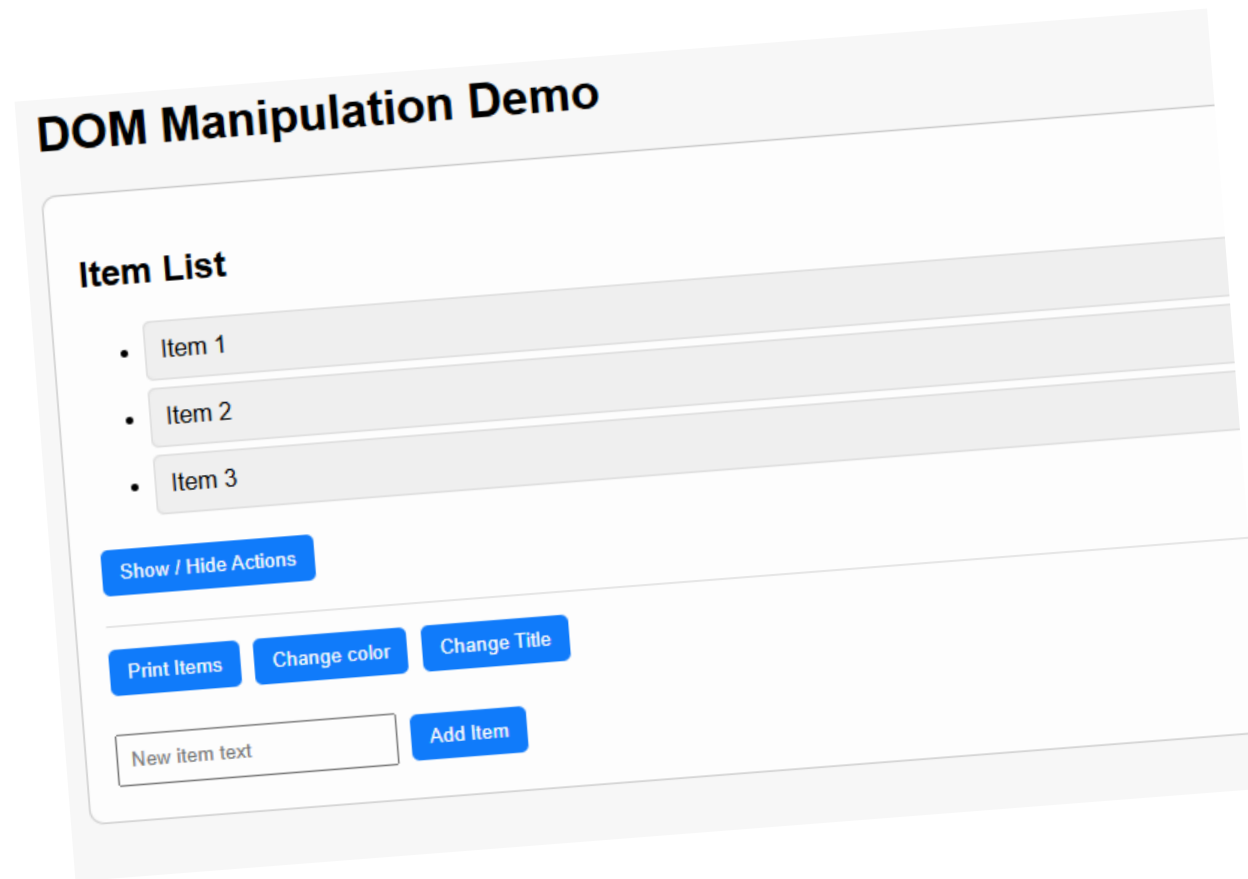


WEB DESIGN

W10 – DOM Manipulation





What will you learn today?



