**Chapter - 19 Events**

**Events** are functions executed at the time when a specific action occurs.

There are two types of events namely:

* Browser events
* Synthetic events

**Browser events:**

These are pre-determined and executed by the browser when an action occurs.

Here is the list of browser events:

* onClick()
* onmouseover()
* onmouseout()
* onmousedown()
* onmouseup()

**Synthetic Events:**

We can create and dispatch our own events using Event objects. In this way of creating events are called synthetic events.

Eg:

Let startEvent=new Event(‘start’);

document.addEventListener(‘start’,function(event){},false);

document.dispatchEvent(startEvent);

In the above example, addEventListener() is the event handlers. The syntax for the addEventListener() is

element.addEventListener(event,handler,[options]);

So event would be ‘start’ and the handler is a callback function, options can be a boolean, these options are not mandatory.

**Event capture and Event Bubbling:**

The last parameter is set to false. That is useCapture is set to false. If it is set to true, it means the parent element will be notified of the event first and only then the element that was actually clicked. If it false, event bubbling will be used, i.e first the clicked element will be notified of the event, and then the event will be dispatched progressively to all of its parents.

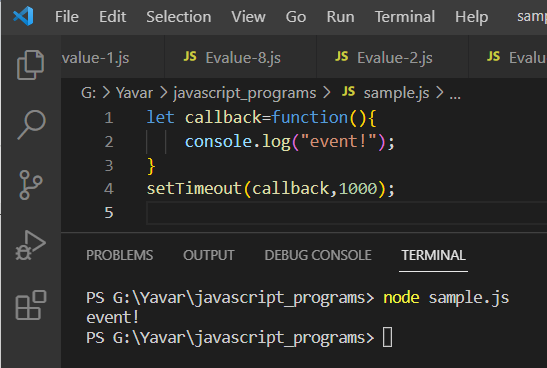
The callback function will be executed when the event occurs. Once the **addEventListener** function is executed the browser will be continuously listening for the “start” event to occur. But it remains dormant until event is actually dispatched using the dispatchEvent method.

One drawback in using addEventListener is if too many listeners are running at the same time it can affect the performance of the program.so if it is no longer required to listen to a listener it is a good idea to call **removeEventListener.**

**syntax :**

document.removeEventListener(“click”,callback);

**setTimeout:**



In the above example, the callback function will be executed after 1 sec when the setTimeout function is called.

**setInterval:**

This will work exactly like setTimeout, except it will continue executing the callback function for an indefinite number of times at a time interval specified as its second argument.

