JÖNKÖPING UNIVERSITY

School of Engineering

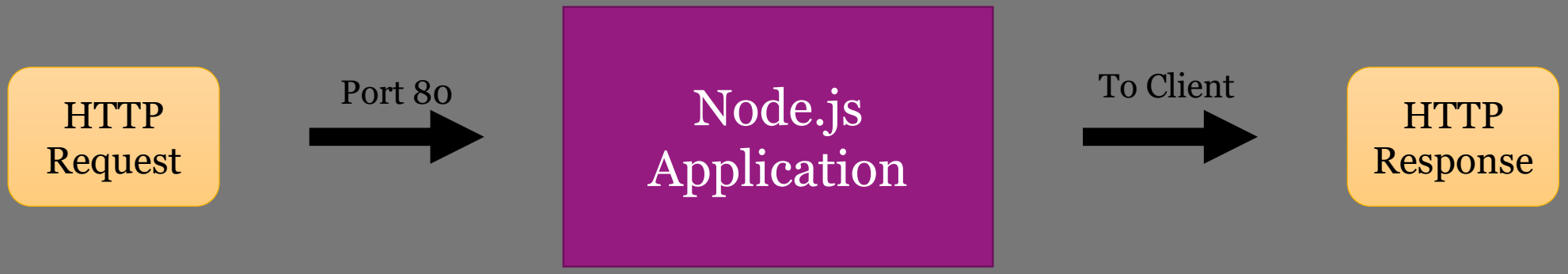
WEB APPLICATIONS IN NODE.JS

Peter Larsson-Green

Jönköping University

Autumn 2018

WEB APPLICATION IN NODE.JS



The `http` module takes care of the hard parts for us.

WEB APPLICATION IN NODE.JS

```
const http = require('http')

const myServer = http.createServer(function(request, response){
  // This callback is called each time
  // a new HTTP request is received.
  response.writeHead(200, {"Content-Type": "text/plain"})
  response.end("Here you go.")
})

myServer.listen(80)
```

Instance of the class `http.Server`

Headers

Status code

Body

Port number

INSPECTING INCOMING REQUESTS

```
GET /path/to/the-page.html HTTP/1.1
Host: website.com
Accept: text/html


the-body
```

```
const myServer = http.createServer(function(request, response){
  const method = request.method // "GET"
  const uri = request.url // "/path/to/the-page.html"
  const version = request.httpVersion // "1.1"
  const headers = request.headers // {host: "website.com", ...}
})
```

INSPECTING INCOMING REQUESTS

```
GET /path/to/the-page.html HTTP/1.1
Host: website.com
Accept: text/html


the-body
```

```
const myServer = http.createServer(function(request, response){
  const bodyParts = []
  request.on('data', function(chunk){
    bodyParts.push(chunk)
  })
  request.on('end', function(){
    const body = Buffer.concat(bodyParts).toString() // "the-body"
  })
})
```

CREATING OUTGOING RESPONSES

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 8

the-body
```

```
const myServer = http.createServer(function(request, response){
  response.statusCode = 200
  response.statusMessage = 'OK'
  response.setHeader("Content-Type", "text/html")
})
```

```
response.writeHead(200, {"Content-Type", "text/html"})
```

CREATING OUTGOING RESPONSES

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 8

the-body
```

```
const myServer = http.createServer(function(request, response){
  response.write("the")
  response.write("-body")
  response.end()
})
```

```
response.end("the-body")
```

PUTTING IT ALL TOGETHER

The server has some resources (data stored in files/database/code).

1. Inspect the incoming HTTP request:
   - URI – Which resource?
   - METHOD – What to do?
2. Do as told.
3. Send back appropriate HTTP response.

EXAMPLE

A web application in Node.js.