✓ Understand **what is the DOM**

✓ **Select DOM** elements

(as example using the ID of the element)

✓ 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

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>
```

1 - Get the HTML element of id "text"

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

2 - Change its value dynamically

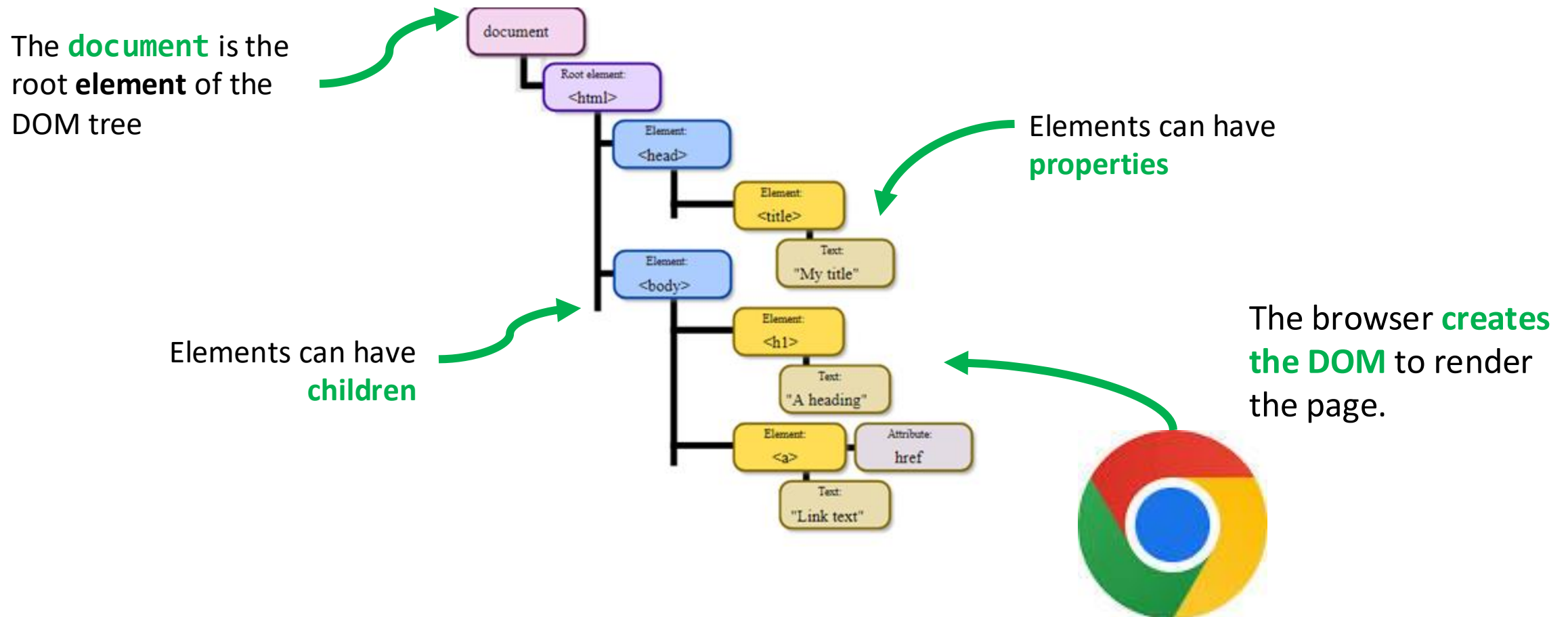
