19. EventEmitter

Monday, June 12, 2023 1:06 PM

Concepts are similar to front-end

Events:
1. EventEmitter
2. EventListener

// Import events modules

```
const EventEmitter = require('events');

// Listen to Events => React accordingly
// Similar to setting up an Event Listener that listens to Button
clicks
// myEmitter will eventually emit a named event
const myEmitter = new EventEmitter();

// myEmitter.on = Observers
// Observers listen to Events
// myEmitter listens on newSale event, followed by a Callback func
myEmitter.on('newSale', () => {
    console.log('There was a new sale!')
})

// myEmitter listens on newSale event, followed by a Callback func
myEmitter.on('newSale', () => {
    console.log('Customer name: Jonas');
})

myEmitter.on('newSale', stock => {
    // amount of items left
    console.log(`There are now ${stock} items left in stock.`);
})

// Emitter
// An online store
// can pass 2nd arguments into an Emitter
myEmitter.emit('newSale', 9); // as if we're clicking on a button
```

```
$ node events.js
```

```
myEmitter.emit('newSale', 9); // as if we're clicking on a
button
```

```
$ node events.js
There was a new sale!
Costumer name: Jonas
There are now 9 items left in stock.
```

// myEmitter.on are run Synchronously

====Using ES6 class to construct a new class named 'Sales'

```
const EventEmitter = require('events');
// Listen to Events => React accordingly
// Similar to setting up an Event Listener that listens to
Button clicks
// myEmitter will eventually emit a named event
// const myEmitter = new EventEmitter();

// ES6
// class Sales inherits all classes from EventEmitter class
class Sales extends EventEmitter {
    constructor() {
        super();
    }

}
const myEmitter = new Sales();

// myEmitter.on = Observers
// Observers listen to Events
// myEmitter listens on newSale event, followed by a Callback
func
myEmitter.on('newSale', () => {
    console.log('There was a new sale!')
})
// myEmitter listens on newSale event, followed by a Callback
func
myEmitter.on('newSale', () => {
    console.log('Costumer name: Jonas');
})
myEmitter.on('newSale', stock => {
    // amount of items left
    console.log(`There are now ${stock} items left in stock.`);
```

```
})
// Emitter
// An online store or something
// can pass 2nd arguments into an Emitter
myEmitter.emit('newSale', 9); // as if we're clicking on a
button
```

```
$ node events.js
There was a new sale!
Costumer name: Jonas
There are now 9 items left in stock.
```

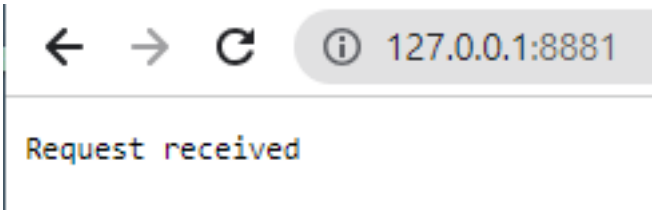
====Creating a small Web Server that listens to Events it emits

```
const EventEmitter = require('events');
const http = require('http');

// Listen to Events => React accordingly
// Similar to setting up an Event Listener that listens to
Button clicks
// myEmitter will eventually emit a named event
// const myEmitter = new EventEmitter();
// class Sales inherits all classes from EventEmitter class
// http, fs modules all implement inheritance of EventEmitter
internally
class Sales extends EventEmitter {
    constructor() {
        super();
    }

}
const myEmitter = new Sales();

// myEmitter.on = Observers
// Observers listen to Events
// myEmitter listens on newSale event, followed by a Callback
func
myEmitter.on('newSale', () => {
    console.log('There was a new sale!')
})
// myEmitter listens on newSale event, followed by a Callback
func
myEmitter.on('newSale', () => {
    console.log('Costumer name: Jonas');
})
```

```
myEmitter.on('newSale', stock => {
    // amount of items left
    console.log(`There are now ${stock} items left in stock.`);
})

// Emitter
// An online store or something
// can pass 2nd arguments into an Emitter
myEmitter.emit('newSale', 9); // as if we're clicking on a
button

// ======Create a small web server that listens to Events that
it emits
const server = http.createServer();

// Listens to different Events that the server will emit
server.on('request', (req, res) => {
    console.log('Request received! ');
    res.end('Request received');
})

server.on('request', (req, res) => {
    console.log('Another request received! ');
    res.end('Another request received');
})

// Listens to Server shutdown
server.on('Close', () => {
    console.log('Server closed');

})

const localhost = '127.0.0.1';
const port = 8881;
server.listen(port, localhost, () => {
    console.log(`Server has been started on ${localhost}:
${port}\nWaiting for requests...`);
})
```

```
There was a new sale!
Costumer name: Jonas
There are now 9 items left in stock.
Server has been started on 127.0.0.1:8881
Waiting for requests...
Request received!
Another request received!
```

```
Another request received!
Request received!
Another request received!
```



====Creating a small Web Server that listens to Events it emits

```
const EventEmitter = require('events');
const http = require('http');
const url = require('url');

// Listen to Events => React accordingly
// Similar to setting up an Event Listener that listens to
Button clicks
// myEmitter will eventually emit a named event
// const myEmitter = new EventEmitter();
// class Sales inherits all classes from EventEmitter class

// ES6
// http, fs modules all implement inheritance of EventEmitter
internally
class Sales extends EventEmitter {
    constructor() {
        super();
    }

}
const myEmitter = new Sales();

// myEmitter.on = Observers
// Observers listen to Events
// myEmitter listens on newSale event, followed by a Callback
func
myEmitter.on('newSale', () => {
    console.log('There was a new sale!')
})

// myEmitter listens on newSale event, followed by a Callback
func
```

```
myEmitter.on('newSale', () => {
    console.log('Costumer name: Jonas');
})

myEmitter.on('newSale', stock => {
    // amount of items left
    console.log(`There are now ${stock} items left in stock.`);
})

// Emitter
// An online store or something
// can pass 2nd arguments into an Emitter
myEmitter.emit('newSale', 9); // as if we're clicking on a
button

// ======Create a small web server that listens to Events that
it emits
const server = http.createServer();

// Listens to different Events that the server will emit
server.on('request', (req, res) => {
    console.log('Request received! ');
    console.log(req.url);
    res.end('Request received'); // Can only send 1 response
})
server.on('request', (req, res) => {
    console.log('Another request received! ');
})
// Listens to Server shutdown
// Server will NOT shutdown as long as it's still listening on
Events
server.on('close', () => {
    console.log('Server closed');

})
const localhost = '127.0.0.1';
const port = 8881;
server.listen(port, localhost, () => {
    console.log(`Server has been started on ${localhost}:
${port}\nWaiting for requests...`);
})
```

```
There was a new sale!
Costumer name: Jonas
There are now 9 items left in stock.
Server has been started on 127.0.0.1:8881
```