

# DOM - JavaScript

```
JS script.js
const body = document.body
body.append("Hello World")
    ↪ append (method) ParentNo... ⓘ
    ↪ appendChild
```

Hello World

- **append & appendChild**

- appendChild needs Nodes (like Span,div,etc...)

```
body.append("abc" , "def") => allowed , result will be : abcdef
```

- while append can append anything from node to strings
    - appendChild can't accept more than one argument

- **createElement("div")**

- this will create an element , you can create any kind of elements with this
  - now if you want to add some string into this element you can just use either innerText OR textContent

- **querySelector("")**

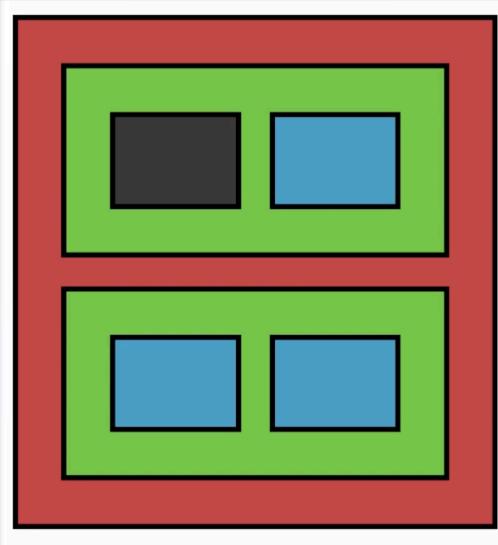
- Manipulates elements just like CSS , but only works on the first element , suppose you've two children with class = "child" , if you do,

```
document.querySelector(".child") => only access the first child
document.querySelectorAll(".child") => will access all the elements
with the child name class
```

## From parent to children :

```
JS script.js
const grandparent = document.querySelector(".grandparent")
const parents = Array.from(grandparent.children)
const parentOne = parents[0]
const children = parentOne.children
    ↪
changeColor(children[0])

>function changeColor(element) { ... }
```

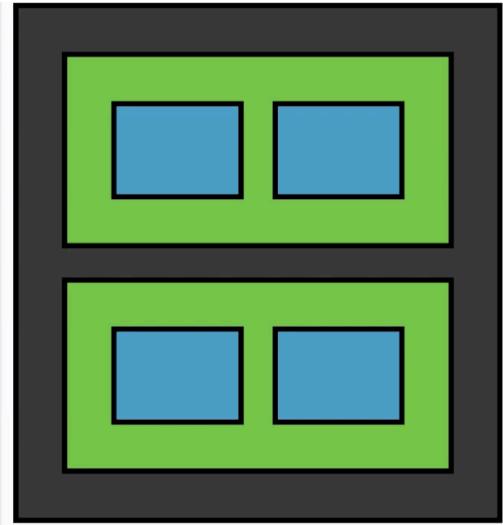


## From children to parent:

```
const childOne = document.querySelector("#child-one")
const parent = childOne.parentElement
const grandparent = parent.parentElement

changeColor(grandparent)

> function changeColor(element) { ... }
```

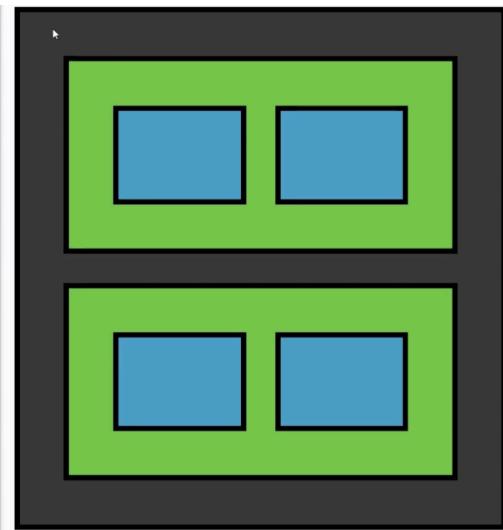


closest is used to access the immediate class

```
const childOne = document.querySelector("#child-one")
const grandparent = childOne.closest(".grandparent")

changeColor(grandparent)

> function changeColor(element) { ... }
```

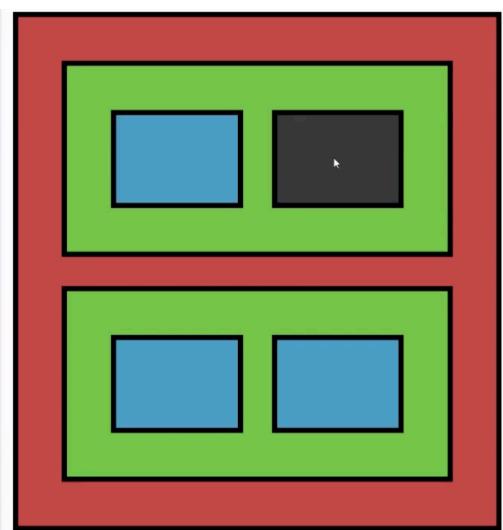


To access the sibling elements:

```
const childOne = document.querySelector("#child-one")
const childTwo = childOne.nextElementSibling

changeColor(childTwo)

> function changeColor(element) { ... }
```

