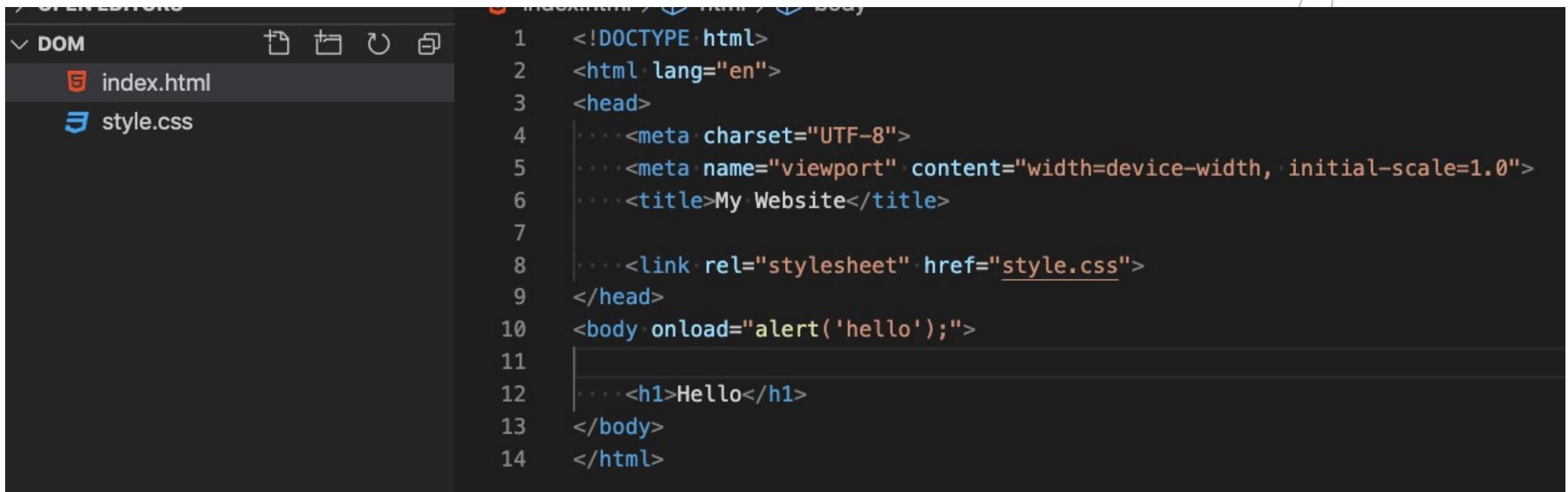


# *THE DOCUMENT OBJECT MODEL (DOM)*

# INLINE JAVASCRIPT

- The onload event occurs when an object has been loaded.



The screenshot shows a code editor interface with two panes. The left pane, titled 'OPENED FILES' and 'DOM', lists two files: 'index.html' (selected) and 'style.css'. The right pane displays the source code for 'index.html'.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Website</title>
    <link rel="stylesheet" href="style.css">
</head>
<body onload="alert('hello');">
    <h1>Hello</h1>
</body>
</html>
```

# *DISADVANTAGE*

- It is not very modular.
- It is difficult to debug.
- It is not a good practice.

Please try to avoid that if you can.

# *INTERNAL JAVASCRIPT*

- Integrate a script tag.

The screenshot shows a code editor interface with a dark theme. On the left, there is a sidebar titled 'DOM' which lists two files: 'index.html' (selected) and 'style.css'. The main area displays the source code for 'index.html' with line numbers from 1 to 18. The code includes HTML tags like <!DOCTYPE html>, <html lang="en">, <head>, <meta charset="UTF-8">, <meta name="viewport" content="width=device-width, initial-scale=1.0">, <title>My Website</title>, <link rel="stylesheet" href="style.css">, </head>, <body>, <h1>Hello</h1>, and a script tag containing 'alert("hello")'. The script tag is highlighted with a yellow background.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  ....<meta charset="UTF-8">
5  ....<meta name="viewport" content="width=device-width, initial-scale=1.0">
6  ....<title>My Website</title>
7
8  ....<link rel="stylesheet" href="style.css">
9  </head>
10 <body>
11
12 ....<h1>Hello</h1>
13
14 ....<script type="text/javascript">
15 ....|.... alert("hello");
16 ....</script>
17 </body>
18 </html>
```

![Screenshot of a browser developer tools DOM panel showing index.js selected. The code editor shows the following HTML structure: <!DOCTYPE html>, <html lang=](index.js)