```
</body>
</html>
```

Hello from JavaScript!

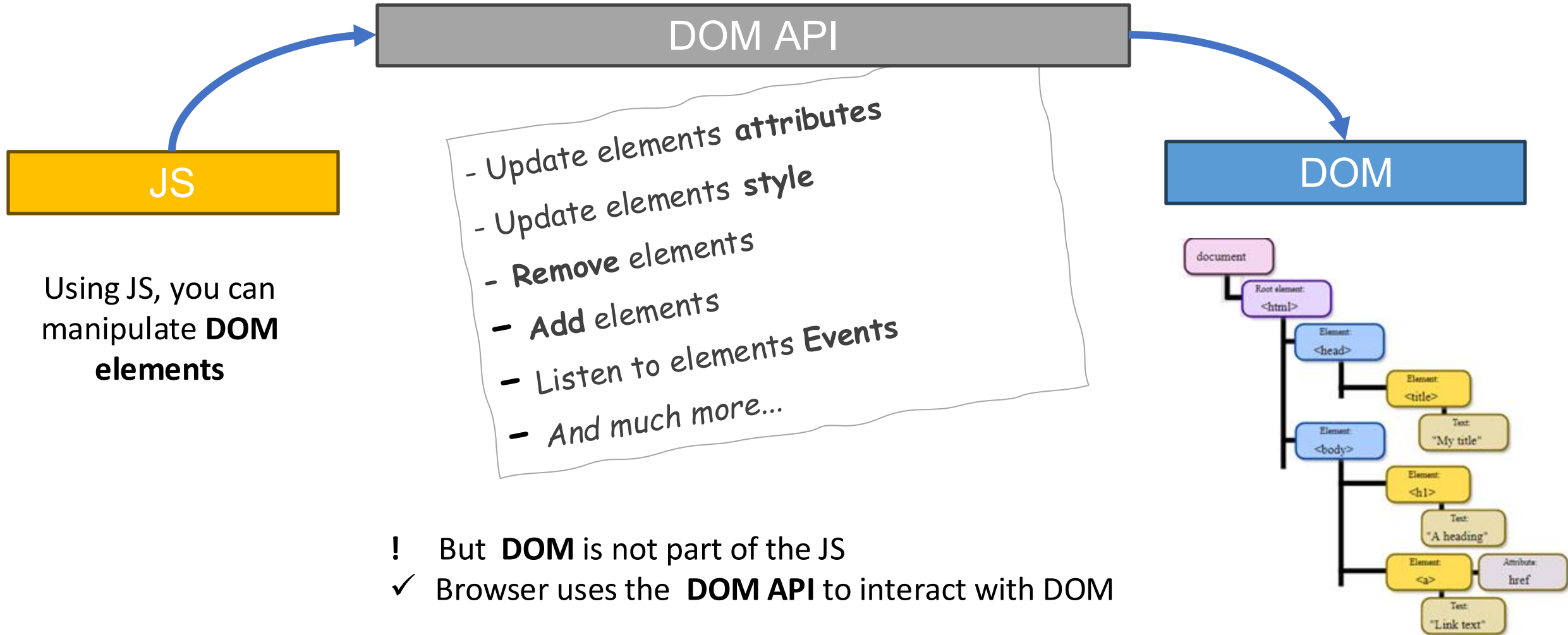
3 - UI is automatically updated

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



What can we do with the **DOM** ?

[MORE INFOS](#)

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 = none</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

