EventEmitter Class

Many objects in a Node emit events, for example, a net.Server emits an event each time a peer connects to it. An fs.readStream emits an event when the file is opened. All objects which emit events are the instances of events.EventEmitter.

As we have seen in the previous section, EventEmitter class lies in the events module. It is accessible via the following code:

```
// Import events module
var events = require('events');

// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

When an EventEmitter instance faces any error, it emits an 'error' event. When a new listener is added, 'newListener' event is fired and when a listener is removed, 'removeListener' event is fired.

EventEmitter provides multiple properties like `on` and `emit`. `on` property is used to bind a function with the event and `emit` is used to fire an event.

Methods

- **addListener(event, listener)** : Adds a listener at the end of the listeners array for the specified event. No checks are made to see if the listener has already been added. Multiple calls passing the same combination of event and listener will result in the listener being added multiple times. Returns emitter, so calls can be chained.
- **on(event, listener)** : Adds a listener at the end of the listeners array for the specified event. No checks are made to see if the listener has already been added. Multiple calls passing the same combination of event and listener will result in the listener being added multiple times. Returns emitter, so calls can be chained.
- **once(event, listener)** : Adds a one time listener to the event. This listener is invoked only the next time an event is fired, after which it is removed. Returns emitter, so calls can be

chained.

- **removeListener(event, listener)** : Removes a listener from the listener array for the specified event. Caution − It changes the array indices in the listener array behind the listener. removeListener will remove, in most occasions, one instance of a listener from the listener array. If any single listener has been added multiple times to the listener array for the specified event, then removeListener must be called multiple times to remove each instance. Returns emitter, so calls can be chained.
- **removeAllListeners([event])** : Removes all listeners, or those of the specified event. It's not a good idea to remove listeners that were added elsewhere in the code, especially when it's on an emitter that you didn't create (e.g. sockets or file streams). Returns emitter, so calls can be chained.
- **setMaxListeners(n)** : By default, EventEmitters will print a warning if more than 10 listeners are added for a particular event. This is a useful default state which helps finding memory leaks. Obviously not all Emitters should be limited to 10. This function allows that to be increased. Set to zero for unlimited.
- **listeners(event)** : Returns an array of listeners for the specified event.
- **emit(event, [arg1], [arg2], [...])** : Executes each of the listeners in order with the supplied arguments. Returns true if the event had listeners, false otherwise.

Class Methods

listenerCount(emitter, event) : Returns the number of listeners for a given event.

- **Example:** create a file named listenerCount.js saved on your machine,

```
//import event module
var events = require('events');
// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();
eventEmitter.on('connection',function(){
    console.log("hello connection")
});
// count the number of listener to this event
eventListeners = require('events').EventEmitter.listenerCount(eventEmitter,'connecti
console.log(eventListeners + " Listner(s) listening to connection event");
```

Execute this piece of code by running the following command:

```
$ node listenerCount.js
```

- **Output:**

```
1 Listner(s) listening to connection event
```

Events

- newListener
  - **event** − String: the event's name
  - **listener** − Function: the event handler's function.

    This event is emitted any time a listener is added. When this event is triggered, the listener may have not yet been added to the array of listeners for the event.
- removeListener
  - **event** − String: The event's name.
  - **listener** − Function: The event handler's function.

    This event is emitted any time someone removes a listener. When this event is triggered, the listener may not yet have been removed from the array of listeners for the event.

Example

Create a .js file named main.js with the following Node.js code

```
var events = require('events');
var eventEmitter = new events.EventEmitter();

// listener #1
var listner1 = function listner1() {
   console.log('listner1 executed.');
}
```

```
// listener #2
var listner2 = function listner2() {
   console.log('listner2 executed.');
}

// Bind the connection event with the listner1 function
eventEmitter.addListener('connection', listner1);

// Bind the connection event with the listner2 function
eventEmitter.on('connection', listner2);

var eventListeners = require('events').EventEmitter.listenerCount
   (eventEmitter,'connection');
console.log(eventListeners + " Listner(s) listening to connection event");

// Fire the connection event
eventEmitter.emit('connection');

// Remove the binding of listner1 function
eventEmitter.removeListener('connection', listner1);
console.log("Listner1 will not listen now.");

// Fire the connection event
eventEmitter.emit('connection');

eventListeners = require('events').EventEmitter.listenerCount(eventEmitter,'connecti
console.log(eventListeners + " Listner(s) listening to connection event");

console.log("Program Ended.");
```

Now run the main.js to see the result.

```
$ node main.js
```

Verify the Output

```
2 Listner(s) listening to connection event

listner1 executed.

listner2 executed.

Listner1 will not listen now.

listner2 executed.

1 Listner(s) listening to connection event

Program Ended.
```