There are various ways of selecting a HTML Element or a node from a DOM tree

document.getElementById( )

It will give us the node having the particular ID value

document.querySelector( )

It will give the very first node it encounter in the Dom tree moving fro top to bottom

document.querySelector( “h1” );

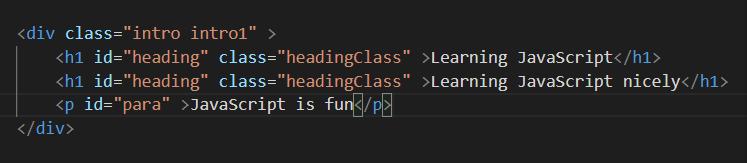
🡪 It will now give us the very first encountered h1

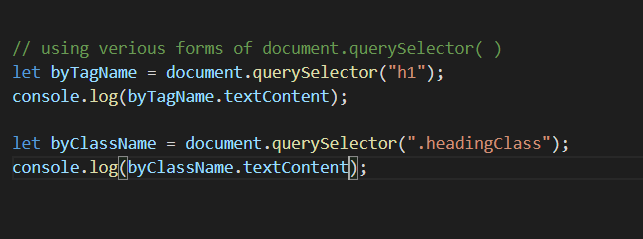
document.querySelector(“ .heading ”);

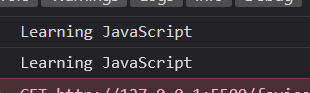
🡪 It will now give us the very first encountered node having the .heading class

document.querySelector(“ #heading ”)

* It will now give us the very first encountered node having the ID value of “heading”



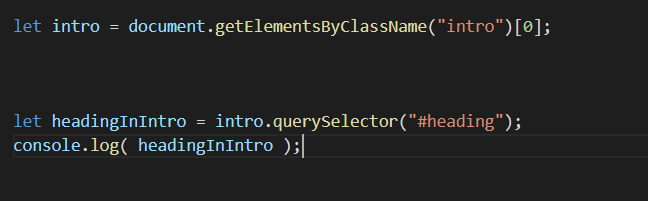


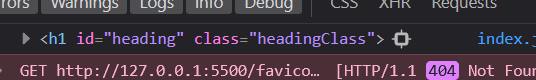


**What was the need of document.querySelector ( )**

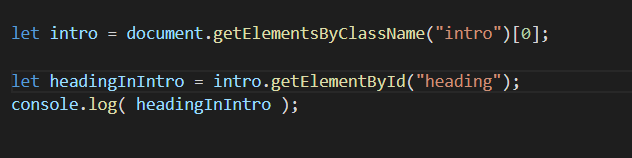
**Reason 1**

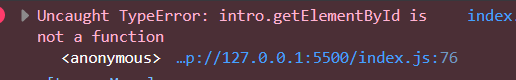
.querySelector( ) method can be applied with every ElementNode object not just only with document. Whereas others methods applies only to document





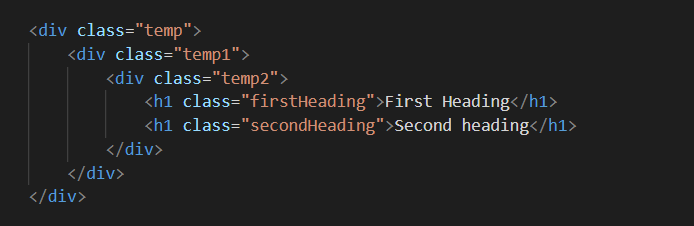
We can’t do this with other methods like





**Reason 2**

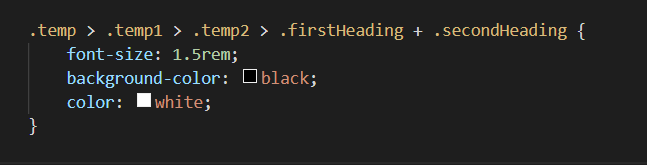
**let say we made a DOM tree like this**



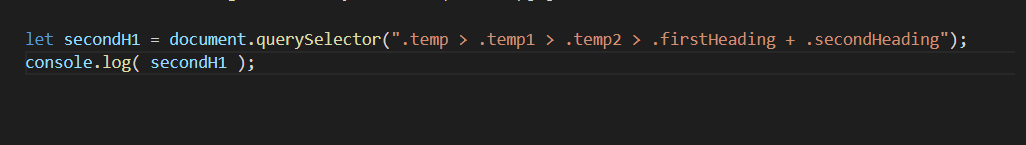
Note : Here it is in HTML format but when it is parsed by browsers it gets changed to a DOM tree

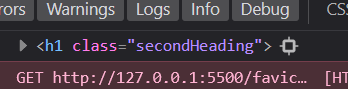
Now let say we wanna select the h1 with class “secondHeading”

We can do this by css as



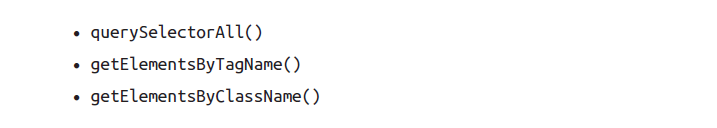
Now by querySelector( ) we can use this very css selector to get this element node





Actually when we were doing querySelector(“.className”) or (“#IdName”)

That time also we were doing selection by css, since classes and IDs are represented that way in css



**.querySelectorAll ( )**

gives us just a static NodeList ( a numbered object, which contains our node objects )

.**getElementsByTagName( ) or .getElementsByClassName ( )**

gives us a live HTML Collection ( a numbered object, which contains our node objects )

**But unlike getElementById( )**

**getElementsByClassName( ) and getElementsByTagname ( )**

are also defined in each node objects

