

모던 웹을 위한 NodeJs 프로그래밍

part2 : NodeJs 기본 - Chapter5 : 이벤트

2017.09.29

NodeJs 는 이벤트 기반 비동기 입출력을 내세우는 플랫폼입니다. 따라서 이벤트는 NodeJs에서 굉장히 중요한 요소 중 하나입니다.

본격적으로 들어가기 전에 알아두어야 할 개념

- `on()` : 이벤트를 연결하는 메소드
- `emit()` : 이벤트를 실행할 때 사용하는 메소드
- `EventEmitter` 객체 : 이벤트를 연결할 수 있는 모든 객체의 조상

이벤트 연결

위에서 언급하였듯이 NodeJs에서는 이벤트 연결 메소드로 `on()` 을 사용합니다.

```
on(EventName, eventHandler) // 이벤트를 연결
```

- `EventName` : 뜻 그대로 이벤트의 이름입니다. (`exit`, `uncaughtException`)
- `eventHandler` : 해당 이벤트가 발생하였을 때 호출되는 함수입니다.
`eventHandler`가 이벤트를 처리해준다고 생각하면 됩니다.

sample code

```
process.on('exit', function(code){  
    /*  
        code  
    */  
});
```

1. **exit** : 프로세스가 종료되기 전 발생하고 이러한 시점에서 종료하는 것을 막을 수 없으며 **exit** 리스너들의 동작이 끝난 후에 프로세스가 종료될 것입니다. 그러므로 **exit** 이벤트에 대한 핸들러는 반드시 동기적으로 작성하여야 합니다.
2. **uncaughtException** : 예외가 발생할 때 실행되는 이벤트
 - 실제 프로젝트에서는 **uncaughtException**은 사용하지 않고 **try-catch** 문을 사용하여 직접 예외처리를 해야합니다.

sample code

```
process.on('exit', function(code){
  console.log('Good Bye')
})

process.on('uncaughtException', function(error){
  console.log('Exception!')
})

var count = 0
var test = function(){
  count += 1
  if(count > 3){ return }

  setTimeout(test, 2000); // 2초뒤 뒤에 test 함수 실행.
  error.error.error() // Error Code
}

setTimeout(test, 2000) // 2초 뒤에 test 함수 실행
```

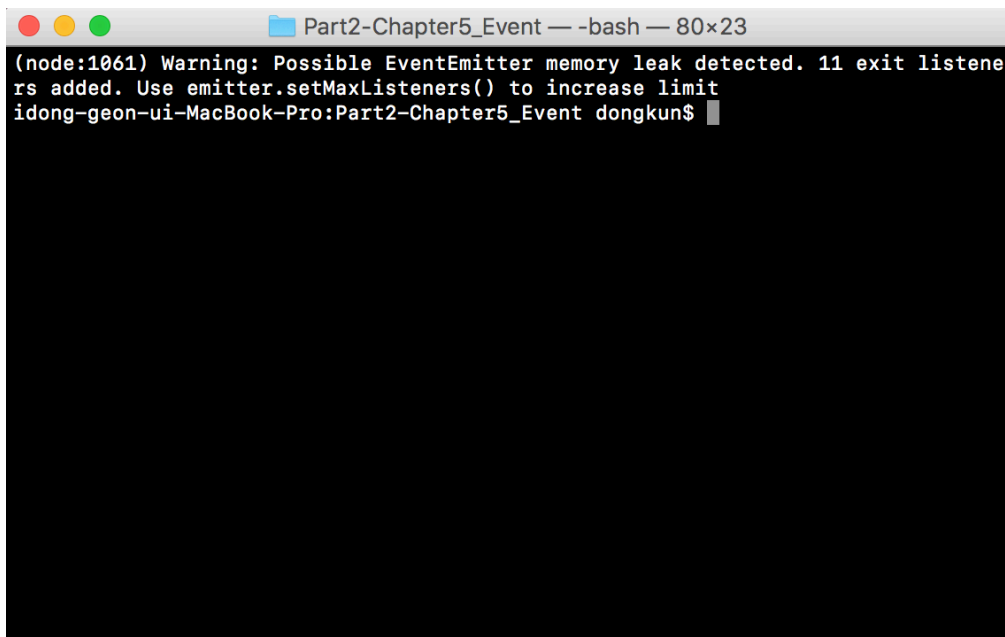
`error.error.error()`로 `uncaughtException` 이벤트가 발생합니다.

이벤트 연결 개수 제한

NodeJs는 한 이벤트에 10개가 넘어가는 이벤트 리스너를 연결할 경우 이를 개발자의 실수로 간주하여 정상적으로 작동을 하나 `warning`을 발생시킵니다.

```
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
```

위와 같이 `exit`이라는 이벤트에 같은 리스너를 11개를 등록하면 `warning`을 발생 시킵니다.

A terminal window titled "Part2-Chapter5_Event — -bash — 80x23" shows a warning message from Node.js. The message reads: "(node:1061) Warning: Possible EventEmitter memory leak detected. 11 exit listeners added. Use emitter.setMaxListeners() to increase limit". Below the message, the prompt "idong-geon-ui-MacBook-Pro:Part2-Chapter5_Event dongkun\$" is visible.

```
(node:1061) Warning: Possible EventEmitter memory leak detected. 11 exit listeners added. Use emitter.setMaxListeners() to increase limit
idong-geon-ui-MacBook-Pro:Part2-Chapter5_Event dongkun$
```

warning

이러한 경고는 `setMaxListeners()` 메소드로 이벤트 리스너 연결 개수를 높일 수 있습니다.

```
process.setMaxListeners(15)

process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
process.on('exit', function(code){})
```

리스너를 무한 개 연결하고 싶다면 매개변수로 0을 사용하면 됩니다.

이벤트 제거

이벤트를 제거할 때는 다음과 같은 메소드들을 사용합니다.

- `removeListener(eventName, handler)` : 특정 이벤트의 이벤트 리스너를 제거.
- `removeAllListeners([eventName])` : 모든 이벤트 리스너를 제거.

sample code

```

var onUncaughtException = function(error){
    console.log("예외가 발생했군, 이번에만 봐주겠다.");

    process.removeListener('uncaughtException',
onUncaughtException)

    // uncaughtException 이벤트가 한번 발생하고 나선 이벤트와 리스너의 연결을
    끊어버린다.
}

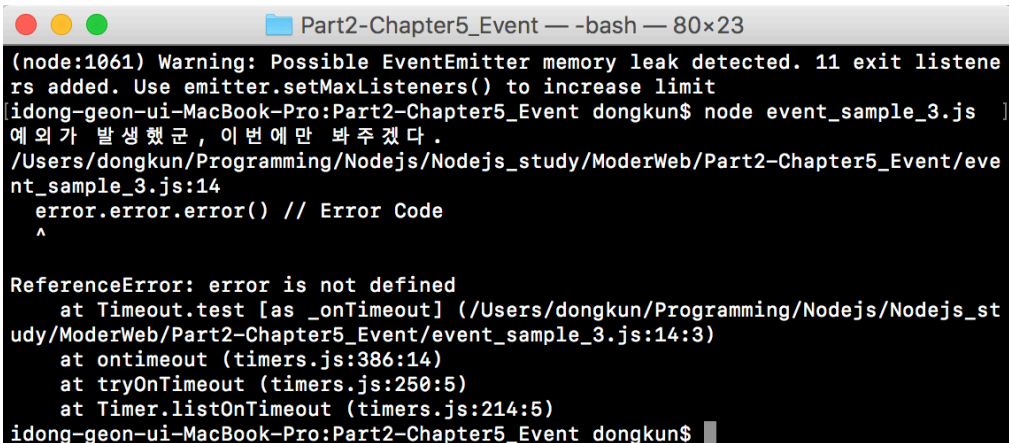
// process 객체에 uncaughtException 이벤트를 연결합니다.

process.on('uncaughtException', onUncaughtException)

var test = function(){
    setTimeout(test, 2000)
    error.error.error() // Error Code
}

setTimeout(test, 2000)

```



```

Part2-Chapter5_Event — -bash — 80x23
(node:1061) Warning: Possible EventEmitter memory leak detected. 11 exit listeners added. Use emitter.setMaxListeners() to increase limit
idong-geon-ui-MacBook-Pro:Part2-Chapter5_Event dongkun$ node event_sample_3.js
예외가 발생했군, 이번에만 봐주겠다.
/Users/dongkun/Programming/Nodejs/Nodejs_study/ModerWeb/Part2-Chapter5_Event/event_sample_3.js:14
  error.error.error() // Error Code
  ^
ReferenceError: error is not defined
    at Timeout.test [as _onTimeout] (/Users/dongkun/Programming/Nodejs/Nodejs_study/ModerWeb/Part2-Chapter5_Event/event_sample_3.js:14:3)
    at ontimeout (timers.js:386:14)
    at tryOnTimeout (timers.js:250:5)
    at Timer.listOnTimeout (timers.js:214:5)
idong-geon-ui-MacBook-Pro:Part2-Chapter5_Event dongkun$

```

remove event

사진을 보면 알 수 있듯이 한번의 이벤트 처리만 하고 에러코드로 인해 프로그램이 종료되었습니다.

위와 같이 딱 한번의 이벤트 발생에 대한 핸들러를 연결하고 싶다면 다음의 메소드를 사용하면 됩니다.

- `once(eventName, eventHandler)` : 이벤트를 리스너를 한 번만 연결합니다.

이벤트 강제 발생

이벤트를 강제로 발생시키는 방법도 존재하지만 약간 주의할 점이 있습니다.

- `emit(eventName, [, arg1], [, arg2] ...)` : 이벤트를 실행합니다.

```
// exit 이벤트 연결
process.on('exit', function(code){
    console.log("Good bye");
})

process.emit('exit')
process.emit('exit')
process.emit('exit')
process.emit('exit')

//
console.log("프로그램 실행 중");
```

```
Part2-Chapter5_Event — -bash — 80x23
idong-geon-ui-MacBook-Pro:Part2-Chapter5_Event dongkun$ node event_sample_4.js
Good bye
Good bye
Good bye
Good bye
프 로 그 램 실행 중
Good bye
idong-geon-ui-MacBook-Pro:Part2-Chapter5_Event dongkun$
```

결과

보시다시피 `emit` 메소드로 이벤트를 발생시켰을 때는 핸들러만 작동합니다. 실제로 해당 이벤트가 발생하지는 않습니다.

프로그램의 강제 종료는 `process.exit()`를 사용합니다.

이벤트 생성

NodeJs에서 이벤트를 연결할 수 있는 모든 객체는 `EventEmitter` 객체를 상속 받습니다. `EventEmitter` 객체는 `process` 객체 안에 있는 생성자 함수로 생성할 수 있는 객체입니다.

`EventEmitter` 객체의 메소드

- `addListener(eventName, eventHandler)`
- `on(eventName, eventHandler)`

- `setMaxListeners(limit)`
- `removeListener(eventName, handler)`
- `removeAllListeners([eventName])`
- `once(eventName, eventHandler)`

sample code

```
var EventEmitter = require('events')
var custom = new EventEmitter()
// var custom = new process.EventEmitter() ->
// DeprecationWarning
custom.on('tick', function(code){
  console.log("이벤트를 발생시킵니다.");
})

custom.emit('tick')

custom.on('dongkeon', function(code){
  console.log("dongkeon");
})

custom.emit('dongkeon')
```

```
Part2-Chapter5_Event — -bash — 80x23
idong-geon-ui-MacBook-Pro:Part2-Chapter5_Event dongkun$ node event_sample_5.js
이 이벤트를 발생시킵니다.
dongkeon
idong-geon-ui-MacBook-Pro:Part2-Chapter5_Event dongkun$
```

result

이렇게 `EventEmitter`에 새로운 이벤트를 추가하여 사용할 수 있습니다.