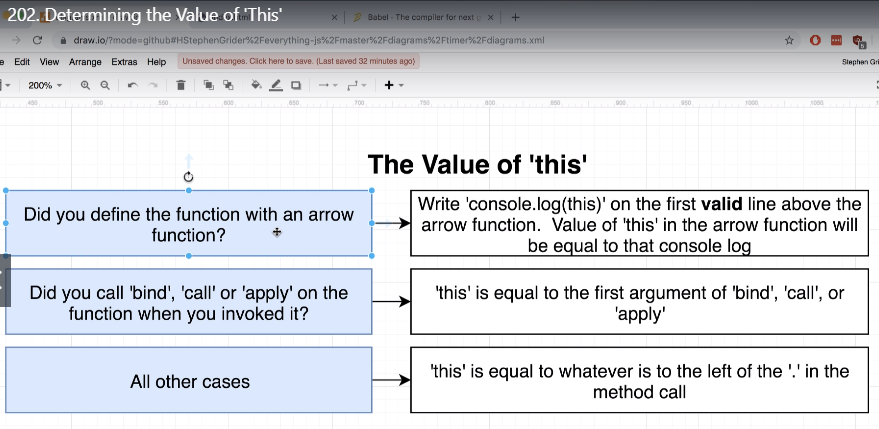


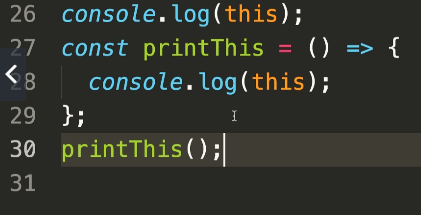
This is to make a timer folder and open it in VSC

Learning about THIS. When trying to identify what the value of THIS refers to, we go through boxes 1 through 3, top to bottom to test if THIS applies there. If not, we check the next box until we find our answer.

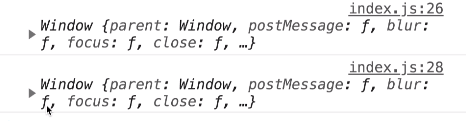


**The value of this first box-** this inside an arrow function

Example 1: the arrow function

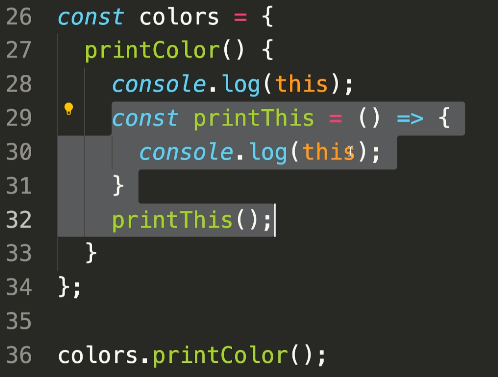


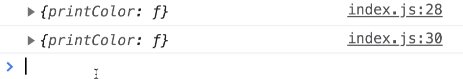
We made an arrow function and logged this inside it. In the first line of code above the arrow we logged this. Then ran the arrow function printThis.



Now in the console we see that in this case, THIS points to the window.

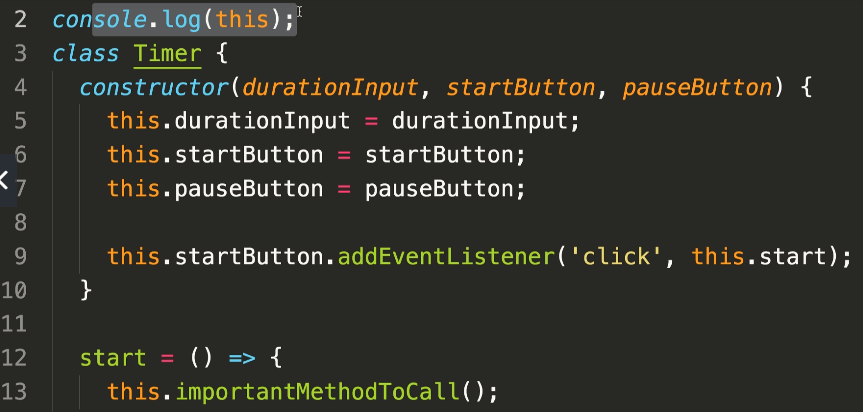
Example2



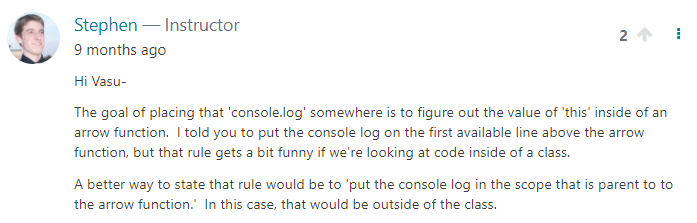


In this case, this on line 28 points to the print color function. So when we make this inside the arrow function, it also points to the print color function