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

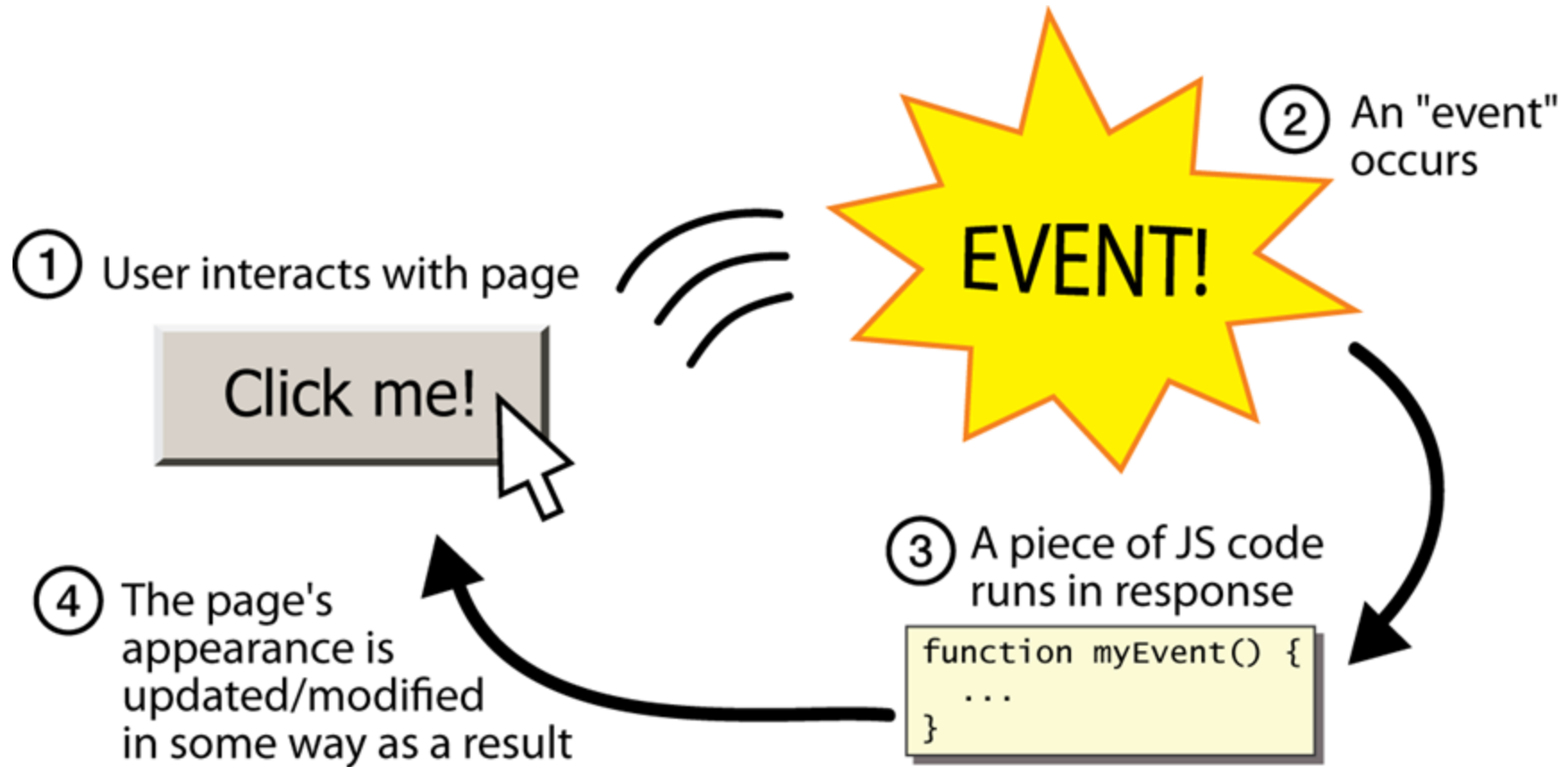
We get the DIV container , using getElementById

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

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

Register an **Event Handler**, for the event **name** to the **element**, and the **function** to execute when the event happens.

