Node Examples

Example: Read contents of a File

Blocking

```
Read file from Filesystem, set equal to “contents”
Print contents
Do something else
```

Non Blocking

```
Read file from Filesystem
  whenever you’re complete, print the contents
Do Something else
```

callback

Blocking vs Non-Blocking

Blocking

```
const contents = fs.readFileSync('/etc/hosts');
console.log(contents);
console.log('Doing something else');
```

Non Blocking

```
fs.readFile('/etc/hosts', function(err, contents) {
  console.log(contents);
});
console.log('Doing something else');
```

callback

Callback Alternate Syntax

```
fs.readFile('/etc/hosts', function(err, contents) {
  console.log(contents);
});
console.log('Doing something else');
```

equivalent

```
const callback = function(err, contents) {
  console.log(contents);
};
fs.readFile('/etc/hosts', callback);
console.log('Doing something else');
```

Callback Arrow Notation

```
fs.readFile('/etc/hosts', function(err, contents) {
  console.log(contents);
});
console.log('Doing something else');
```

equivalent

```
fs.readFile('/etc/hosts', (err, contents) => {
  console.log(contents);
});
console.log('Doing something else');
```

Three Callback Styles

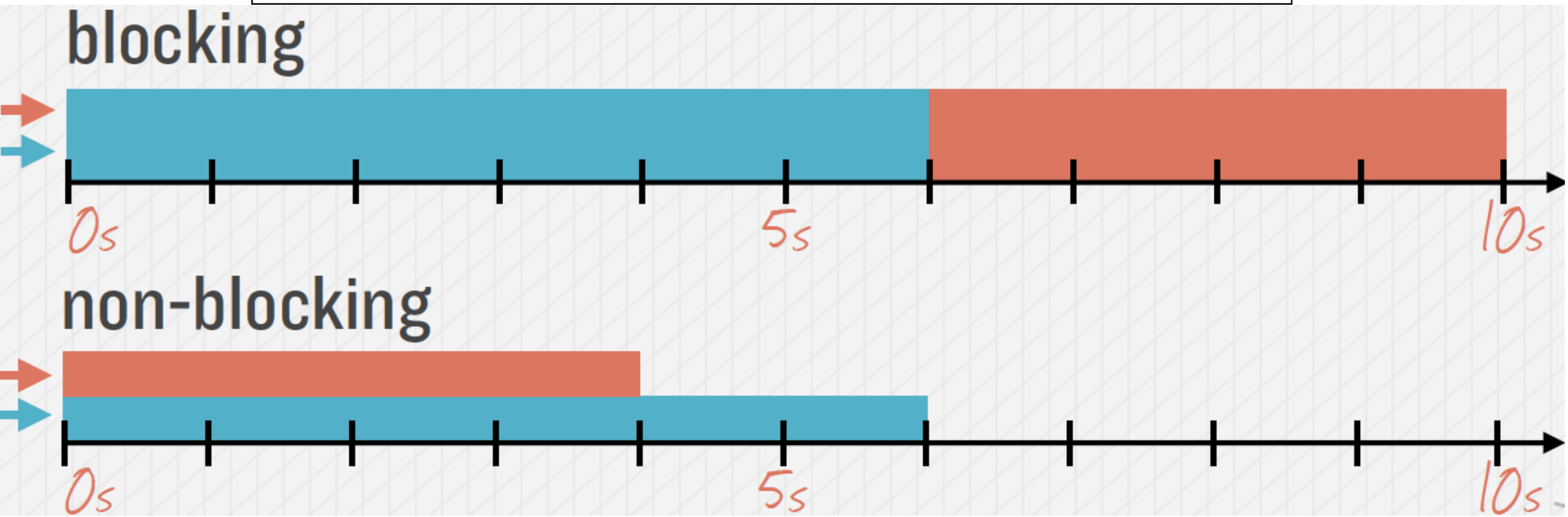
```
fs.readFile('/etc/hosts', function(err, contents) {
  console.log(contents);
});
console.log('Doing something else');
```

```
const callback = function(err, contents) {
  console.log(contents);
};
fs.readFile('/etc/hosts', callback);
console.log('Doing something else');
```

```
fs.readFile('/etc/hosts', (err, contents) => {
  console.log(contents);
});
console.log('Doing something else');
```