Example 3

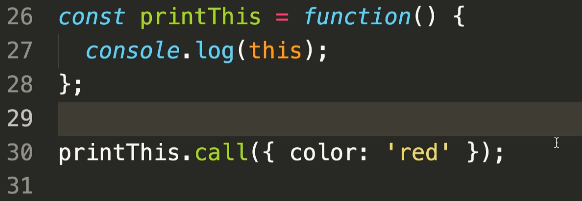


Our arrow function is line 12. The first VALID line of code isn’t line 11, it has to be outside of the class Timer. So it’s line 2.



**The value of this second box**- bind/call/or apply are built in functions that belong to js

Example 1: this is not in an arrow function this time

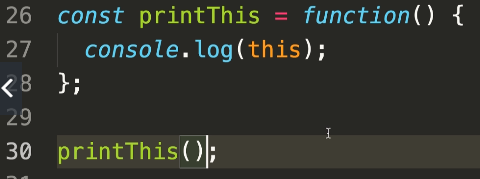


//the value of this, is equal to the object color:red

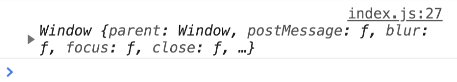


It’s the same if we used .apply instead of .call

Example 2:

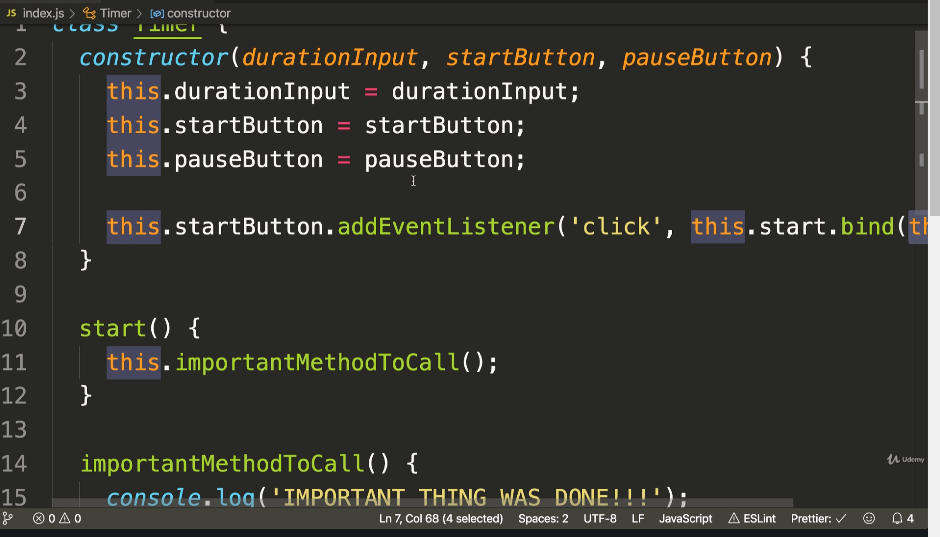


This time we don’t use .call or .apply, we just call the function directly:



Now it points to the window.

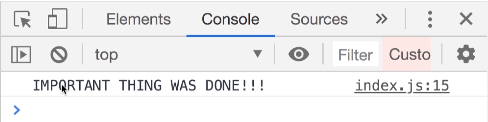
Example 3



Line 7:

this.startButton.addEventListener(‘click’,this.start.bind(this));

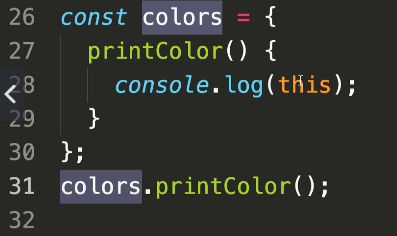
now, the value of THIS in the start function (line 11) is going to be forced to be the value of THIS that’s inside the constructor. The constructor is always going to be equal to the instance of the class.