```
element.removeEventListener(name, function)
```

Remove the Event Handler **function** for the event **name** from the **element**.

DOM API - Get Event information

<code>event.target</code>	a reference to the object related to the event
<code>event.clientX</code>	The X coordinate at which the event occurred
<code>event.clientY</code>	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 web application titled "DOM Manipulation Demo". The application has a dark-themed code editor on the left and a light-themed UI on the right. The code editor has three tabs: HTML, CSS, and JavaScript. The HTML tab is active, showing the following code:

```
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 -->
```

The CSS tab is also active, showing the following code:

```
DOM Manipulation Demo
```

The JavaScript tab is active, showing the following code:

```
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);
```

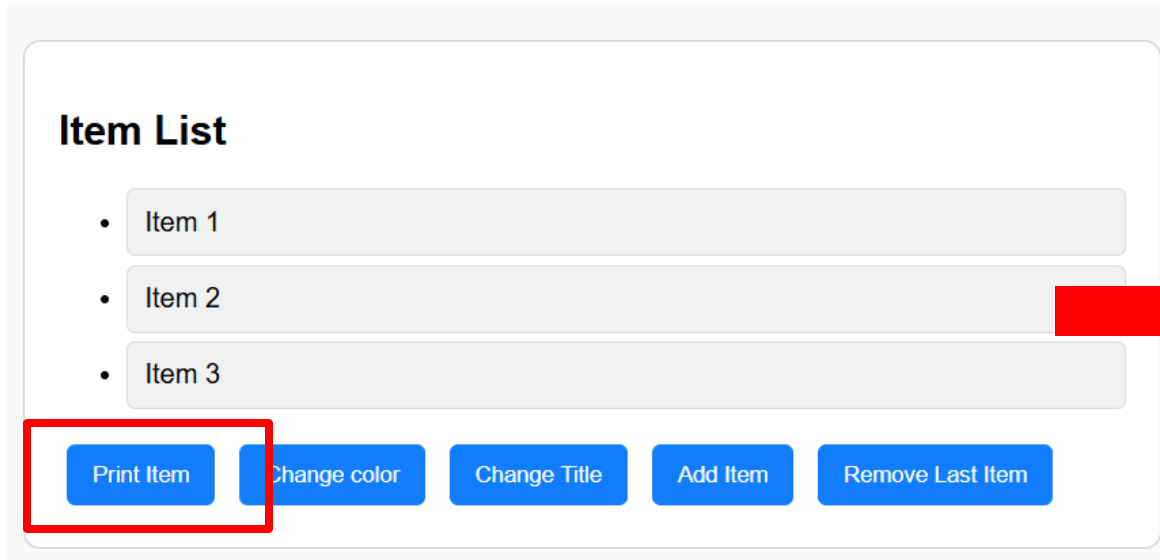
The UI on the right has a title "DOM Manipulation Demo". Below the title is a section titled "Item List" containing three items: "Item 1", "Item 2", and "Item 3". Below the list is a button labeled "Show / Hide Actions". Below that is a row of three buttons: "Print Items", "Change color", and "Change Title". Below these buttons is a form with a text input labeled "New item text" and a button labeled "Add Item".

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

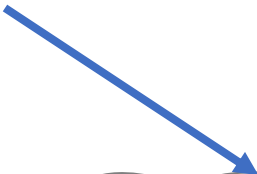
When clicking the “Print Item” button, the list of items is displayed on console



```
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);
  }
}
```

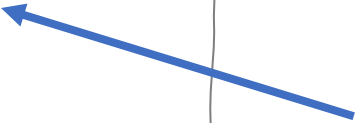
Display items lists upon button click

```
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:"

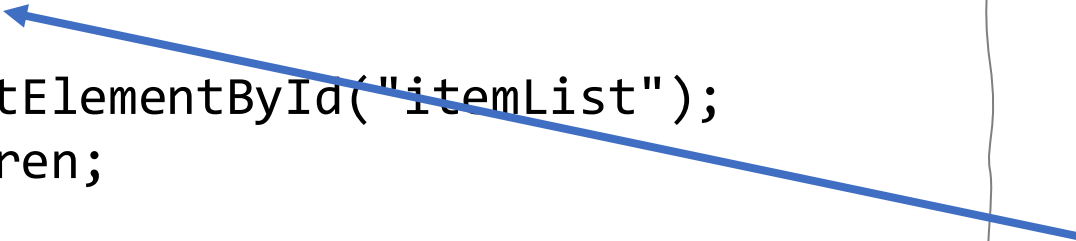


Display items lists upon button click

```
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.



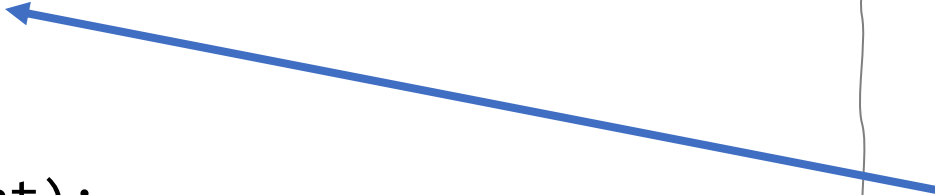
Display items lists upon button click

```
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.

