**JavaScript and** [**The Document Object Model**](https://www.w3schools.com/jsref/dom_obj_style.asp)

1. Inline JavaScript – avoided as much as possible.
2. Script tag is just like internal CSS. <script type = “javascript”</script>
3. External file. Index.js

JS uses camelCase while it is different in CSS separated by dashes or CSS. All the property names are given in DOM Style Object documentation. The values have to be strings.

JS is kept at the end, while CSS is kept at the start, as JS is used to alter CSS and HTML, so we need to load CSS beforehand. Also, atleast loading of the content should load up before JS Script is read line by line, which makes the loading of our website faster.

**DOM:**

* DOM is catalogue in tree structure of the HTML code as tag elements being converted to object with their relation between each other.
* An object in the DOM can have Properties and Methods.
* document.firstElementChild.lastElementChild.lastElementChild.lastElementChild.innerHTML = "Angela"

**Ways to access HTML Elements and manipulate style**

* document.getElement**s**ByTagName(“selector name”) <- Fetches an array of all the element objects with this particular name.
* document.getElement**s**ByClassName(“btn”) <-We again get all the elements with this specific class.
* document.getElementById(“selector name”)<- this returns a single item which uniquely identifies with the id.
* document.querySelector(“selector name”) <- This returns a single item. We find it by using selectors instead of tag or class or id. The argument here can be class, id, tag name, and we can also combine our selectors.   
  **Space separated** – hierarchical selection. We are looking for the element at the end.  
  **dot separated** – an element which has both. Class A and class B.  
  If there are multiple items with the same selector query, it will return only the first one.
* document.querySelectorAll(“selector name”) <- This one returns all the items satisfying the query as an array.
* document.getElementById(“title”).innerHTML = “Change the text”;

**Separation of Concerns – Structure vs style vs behaviour.**

This means that we should use HTML for structure and only structure, CSS for only styling and JS only for behaviour. But since a lot of behaviour is managed by means of styling, so we need to do something about it. Three functions we can use to update the styling through JS as a part of behaviour, instead of making actual changes in the existing css through JS forever are-

* document.querySelector(“button”).classList<-returns the list of classes in the element.
* document.querySelector(“button”).classList.add(“myclassname”) <- adds a class to the html for which we might have added CSS already.
* document.querySelector(“button”).classList.remove(“myclassname”) <-removes it.
* document.querySelector(“button”).classList.toggle(“myclassname”) <-toggles the presence in the list of given class.

**Text and text content property Manipulation**

* document.getElementById(“title”).innerHTML = “Change the text”; <- It will allow us the content within the tag which has the selector, it will return everything between the tags, even if it is other tags and more html code.
* document.getElementById(“title”).textContent – instead returns only the simple language text between the tags.

[**HTML element attributes manipulation**](https://www.w3schools.com/jsref/dom_obj_style.asp)

* Additionalities of a tag are attributes like classs, href, src etc.
* document.querySelector(“a”).getAttribute(“href”)
* document.querySelector(“a”).setAttribute(“href”, “ The new value for the attribute you just picked.” )

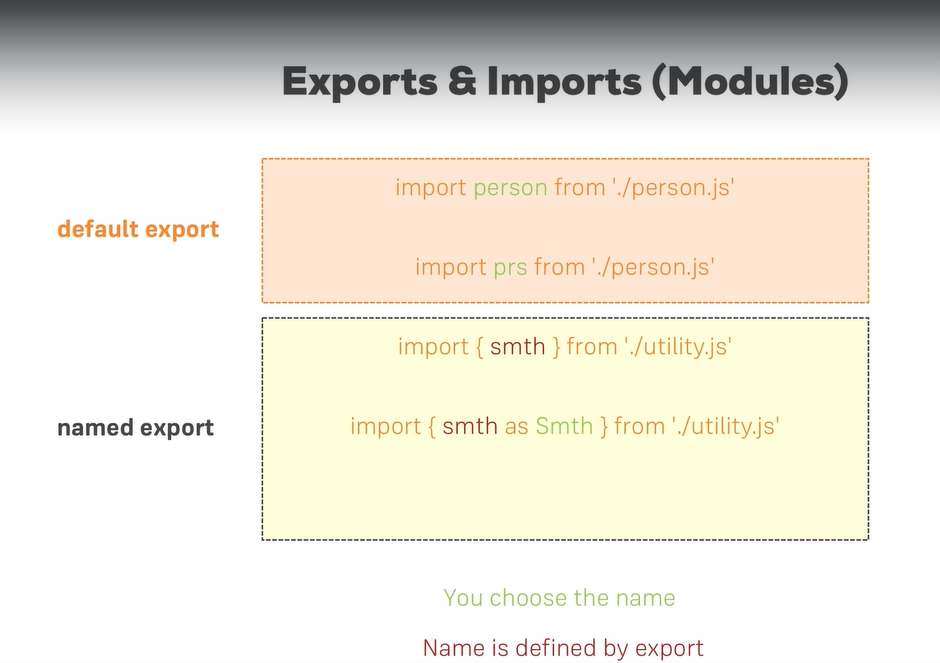
Adding audio to the file as an audio object: var snare = new Audio("sounds/snare.mp3"); snare.play();