Blocking vs Non-blocking Performance

```
const contents1 = fs.readFileSync('/etc/hosts');
const contents2 = fs.readFileSync('/etc/inetcfg');
console.log(contents1);
console.log(contents2);
```




```
const callback = function(err, contents) {
  console.log(contents);
}
fs.readFile('/etc/hosts', callback);
fs.readFile('/etc/inetcfg', callback);
```

node.js Hello World

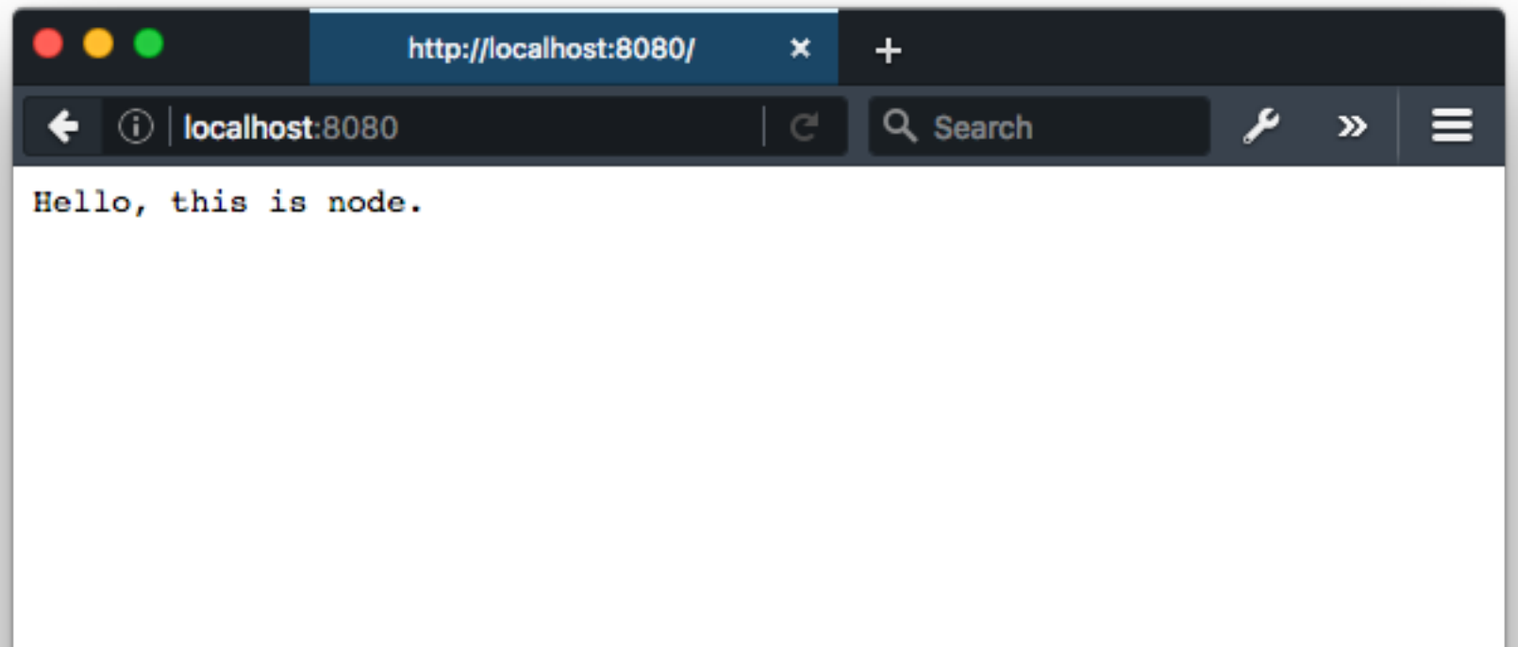
```
const http =   require('http');
http.createServer(function(request, response) {
  response.writeHead(200);
  response.write("Hello, this is node.");
  response.end();
}).listen(8080);
console.log('Listening on port 8080...');
```

```
$ node hello.js
```

```
Listening on port 8080...
```

```
$ curl http://localhost:8080
```