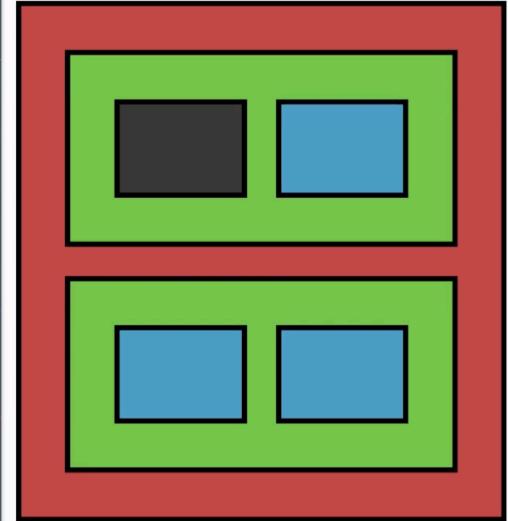


### Selecting the previous sibling:

```
const childOne = document.querySelector("#child-one")
const childTwo = childOne.nextElementSibling

changeColor(childTwo.previousElementSibling)

>function changeColor(element) { ... }
```



.remove() will totally remove the element from the page and by .append() we can have that removed element

```
const body = document.body
const div = document.querySelector("div")
const spanHi = document.querySelector("#hi")
const spanBye = document.querySelector("#bye")

.remove()

```

spanHi.getAttribute("id") is similar to => spanHi.id

spanHi.setAttribute("id","abcd") is similar to => spanHi.id = "abcd"

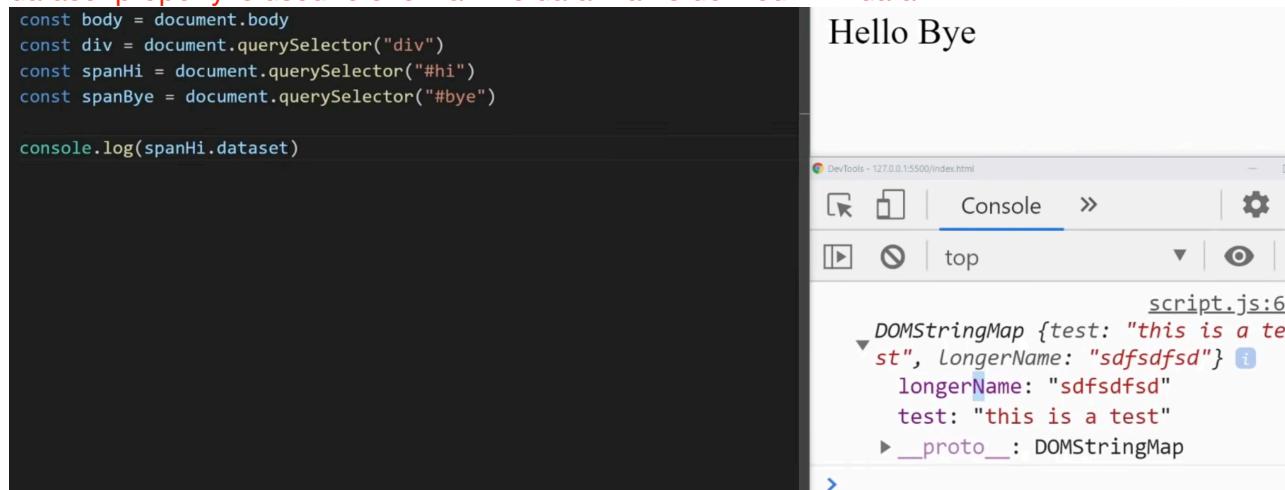
spanHi.removeAttribute("id") => removes the id attribute from the spanHi

### Data- ... :

in this way you can declare the names , hyphen(-) are converted into the capital letter.  
data-test => dataTest

```
<> index.html
<!DOCTYPE html>
<html lang="en">
<head>
  <title>DOM Manipulation</title>
  <script src="script.js" defer></script>
</head>
<body>
  <div>
    <span id="hi" data-test="this is a test"
          data-longer-name="sdfsdfsd">Hello</span>
    <span id="bye">Bye</span>
  </div>
</body>
</html>
```

dataset property is used to show all the data that is defined with data-



The screenshot shows a browser developer tools console window. On the left, the JavaScript code is displayed:

```
const body = document.body
const div = document.querySelector("div")
const spanHi = document.querySelector("#hi")
const spanBye = document.querySelector("#bye")

console.log(spanHi.dataset)
```

On the right, the browser window displays the page content: "Hello Bye". The developer tools console shows the output of the log statement:

```
script.js:6
DOMStringMap {test: "this is a te
st", LongerName: "sdfsdfsd"} ⓘ
longerName: "sdfsdfsd"
test: "this is a test"
▶ __proto__: DOMStringMap
```

classList.add, .remove, .toggle( , boolean)

```
spanHi.classList.add("new-class")
```