Module 3
Events & Streams

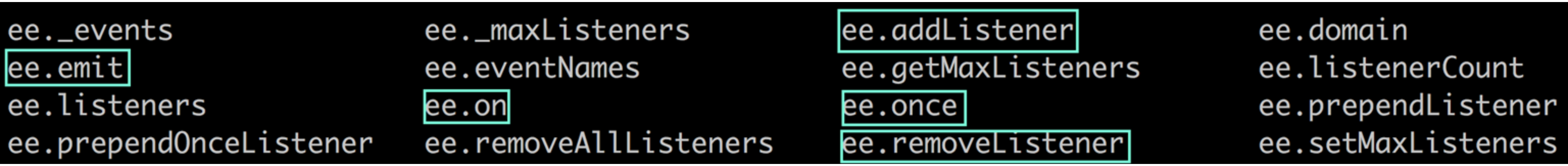
Outline

- Node’s EventEmitter class
- Async programming with Events
- Callbacks vs Events
- Understanding streams
- Types of streams
- Piping between streams

Events - Official Site Says

- Much of the Node.js core API is built around an idiomatic **asynchronous event-driven architecture** in which certain kinds of objects (called "**emitters**") periodically emit named events that cause Function objects ("**listeners**") to be called.
- All objects that emit events are instances of the **EventEmitter** class

EventEmitter - Core Object for events



```
ee._events                 ee._maxListeners           ee.addListener             ee.domain
ee.emit                    ee.eventNames              ee.getMaxListeners         ee.listenerCount
ee.listeners               ee.on                      ee.once                    ee.prependListener
ee.prependOnceListener     ee.removeAllListeners      ee.removeListener          ee.setMaxListeners
```

EventEmitter

- The publisher uses **event.emit(type,[args])**
- The subscriber uses **event.on(type, handler)**

Events vs callbacks

- The publisher uses **event.emit(type,[args])**
- The subscriber uses **event.on(type, handler)**

Stream - Official Site Says

- A **stream** is an abstract interface for working with streaming data in Node.js
- Simply put continuous flow of data from source to destination , like unix pipes
- Stream is an **EventEmitter** with some specials methods

Types of Stream

Readable
Writable
Duplex
Transform

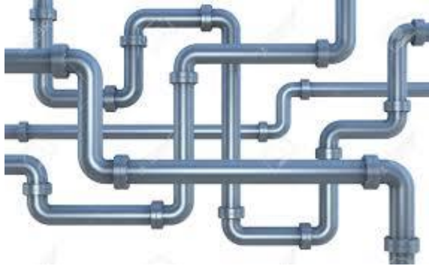
Readable Stream

- Inherits from **require(‘stream’).Stream**
- **Property**: readable (bool)
- **Events**: ‘data’, ‘end’, ‘close’, ‘error’
- **Methods**: pause(), resume(), end(), destroy()

Writable Stream

- Inherits from EventEmitter
- **Property:** writable (boolean)
- **Events:** ‘drain’, ‘error’, ‘close’,
- **Methods:** write(), end(), destroy()

piping



- An Input stream can be piped to output stream
- Pipes can be chained
- Handles back pressure automatically

src => pipe => dest

```
readableStream.pipe(writableStream);
```

Summary

- Understanding Events
- EventEmitter class
- Understanding Streams
- Reading and writing streams
- Using pipe()

Check your knowledge

- How many types of streams?
- Stream is a ____________ with special methods
- Src -> _________ -> dest
- On vs Once
- How do you unregister from an event?
- Function to raise an event?
- All objects that emit events are instances of the _________ class