So we learned that by using bind, call, or apply, the value of THIS overrides the original value of this inside the function, with whatever value we place in the parameters. Otherwise, when we don’t use bind call or apply, THIS will refer to the window like we see here.

**The value of this third box**- all other cases. This is equal to whatever is to the left of the . in the method

Example 1



First we check if THIS falls in the first 2 “boxes”, it’s not in an arrow function. And it’s not a bind,call,or apply function called. So it falls in the last box category: other. So we look to the left of the . when we call the print color method. It’s colors. That means we’re referring to the colors object (which contains a function.



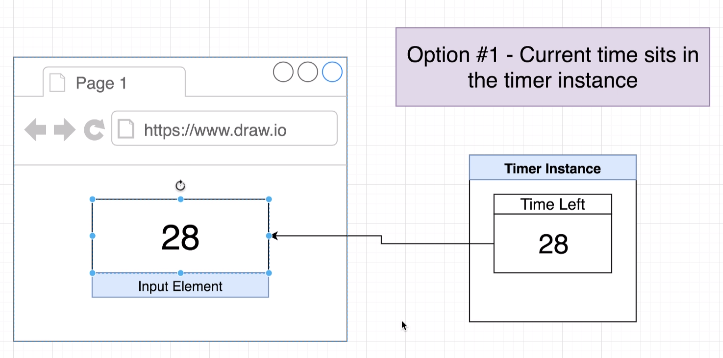
Example 2



Now when we call the method printColor(), it points to the randomobject, and it’ll print the data of a:1

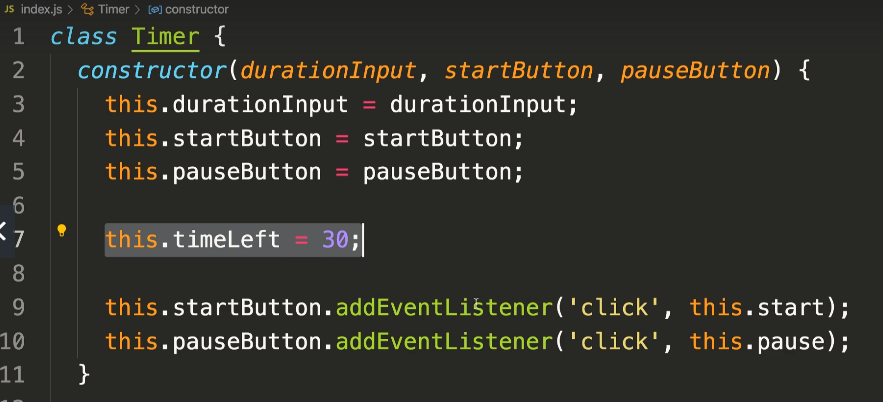


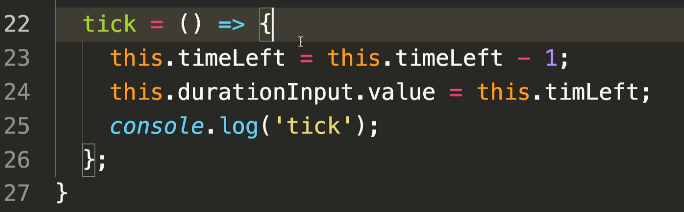
We want our tick method to count down the timer. This is one way to do it. It’s the more common way



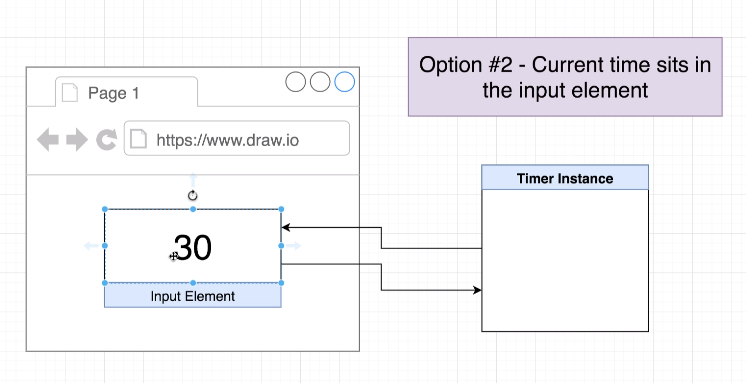
So an option number one in deciding where to store the time left on our timer we might decide to create an instance variable on our timer instance called something like time left.