[Arrow Function](https://youtu.be/h33Srr5J9nY)

[Promises](https://youtu.be/DHvZLI7Db8E)

[Async and Await functions](https://blog.webdevsimplified.com/2021-11/async-await/)

[Async, await and callbacks](https://youtu.be/bWaucYA1YRI) <- to solve problems which have time dependent resource calls using asynchronous programming.



JS Inheritance example:

class Human {

constructor(){

this.gender = “M”;}

printGender(){ console.log(this.gender);}

}

Class Person extends Human{

constructor() { super();

this.name = ‘Max’;

this.gender = “F”;}

printName(){

console.log(this.name);}}

const obj = new Person();

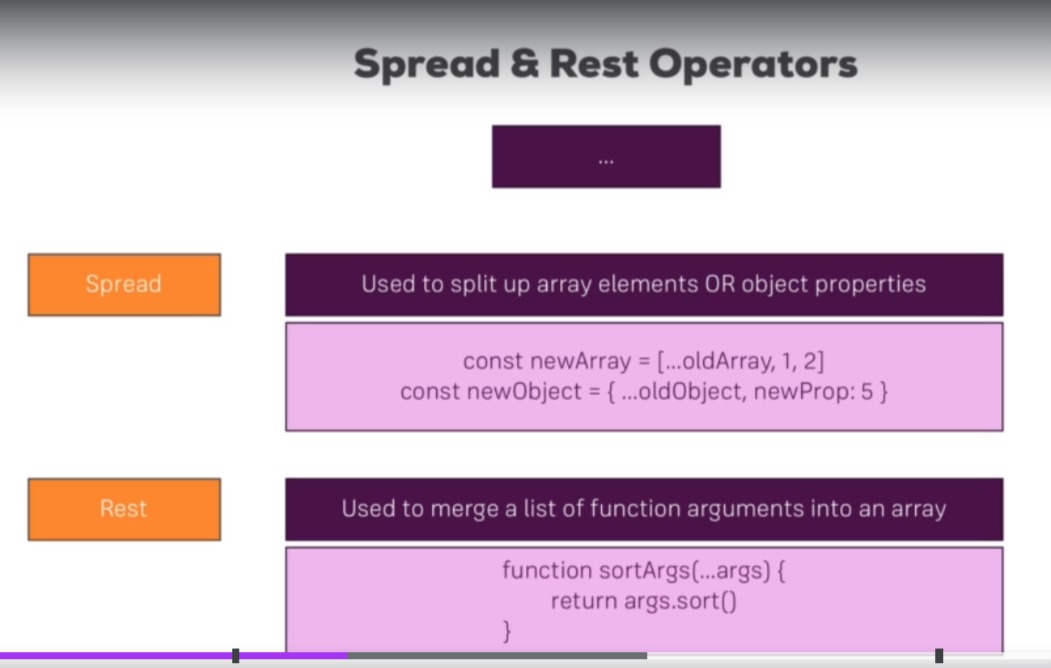
person.printName();

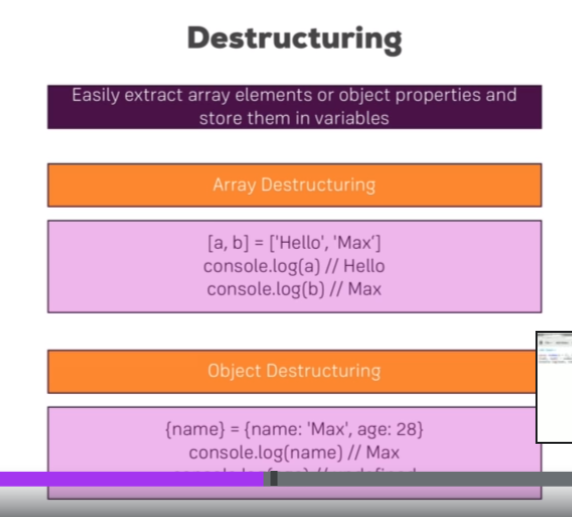
person.printGender();