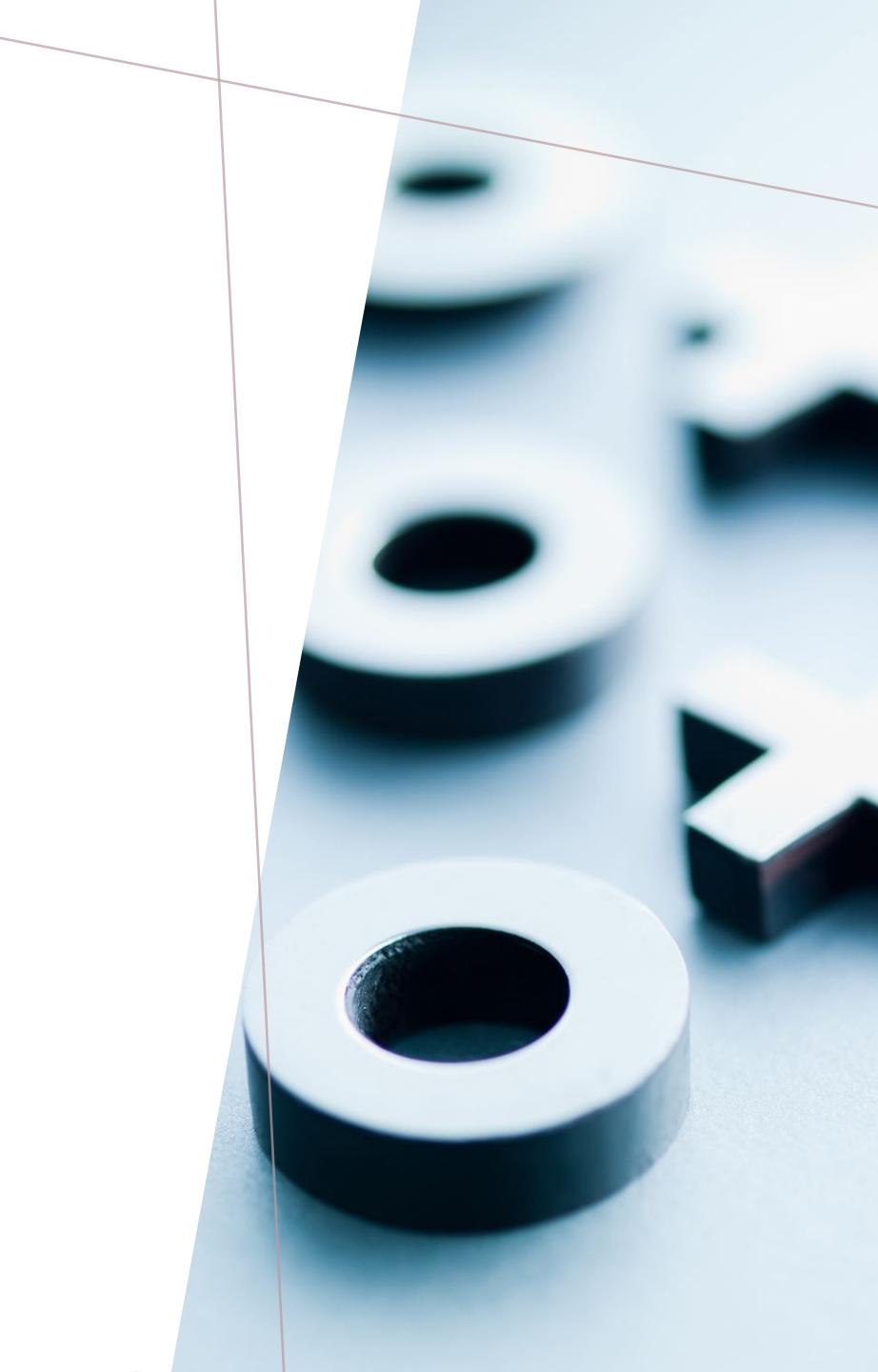
```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |  <meta charset="UTF-8">
5  |  <meta name="viewport" content="width=device-width, initial-scale=1.0">
6  |  <title>My Website</title>
7  </head>
8  <body>
9  |  <h1>Hello</h1>
10 |  <script src="index.js"></script>
11 </body>
12 </html>
```

```
1  alert("hello");
```

# EXTERNAL JAVASCRIPT

YOU WILL INSERT THE SCRIPT TAG WITH A EXTERNAL SOURCE.

*WHAT HAPPENS IF I  
WRITE THE CSS LINK  
TAG AT THE END OF  
THE FILE?*



# QUERYSELECTOR

DOM

- index.js
- indexDOM.html
- indexDOM.js
- indexexternal.html
- indexinline.html
- indexinternal.html
- style.css

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>My Website</title>
7 </head>
8 <body>
9   <h1>Hello</h1>
10  <script src="indexDOM.js"></script>
11
12 </body>
13
14 </html>
```

DOM

- index.js
- indexDOM.html