Then every single time that tick method ran we could subtract one from that value and then take that value and update our text or submit our input element with it. So in other words when we run tick. We'll count down twenty nine to twenty eight then take the value of twenty eight and throw it into the input element does some code that looks like that might look like this:

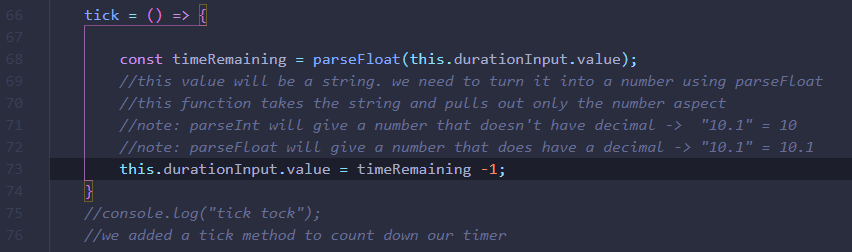


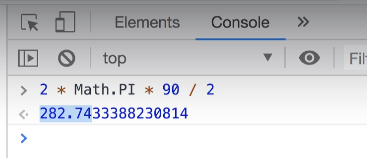


Second option:

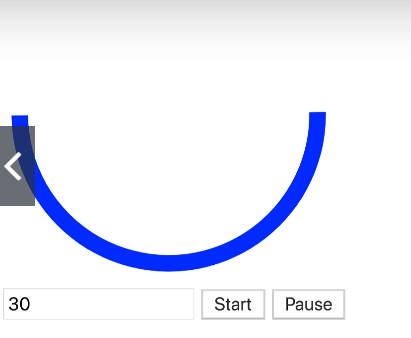


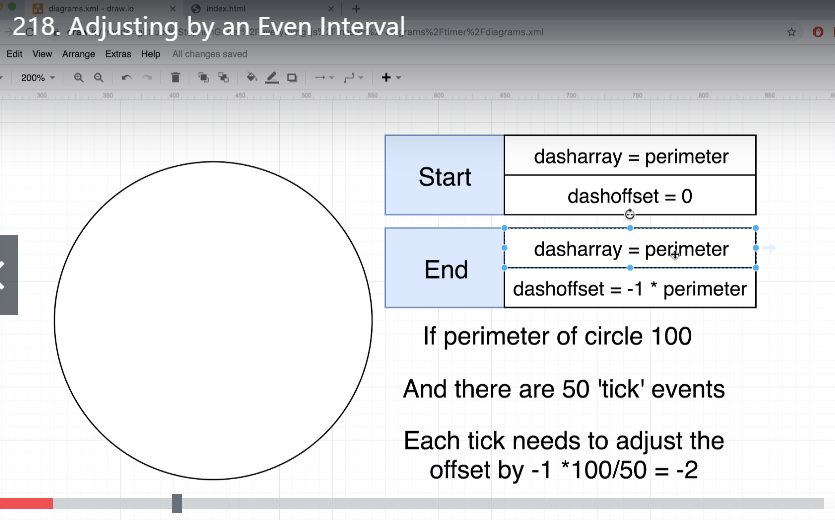
We’re going to store our current time into the input element:

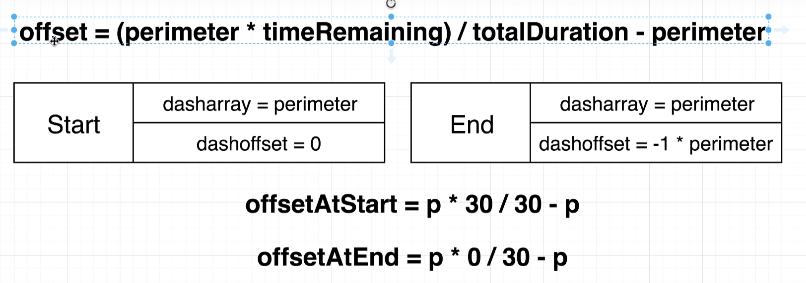




Leads to the image below.





How do we start dash offset of 0 and get to -perimeter?  


offsetAtStart = 0. offsetAtEnd comes to –p. this is the formula we need.

To run this, onTick method needs access to the time remaining.