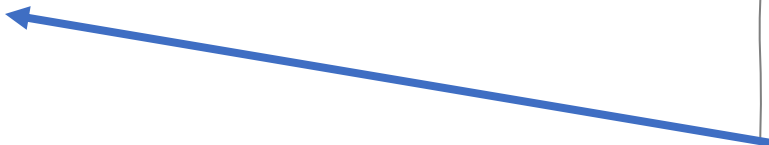


Display items lists upon button click

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

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

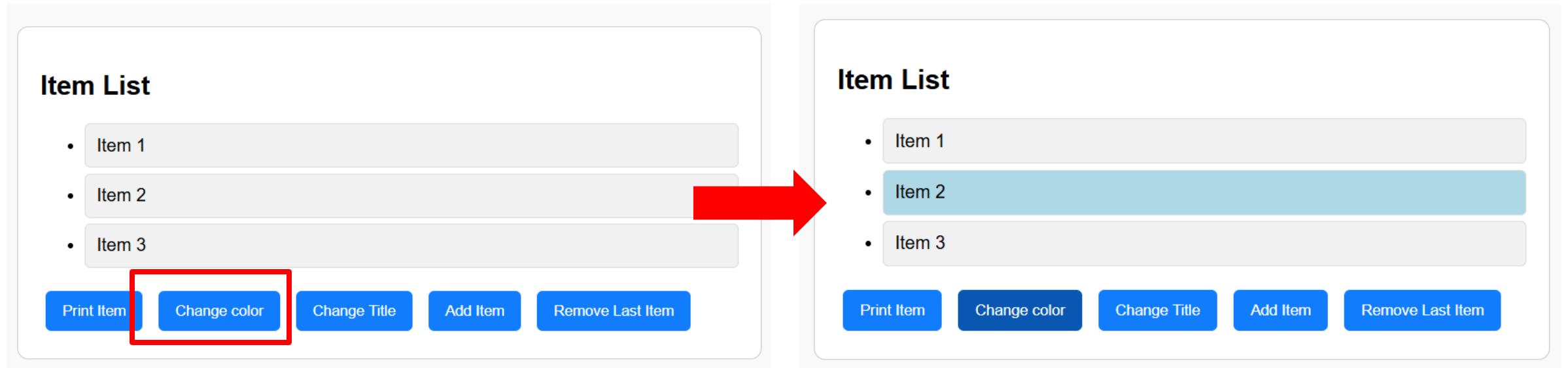
  for (const item of items) {
    console.log(item.textContent);
  }
}
```



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

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

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

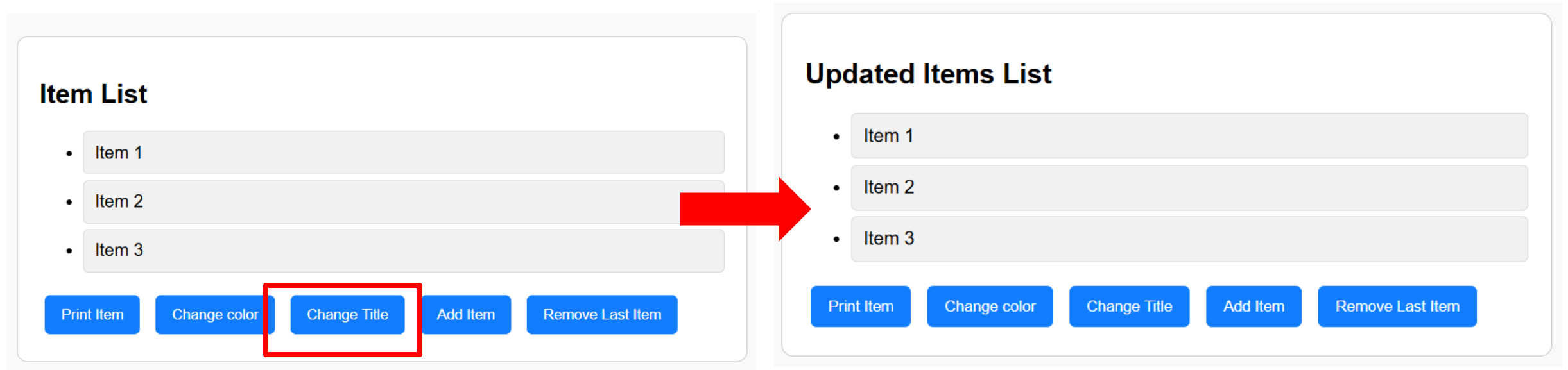
```
function onChangeColor() {  
  const list = document.getElementById("itemList");  
  const secondItem = list.children[1];  
  secondItem.style.backgroundColor = "lightblue";  
}
```

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**".



Change a text upon click

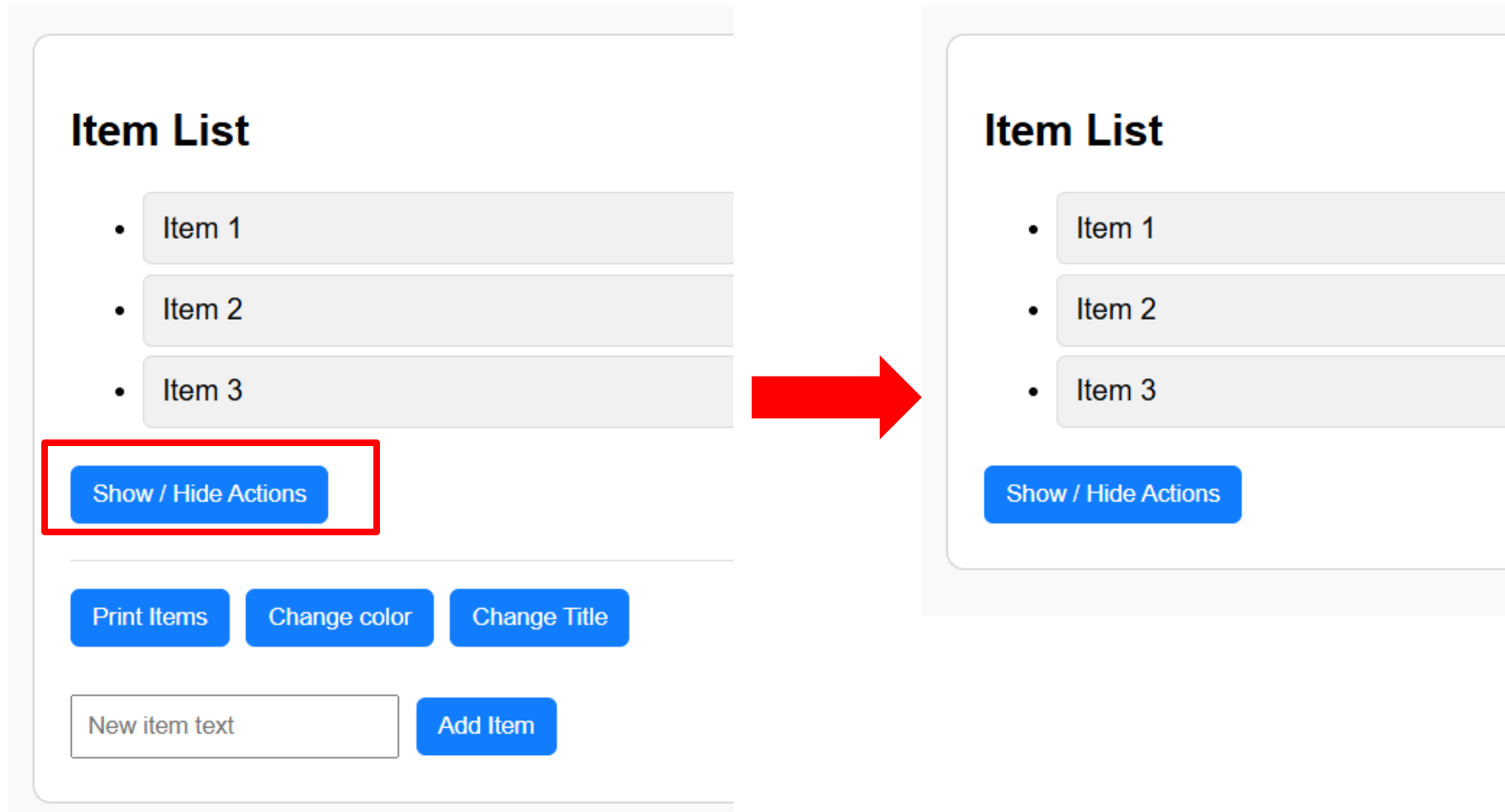
Get the title form the element ID in HTML

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

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

Select the <input> element using its id.

```
function onAddItem() {  
  const input = document.getElementById("newItemText");  
  const text = input.value;  
  if (text === "") return;
```

Use .value to get the text the user typed

```
  const li = document.createElement("li");  
  li.className = "item";  
  li.textContent = text;
```

Create a new DOM element LI

```
  document.getElementById("itemList").appendChild(li);  
  input.value = "";
```

Append the new item to the list

Clear the input field

```
}
```



WHAT WE HAVE LEARNT



✓ Understand **what is the DOM**

✓ **Select DOM** elements

(as example using the ID of the element)

✓ 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