*output:*

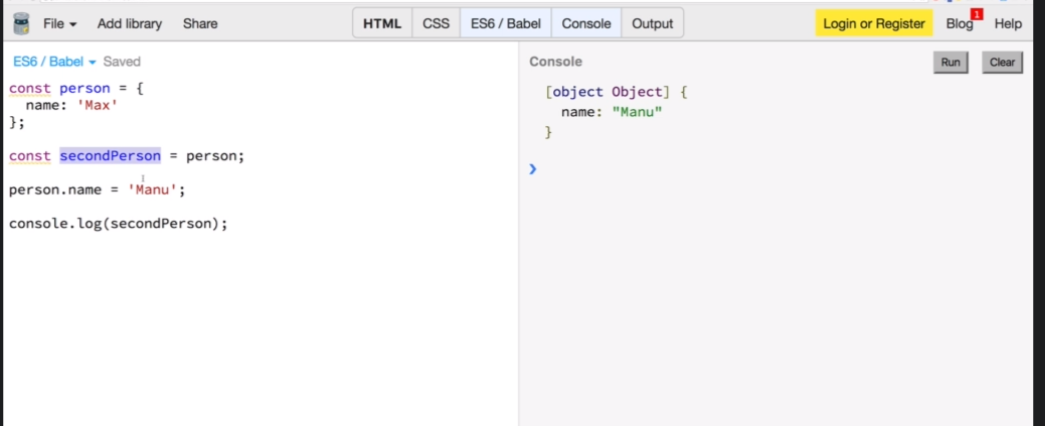
*“Max”*

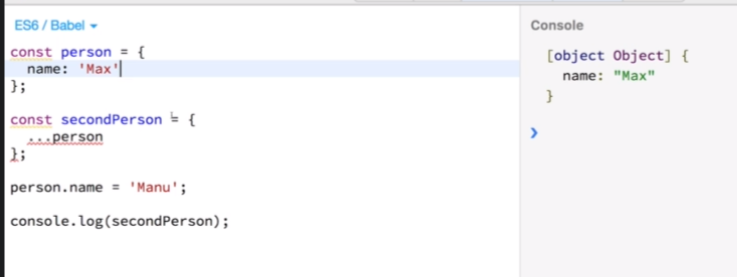
*“F”*

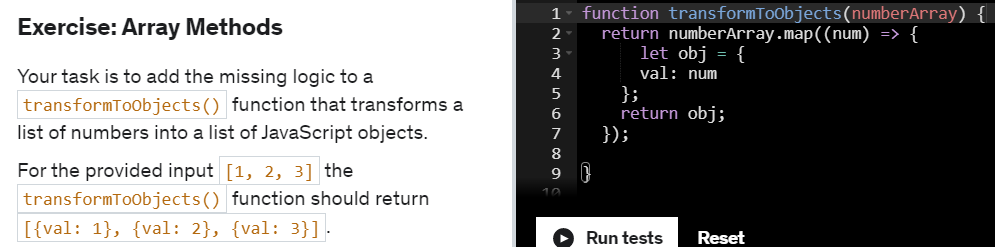




Reference data types in JavaScript ES7:

 The secondPerson constant is only a reference to the constant person. So even if we changed the person after saving its values in secondPerson, it wil change everywhere.

 DEEP COPYING



Other Array methods:

**JS Array Functions**

Not really next-gen JavaScript, but also important: JavaScript array functions like map() , filter() , reduce()  etc.

You'll see me use them quite a bit since a lot of React concepts rely on working with arrays (in immutable ways).

The following page gives a good overview over the various methods you can use on the array prototype - feel free to click through them and refresh your knowledge as required: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array>

Particularly important in this course are:

* map()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map>
* find()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/find>
* findIndex()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex>
* filter()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter>
* reduce()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce?v=b>
* concat()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/concat?v=b>
* slice()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/slice>
* splice()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice>