 'Item List' and 'Updated Items List'.

Left Panel (Item List):

- Title:** Item List
- Content:** A list of items: Item 1, Item 2, Item 3.
- Buttons:** Print Item, Change color, Change Title (highlighted with a red border), Add Item, Remove Last Item.

Right Panel (Updated Items List):

- Title:** Updated Items List
- Content:** A list of items: Item 1, Item 2, Item 3.
- Buttons:** Print Item, Change color, Change Title, Add Item, Remove Last Item.

A large red arrow points from the 'Item List' panel to the 'Updated Items List' panel, indicating the flow or result of interacting with the 'Change Title' button.

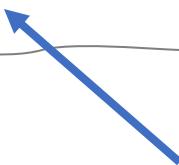
Change a text upon click

```
function onChangeTitle() {  
  const title = document.getElementById("containerTitle");  
  title.textContent = "Updated Items List";  
}
```

Get the title form the element ID in HTML

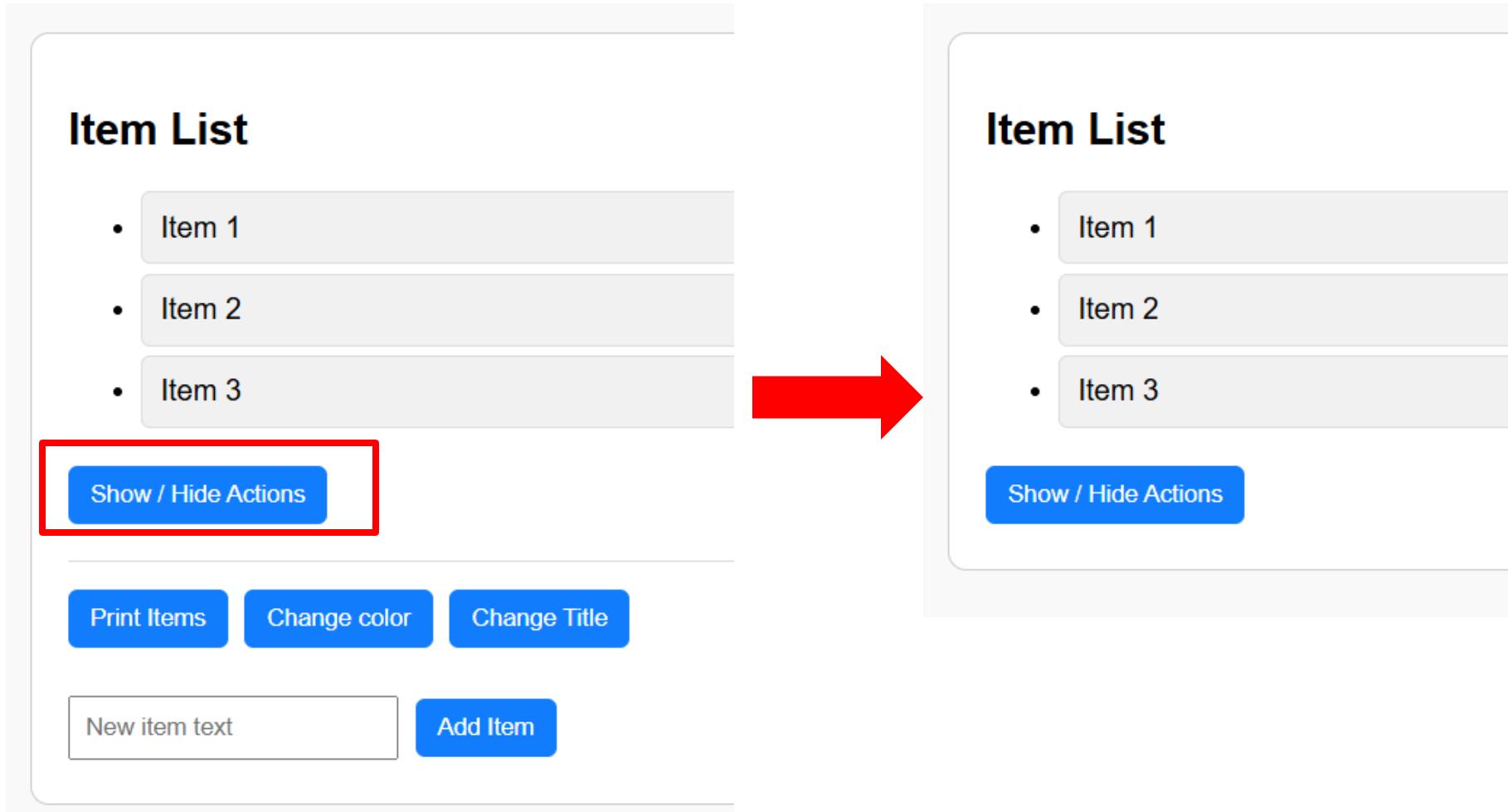


Change the element text



Case 4

When clicking the “Show/Hide” button, the DIV containing the actions button is displayed / hidden



Show / Hide

```
function show(element) {  
  element.style.display = "block";  
}  
  
function hide(element) {  
  element.style.display = "none";  
}  
  
function isVisible(element) {  
  return element.style.display !== "none";  
}
```

Let's first create 3 small reusable functions
To make the code more readable

```
function onToggleActions() {  
  const actionsDiv =  
    document.getElementById("actions");  
  
  if (isVisible(actionsDiv)) {  
    hide(actionsDiv);  
  } else {  
    show(actionsDiv);  
  }  
}
```

Now the action is just
To get the DOM element can make it visible / invisible

Case 5

When clicking the "Add Item" button, append a new element with the text entered in the input.

Item List

- Item 1
- Item 2
- Item 3

Show / Hide Actions

Print Items Change color Change Title

Add Item

Item List

- Item 1
- Item 2
- Item 3
- RONAN IS THE BEST

Show / Hide Actions

Print Items Change color Change Title

Add Item

Add a new item

```
function onAddItem() {  
    const input = document.getElementById("newItemText");  
    const text = input.value;  
    if (text === "") return;  
  
    const li = document.createElement("li");  
    li.className = "item";  
    li.textContent = text;  
  
    document.getElementById("itemList").appendChild(li);  
    input.value = "";  
}
```

Select the <input> element using its id.

Use .value to get the text the user typed

Create a new DOM element LI

Append the new item to the list

Clear the input field



WHAT WE HAVE LEARNT



- ✓ Understand **what is the DOM**

(as example using the ID of the element)

- ✓ Select **DOM** elements

- ✓ Change DOM **element properties**

(as example the text content)

- ✓ Change DOM **element styles**

(as example the visibility, color)

- ✓ Handle **DOM events**

(as example a click on button)

- ✓ Get **DOM input value**

- ✓ Create a new DOM element

AFTER THIS SESSION

1 – **Read** the following chapter on W3School:

https://www.w3schools.com/js/js_htmldom.asp

2 - You can also **watch the following videos**

<https://www.youtube.com/watch?v=NO5kUNxGlu0>

<https://www.youtube.com/watch?v=FQtjI1PC5Z0>

<https://www.youtube.com/watch?v=RKXIMnSwUcg>

<https://www.youtube.com/watch?v=WCRi7y6aNrQ>

https://www.youtube.com/watch?v=g_vXSKbfUiQ