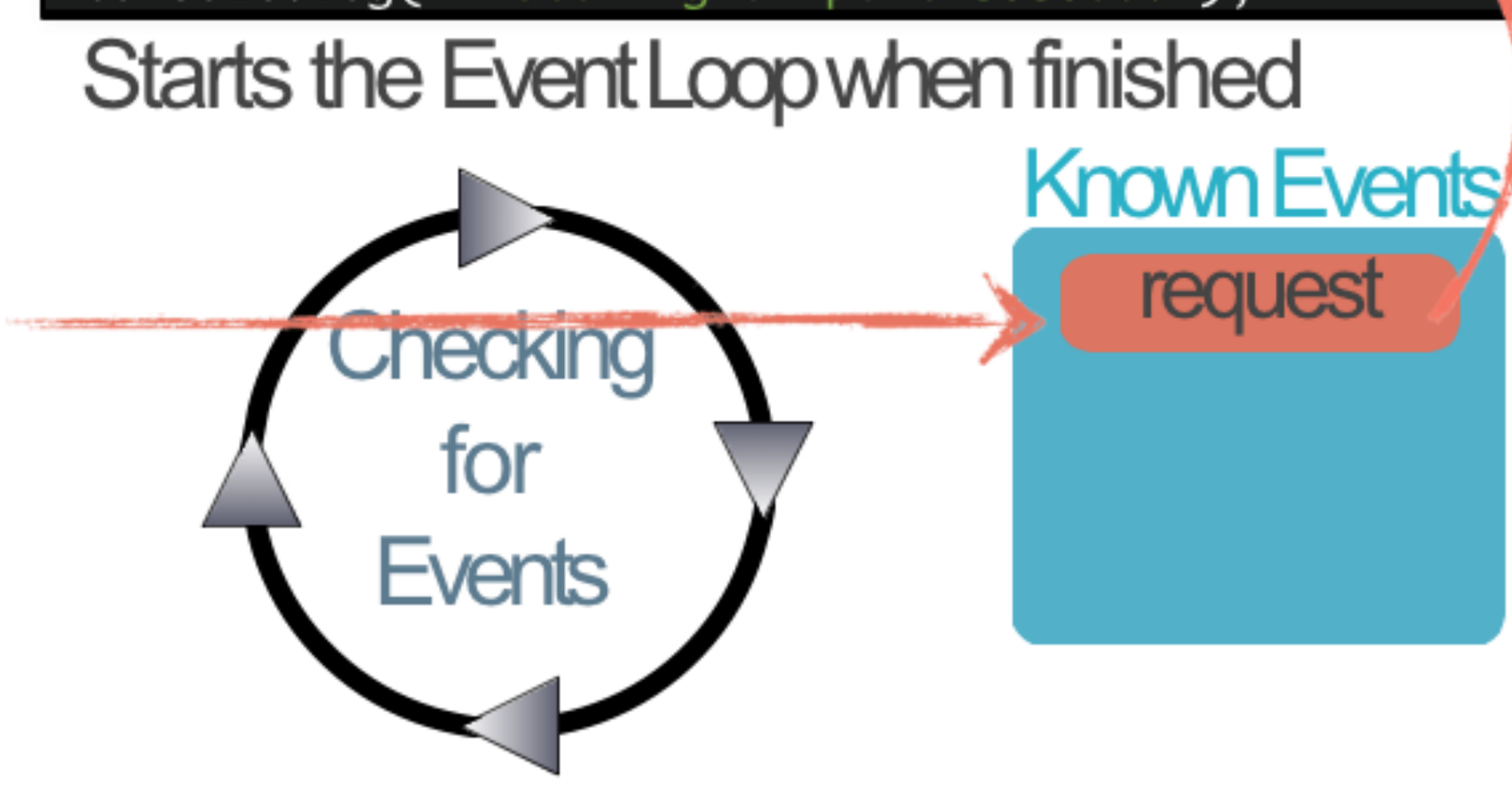
```
Hello, this is node.
```

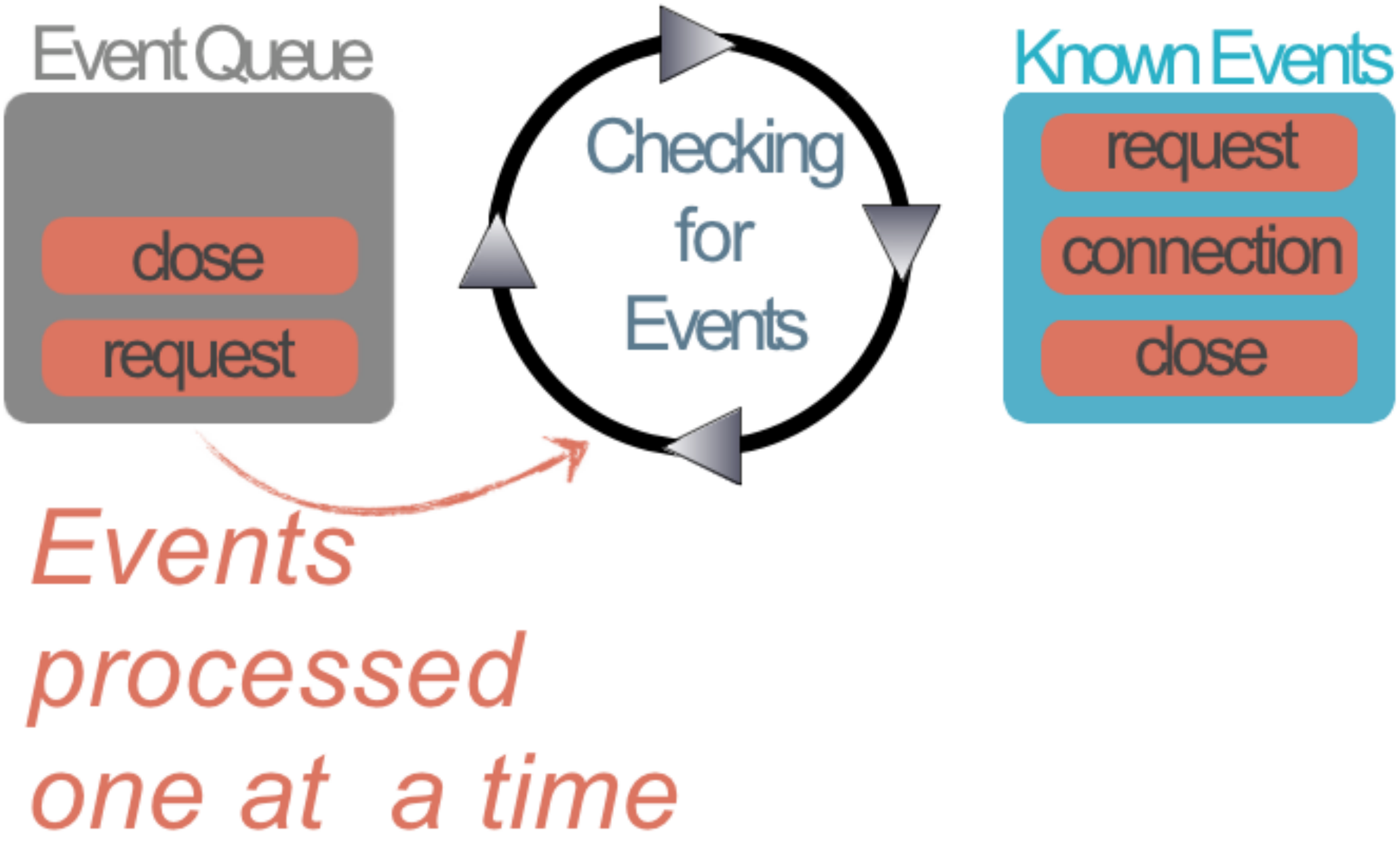


The Event Loop

```
var http = require('http');
http.createServer(function(request, response) {
  ...
}).listen(8080);
console.log('Listening on port 8080...');
```

Starts the Event Loop when finished




Event Queue
close
request
Checking for Events
Known Events
request
connection
close
Events processed one at a time

Typical Blocking Calls

- Calls out to web services
- Reads/Writes on the Database
- Calls to extensions

- Synchronous version of these types of calls must be avoided in node applications
- Instead, any activity likely to be blocked is to be called asynchronously
- Callbacks!