```
1 document.querySelector("h1").innerHTML = "Good Bye";
```

Good Bye

# ***DOCUMENT OBJECT MODEL***

**DOMINATING THE DOM TO ADD  
FUNCTIONALITY TO HTML ELEMENTS.**

# STATIC WEBSITES.

We basically planned what content our website should have and also the appearance of that content.

We wrote the HTML and the CSS code, then we hit save and refreshed our browser, and there is our site.

If we want our website to be interactive, then we need to be able to change parts of the website on the fly, that means when a user clicks on a button, we will need to respond to that by changing the content or the appearance of our website.

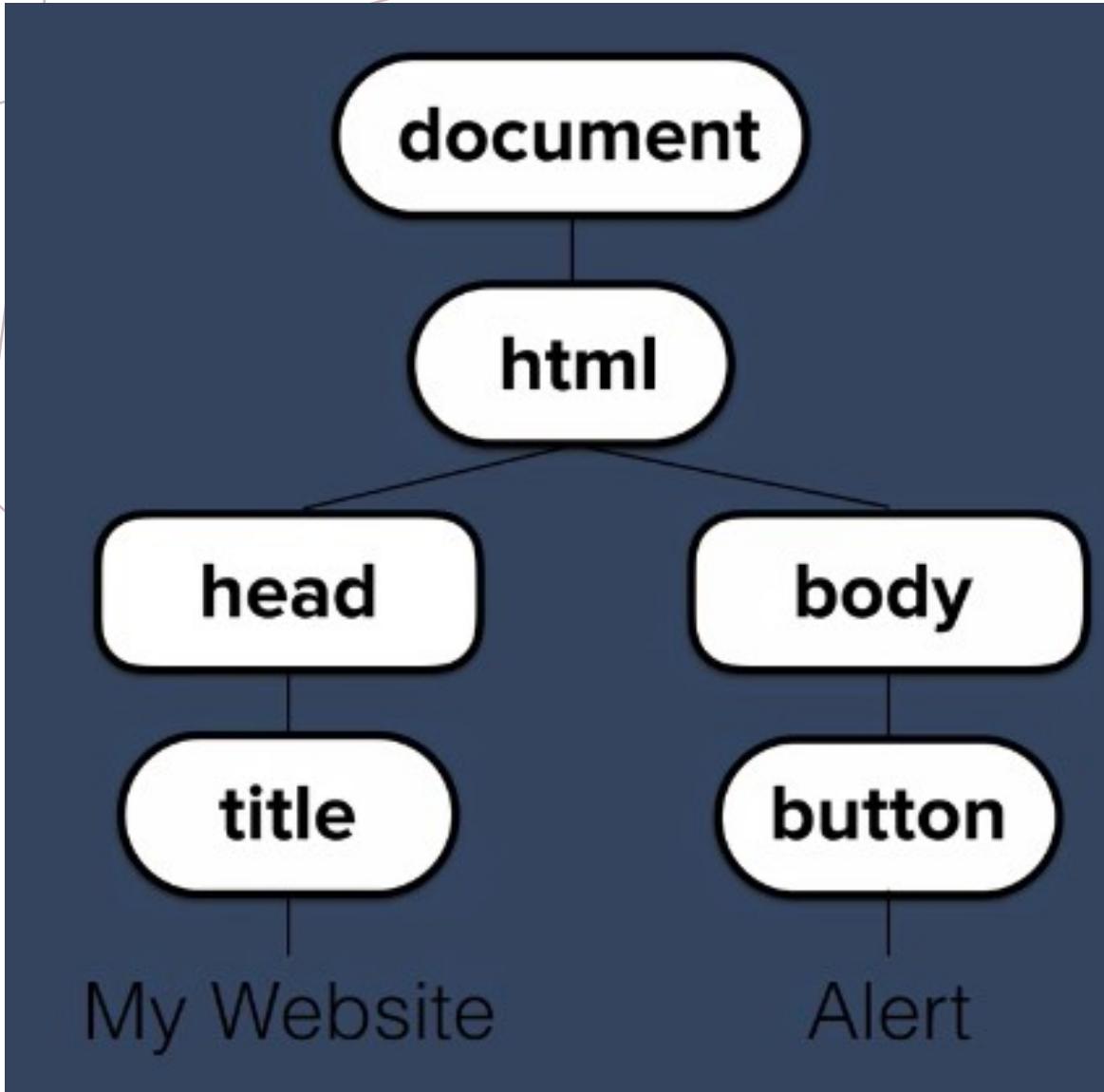
# DOM

Catalogs the web page into individual objects that we can select and manipulate.

```
<html>  
  <head>  
    <title>My Website</title>  
  </head>  
  <body>  
    <button>Alert</button>  
  </body>  
</html>
```



On the left we have the html code of a really basic website. On the right is roughly the structure of our website that you might see in the browser.



- The task of converting an HTML file into the DOM is done by the browser when you load up the web page. It turns each of these elements and their associated data into a tree structure with a whole bunch of objects that you can select and manipulate.

The tree model on the left is usually how you will see the DOM represented. All of the elements in our HTML has been converted into objects, and their relationships to each other mapped out in the tree diagram.

Ex: the head section is a descendant of the HTML object but the head and the body, they are siblings, they are not descendants of each other and everything that is contained inside your HTML document is contained in an object called the document.

