JS.2

▼ How to manipulate generic attributes on elements?

The element.getAttribute(key) method is used to get the value a certain attribute by its key.

The element.removeAttribute(key) is used to remove an attribute.

The element.setAttribute(key, value) is used to write a new attribute (or update the value of an old one that already exists).

The element.hasAttribute(key) method is used to check whether an attribute exists or not.

```
<div id="banner">
    <button disabled="disabled" id="login">Login</button>
</div>

const banner = document.querySelector("#banner");
banner.getAttribute("id"); // "banner"

const button = document.querySelector("#login");
button.removeAttribute("disabled");

banner.setAttribute("id", "navbar");

button.hasAttribute("disabled"); // false
```

▼ Should you update classes over styles?

Yes

▼ Why use .style property?

Sometimes you need to compute a dynamic value based on some variables

```
const banner = document.querySelector("#banner");
banner.style.backgroundColor = "red";

// hide element
banner.style.display = "none";

// show element by resetting it's display
banner.style.display = ""; //or "initial"
```

▼ How to remove an element from the DOM?

```
<h1 id="headline">Welcome</h1>

const headline = document.querySelector("#headline");
headline.remove();
```

▼ What does document.documentElement return?

the Element that is the root element of the document (for example, the <html> element for HTML documents).

```
const element = document.documentElement
```

▼ How to access the <body> element of the page?

```
document.body // the <body> of the page
```

▼ What is a data attribute?

the HTML spec recommends that developers prefix their own custom attributes with data-.

```
<-- This is recommended -->
<form id="payment-form" data-currency="EUR">
    ...
</form>
```

▼ What is a dataset object?

To read a data attribute, you can access the dataset object on an element.

```
const form = document.querySelector("#payment-form");
console.log(form.dataset); // {currency: "EUR"}
const currency = form.dataset.currency; // "EUR"

console.log(form.dataset)
/*
{
    userId: "2",
    currency: "EUR"
}
*/
```

▼ How to write data attributes on a dataset object?

```
const navbar = document.querySelector("#navbar");
navbar.dataset.userId = 43;
navbar.dataset.rememberMe = false;

<div id="navbar" data-user-id="43" data-remember-me="false"></div>
```

▼ How to get the parent element of the current element?

The element.parentElement property returns the parent element of the current element.

```
<div class="article">
    <h1>Hello World</h1>
    <p>Lorem ipsum</p>
</div>

const h1 = document.querySelector("h1");
console.log(h1.parentElement); // <div class="article">...</div>
```

▼ How to get the **closest** parent element of the current element?

The element.closest("CSS-selector") method returns the closest parent that matches the CSS-selector you specified. It searches for parent elements and goes up one by one.

```
<div class="main">
    <div class="banner">
        <h1>Hello World</h1>
    </div>
</div>

const h1 = document.querySelector("h1");
h1.closest(".main");
console.log(h1); // <div class="main">...</div>
```

▼ How to append HTML?

element.insertAdjacentHTML("beforeend", htmlString)

```
<div id="job-positions">
    <div class="position">2009-2013</div>
    <div class="position">2013-2015</div>
</div>

const positions = document.querySelector("#job-positions");
positions.insertAdjacentHTML("beforeend", `<div class="position">2015-2020</div>`);

// after
<div id="job-positions">
    <div class="position">2009-2013</div>
    <div class="position">2013-2015</div>
    <div class="position">2015-2020</div>
</div>
```

▼ How to prepend HTML?

element.insertAdjacentHTML("afterbegin", htmlString)

```
<div id="job-positions">
    <div class="position">2009-2013</div>
    <div class="position">2013-2015</div>
</div>

const positions = document.querySelector("#job-positions");
positions.insertAdjacentHTML("afterbegin", `<div class="position">2007-2009</div>`);

// after
<div id="job-positions">
    <div class="position">2007-2009</div>
    <div class="position">2009-2013</div>
    <div class="position">2013-2015</div>
</div>
```

▼ How to insert an array of items into the DOM?

The insertAdjacentHTML method presents the same security risk as innerHTML.

So, you should not use it if the variables you're interpolating might be coming from the user.

```
<ul id="apps-list"></ul>

const apps = ["Calculator", "Phone", "Messages"];
const list = document.querySelctor("#apps-list");

apps.forEach(app => {
    list.insertAdjacentHTML("beforeend", `<li>${app}</li>`);
});

// after
<ul id="apps-list">
    <li>Calculator</li>
    <li>Phone</li>
    <li>Messages</li>
</ul>
```

▼ What does the innerHTML method do?

write HTML and overwrite all the previous values

▼ What does the element.addEventListener(eventType, callback) method do?

allows you to wait for an event (let's say click for now) to happen on an element

Once that event occurs (the user clicks on the button), the callback function will execute.

▼ How to change a button to 'Loading' and disable it?

```
const button = document.querySelector("#app-button");

button.addEventListener("click", () => {
    button.setAttribute("disabled", "disabled");
    button.textContent = "Loading...";
});
```

▼ Web accessible click event?

make your website accessible for users with screen readers, you should only add a click event listener on <button> and <a> elements

An easier solution would be to wrap any item with a <button> element and then adjust the styles of that button to not make it look like a button.

```
<button class="clickable">
    <img src="..." alt="...">
</button>

.clickable {
    all: unset;
}
```

▼ Within an event listener, what argument does the callback receive from the browser?

event details

You can add the event parameter to be able to access the event details:

```
button.addEventListener("click", event => {
    // event callback
});
```

▼ What is event.currentTarget?

The event.currentTarget refers to the element to which the event listener has been attached.

```
button.addEventListener("click", event => {
    console.log(event.currentTarget); // same as the variable 'button'
    console.log(event.currentTarget.textContent); // text of button
});
```

▼ Why use debugger in an event listener?

you can add a debugger; statement which will pause the execution of your code and thus allowing you to see the event.currentTarget in the dev tools console:

```
button.addEventListener("click", event => {
    console.log(event);
    debugger;
});
```

▼ How to add an event listener on multiple elements?

```
const buttons = document.querySelectorAll("button");

const handleButtonClick = event => {
    console.log(event.currentTarget); // the 'button' that was clicked.
}

buttons.forEach(button => {
    button.addEventListener("click", handleButtonClick);
});
```

▼
▼
▼
▼
▼
▼
▼
▼
▼
▼
▼
▼
▼
▼
▼
▼
▼
▼
▼