```
DOM
  index.js
  indexDOM.html
  indexDOM.js
  indexDOMEX.html      3
  indexDOMEX.js
  indexexternal.html
  indexInline.html
  indexInternal.html
  style.css
  styleDOMEX.css

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  ...<meta charset="UTF-8">
5  ...<meta name="viewport" content="width=device-width, initial-scale=1.0">
6  ...<title>my website</title>
7  ...<link rel="stylesheet" href="styleDOMEX.css">
8  </head>
9  <body>
10 ...
11 ...<h1>Hello</h1>
12 ...
13 ...<input type="checkbox" name="" id="">
14 ...<button style=":active: color:red;">Click me</button>
15 ...
16 ...<ul>
17 ...  ...<li class="list">
18 ...    ...<a href="https://www.google.com">google</a>
19 ...  ...</li>
20 ...  ...<li class="list">Second</li>
21 ...  ...<li class="list">Third</li>
22 ...</ul>
23 ...
24 ...<script src="indexDOMEX.js"></script>
25 ...
26 </body>
27 </html>
```

---

# Hello



Click me

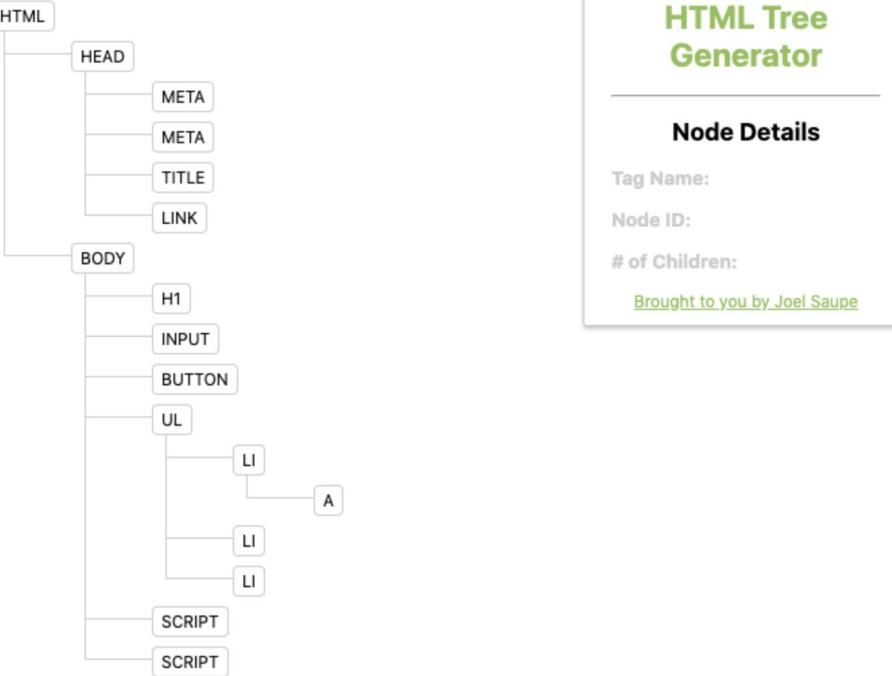
- [google](https://www.google.com)
- Second
- Third

# NAVIGATE THROUGH THE TREE

## Document

Document.firstChild: The firstElementChild property returns the first child element of the specified element.

How to select the h1?



Hello

Click me

- google
- Second
- Third

A screenshot of a browser's developer tools console, specifically the Elements tab, demonstrating the use of the `firstElementChild` property. The code and resulting output are as follows:

```
> document;
<- > #document
> document.firstChild
<- <html lang="en">
  > <head>...</head>
  > <body>...</body>
</html>
> document.firstChild.firstChild
<- > <head>...</head>
> document.firstChild.lastElementChild
<- > <body>...</body>
>
```

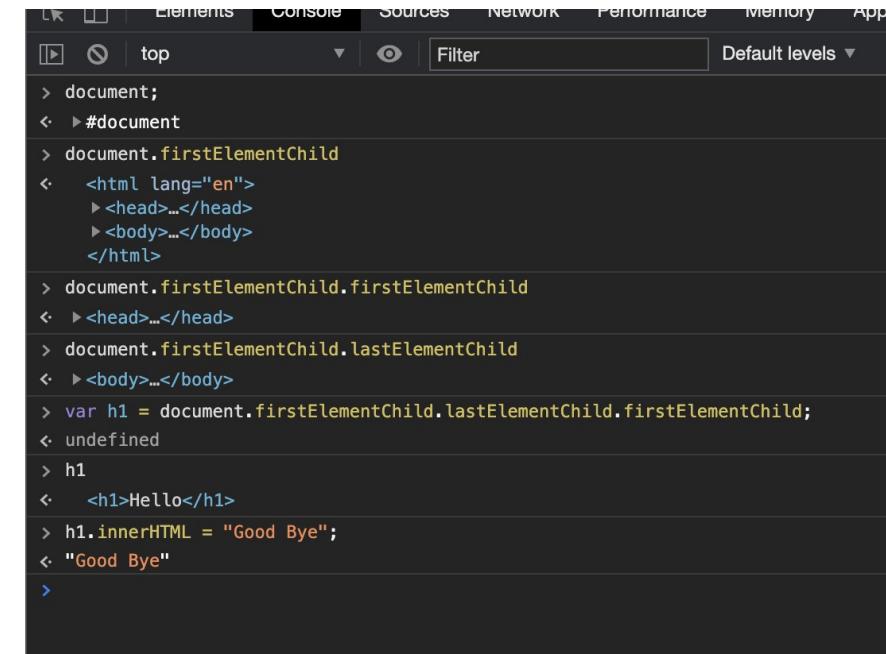
# *MANIPULATING THE DOM.*

- The innerHTML property sets or returns the HTML content (inner HTML) of an element.

**Good Bye**

Click me

- [google](#)
- Second
- Third



A screenshot of a browser's developer tools console. The console tab is selected. The code shown is:

```
> document;
<‐ > #document
> document.firstChild
<‐ <html lang="en">
  > <head>...</head>
  > <body>...</body>
</html>
> document.firstChild.firstChild
<‐ > <head>...</head>
> document.firstChild.lastElementChild
<‐ > <body>...</body>
> var h1 = document.firstChild.lastElementChild.firstChild;
<‐ undefined
> h1
<‐ <h1>Hello</h1>
> h1.innerHTML = "Good Bye";
<‐ "Good Bye"
>
```

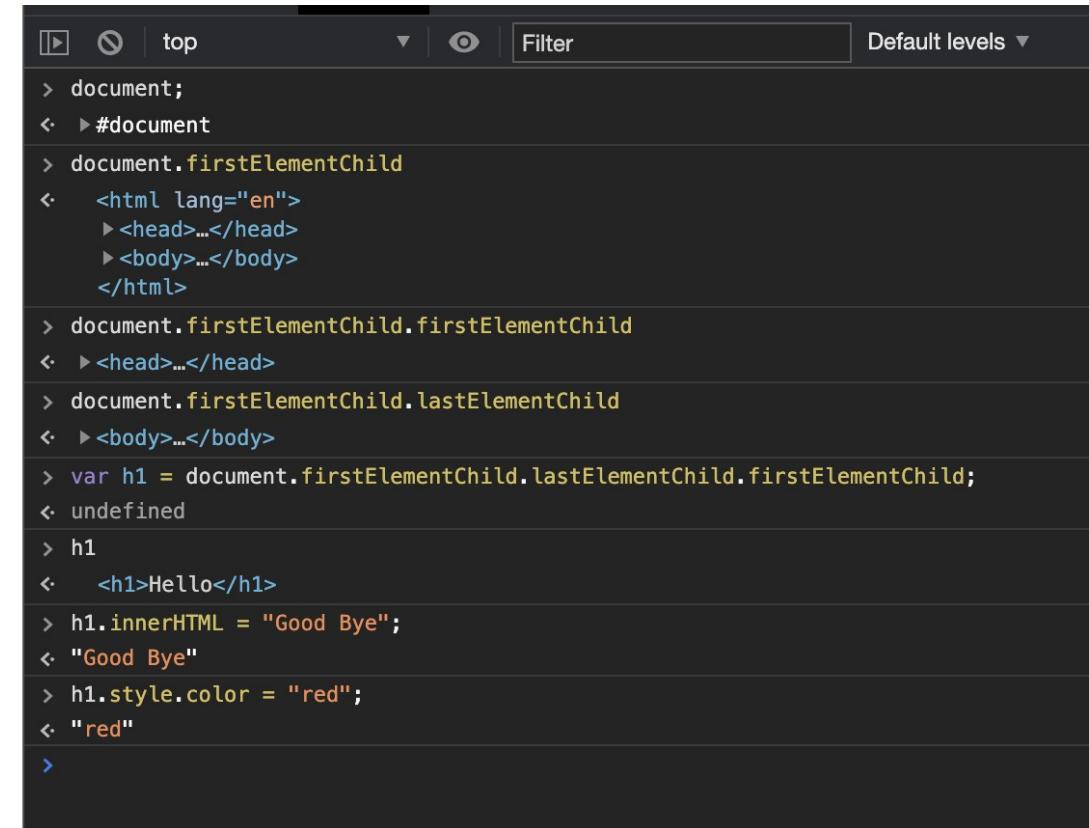
# *CHANGING THE STYLE ON THE FLY*

- *object.style.color*: The color property sets or returns the color of the text.

**Good Bye**

Click me

- [google](#)
- Second
- Third



The screenshot shows a browser's developer tools with the "Elements" tab open. The DOM tree on the left shows the following structure:

- > document;
- < > #document
- > document.firstChild
- < > <html lang="en">
- > <head>...</head>
- > <body>...</body>
- < /html>
- > document.firstChild.firstChild
- < > <head>...</head>
- > document.firstChild.lastElementChild
- < > <body>...</body>
- > var h1 = document.firstChild.lastElementChild.firstChild;
- < undefined
- > h1
- < > <h1>Hello</h1>
- > h1.innerHTML = "Good Bye";
- < "Good Bye"
- > h1.style.color = "red";
- < "red"
- >

The script editor at the bottom has the following code:

```
var h1 = document.firstChild.lastElementChild.firstChild;
h1.innerHTML = "Good Bye";
h1.style.color = "red";
```

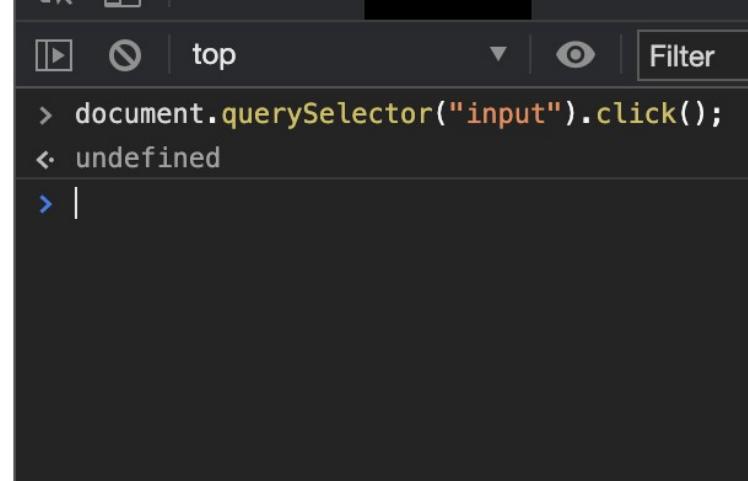
# DOCUMENT.QUERYSELECTOR

It looks into the entire document. In my example it looks for the object that has the selector of “input”. I called a method called click, what click does is that it simulates a mouse click. It will perform an action. It will select one element in the document.

Hello

Click me

- [google](#)
- Second
- Third



```
▶ document.querySelector("input").click();
◀ undefined
▶ |
```

Our objects inside the DOM, can have properties and methods.

Properties: describe something about the object.

Methods: are things that the object can do.

# *EXAMPLE:*



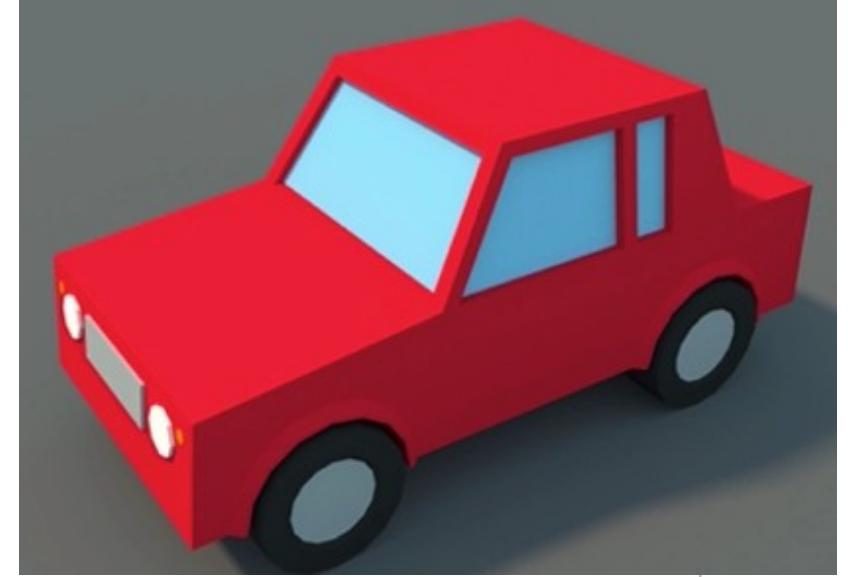
**Properties**

- Colour
- Number of seats
- Number of Doors

**Methods**

- `Brake()`
- `Drive()`
- `Park()`

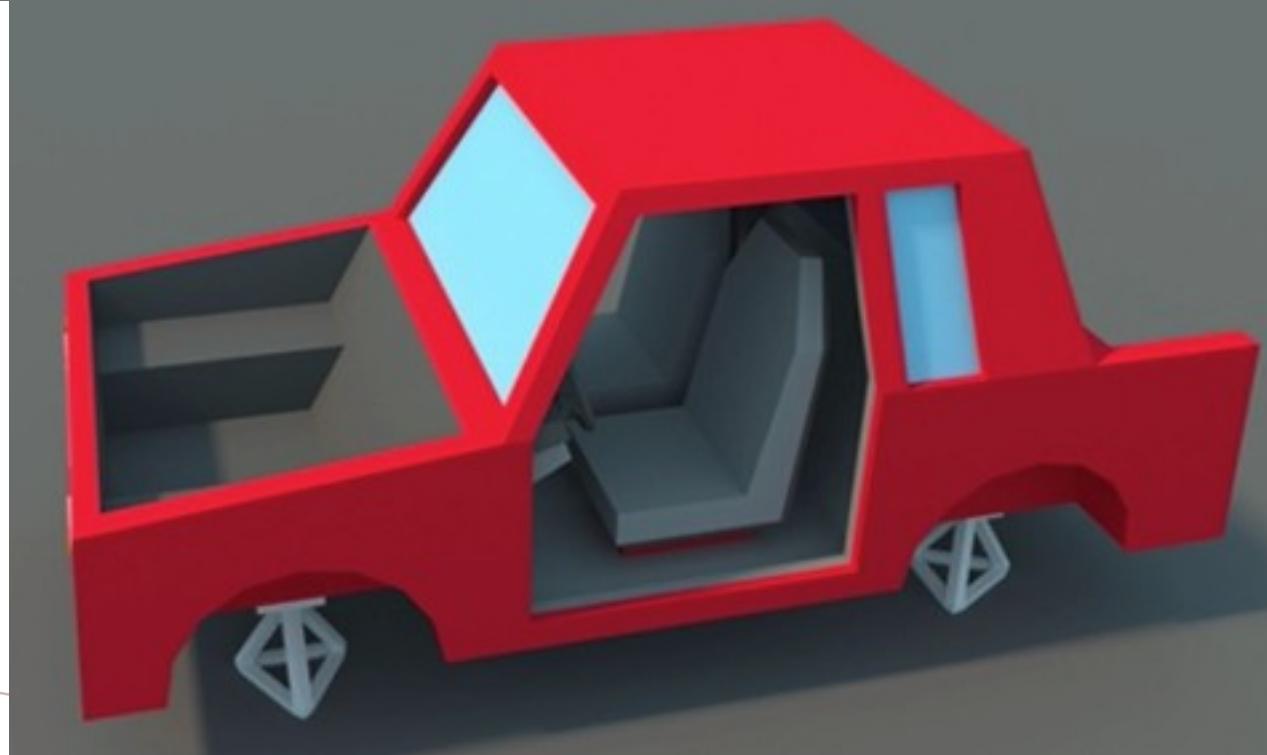
# *GET PROPERTY*



```
car.colour; //red
```

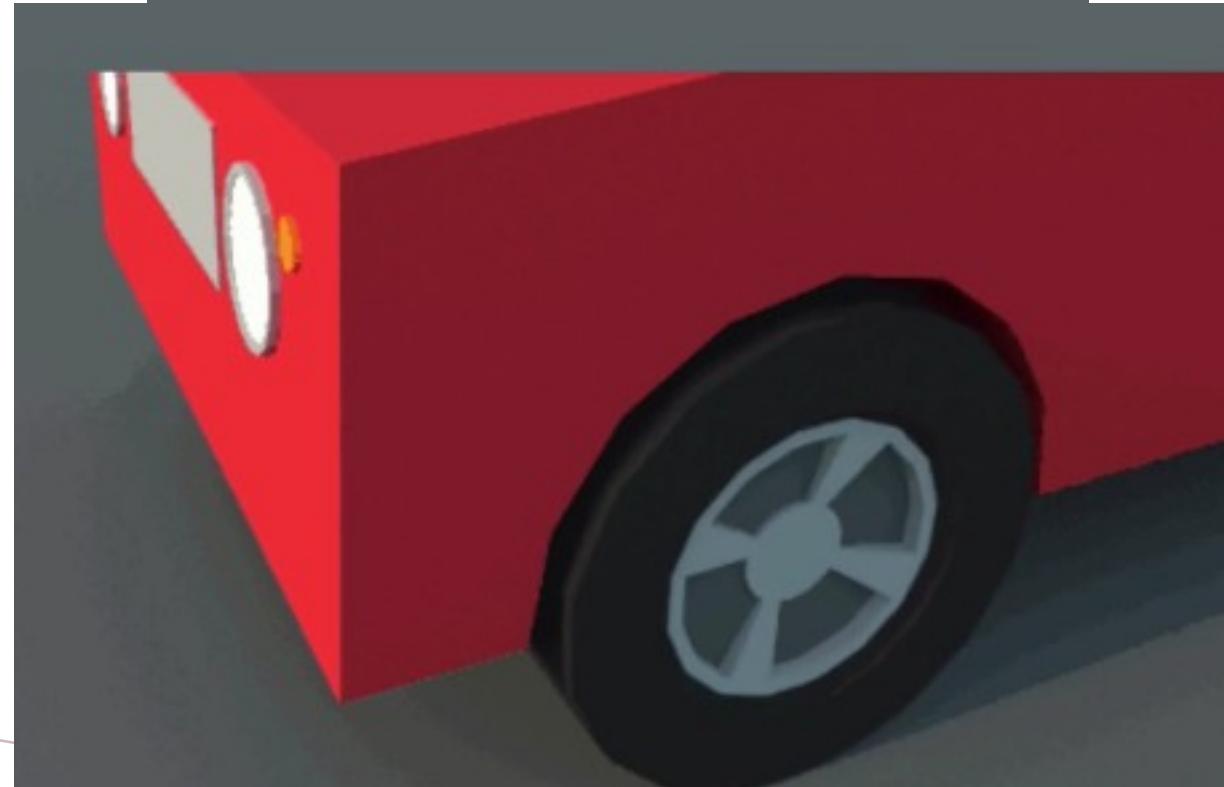
# *SET PROPERTY*

```
car.numberOfDoors = 0;
```



# *CALL METHOD*

```
car.drive();
```



*ADDED  
SOME  
CLASSES  
AND IDS*

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>my website</title>
7      <link rel="stylesheet" href="styleDOMEX.css">
8  </head>
9  <body>
10
11     <h1 id="title">Hello</h1>
12
13     <input type="checkbox" name="" id="">
14     <button class="btn" style=":active: color:red;">Click me</button>
15
16     <ul id="list">
17         <li class="item">
18             <a href="https://www.google.com">google</a>
19         </li>
20         <li class="item">Second</li>
21         <li class="item">Third</li>
22     </ul>
23
24 </body>
25 <footer><script src="indexDOMEX.js"></script></footer>
26 </html>
```

## *ALL DIFFERENT WAYS OF SELECTION ELEMENTS INSIDE THE DOM*

- `document.getElementsByTagName("li");` = you will get an array with all the list items. Please notice the word elements – its plural.  
It is going to select all the `li` elements on the page.

```
> document.getElementsByTagName("li");
<- ▶ HTMLCollection(3) {li.item, li.item, li.item}
  >           I
```

If you try to change the color of the array. You will get an error. You will have to specify the `li` you would like to change using the index of the array.

Hello

Click Me

- [Google](#)
- Second
- Third

```
> document.getElementsByTagName("li");
<- ▶ HTMLCollection(3) {li.item, li.item, li.item}
> document.getElementsByTagName("li").style.color = "purple";
✖ ▶ Uncaught TypeError: Cannot set property 'color' of undefined
  at <anonymous>:1:49
> document.getElementsByTagName("li")[2].style.color = "purple";
<- "purple"
>
```

- `document.getElementsByClassName("btn")` = allows you to select elements base on the name of the class. Please notice `getElements` – its plural so it will give you an array. That means chanaina the color will not work.

```
> document.getElementsByClassName("btn");
<  ▶ HTMLCollection [button.btn]
> |
```

Even if it is only one item with the class name you have to select using the index of the array.

```
> document.getElementsByClassName("btn").style.color = "red";
✖ ▶ Uncaught TypeError: Cannot set property 'color' of undefined
   at <anonymous>:1:52
> document.getElementsByClassName("btn")[0].style.color = "red";
```

- `document.getElementById("title")` = it is going to select by id.  
notice the word element is not longer plural. The reason why is because every single id in the web page should be unique. The web page should not have the same id for multiple elements.

## Hello

Click Me

- [Google](#)
- Second
- Third

```
> document.getElementById("title");
<<  <h1 id="title">Hello</h1>
> document.getElementById("title").innerHTML = "Good Bye";
<< "Good Bye"
>
```

## Good Bye

Click Me

- [Google](#)
- Second
- Third

- `document.querySelector("h1")` = it will return a single item. You have to specify a tag, class or id inside the parenthesis. Ex for id: `document.querySelector("#title")`. Ex for class `document.querySelector(".btn")`

```
> document.querySelector("h1");
<-- <h1 id="title">Good Bye</h1>
> document.querySelector("#title");
<-- <h1 id="title">Good Bye</h1>
> document.querySelector(".btn");
<-- <button class="btn" style="color: red;">Click Me</button>
```

## Combining selectors

```
<h1 id="title">Hello</h1>
<a href="https://www.google.com">Google</a>
<input type="checkbox">

<button class="btn">Click Me</button>

<ul id="list">
  <li class="item"><a href="https://www.google.com">Google</a></li>
  <li class="item">Second</li>
  <li class="item">Third</li>
</ul>
```

```
> document.querySelector("li a");
<-- <a href="https://www.google.com">Google</a>
> |
```

- `document.querySelectorAll("")` – it will return a list of all the item you specify. If you want to change the attribute of the item. You will have to specify its index.

```
> document.querySelectorAll("#list .item");
< NodeList(3) [li.item, li.item, li.item]
> document.querySelectorAll("#list .item")[2].style.color = "blue";
< "blue"
```

# *HOW TO USE JS TO MANIPULATE THE ELEMENTS WE SELECTED.*

- You can change the style of the property using the DOM.
- The value of the attribute has to be a string.

```
> document.querySelector("h1").style.color = "red";
< "red"
> document.querySelector("h1").style.fontSize = "10rem";
< "10rem"
> document.querySelector("h1").style.padding = "30%"
```

# *THE SEPARATION OF CONCERNS: STRUCTURE VS STYLE VS BEHAVIOR*

classList – will add classes to an element.

```
> document.querySelector("button").classList.add("invisible");
<- undefined
> document.querySelector("button").classList.remove("invisible");
<- undefined
> document.querySelector("button").classList.toggle("invisible");
```

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  > <head>...</head>
  > <body>
    <h1 id="title">Hello</h1>
    <a href="https://www.google.com">Google</a>
    <input type="checkbox">
    .. <button class="btn invisible">Click Me</button> = $0
      > <ul id="list">...</ul>
      > <footer>...</footer>
    </body>
  </html>
```

# TEXT MANIPULATION

- innerHTML: give you the html between the element tags.
  - <h1 id="title"><strong>hello</strong></h1> = output:  
<strong>hello</strong>
- textContent: will give you the text content.

```
> document.getElementById("title").innerHTML = "Good Bye";
<- "Good Bye"
> document.getElementById("title").textContent = "Hello";
<- "Hello"
> |
```

# ATTRIBUTE MANIPULATION

- Reminder: classes is an attribute, href is an attribute and src is an attribute.

```
> document.querySelector("a");
<  <a href="https://www.google.com">Google</a>
> document.querySelector("a").attributes;
<  ▷ NamedNodeMap {0: href, href: href, length: 1}
> document.querySelector("a").getAttribute("href");
<  "https://www.google.com"
> document.querySelector("a").setAttribute("href", "https://www.bing.com");|
```