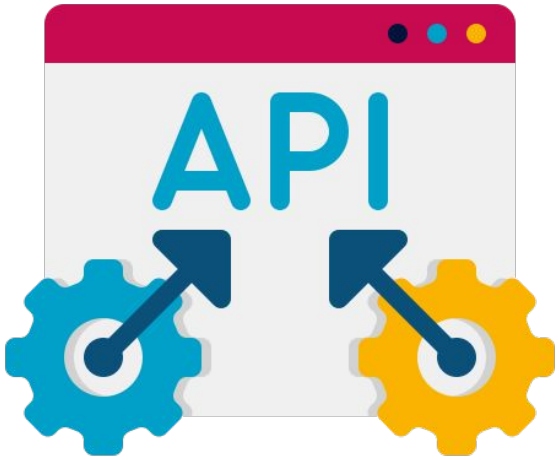


Node.js
Request and Response Objects

SENG 4640
Software Engineering for Web Apps
Winter 2023

Sina Keshvadi
Thompson Rivers University

Review

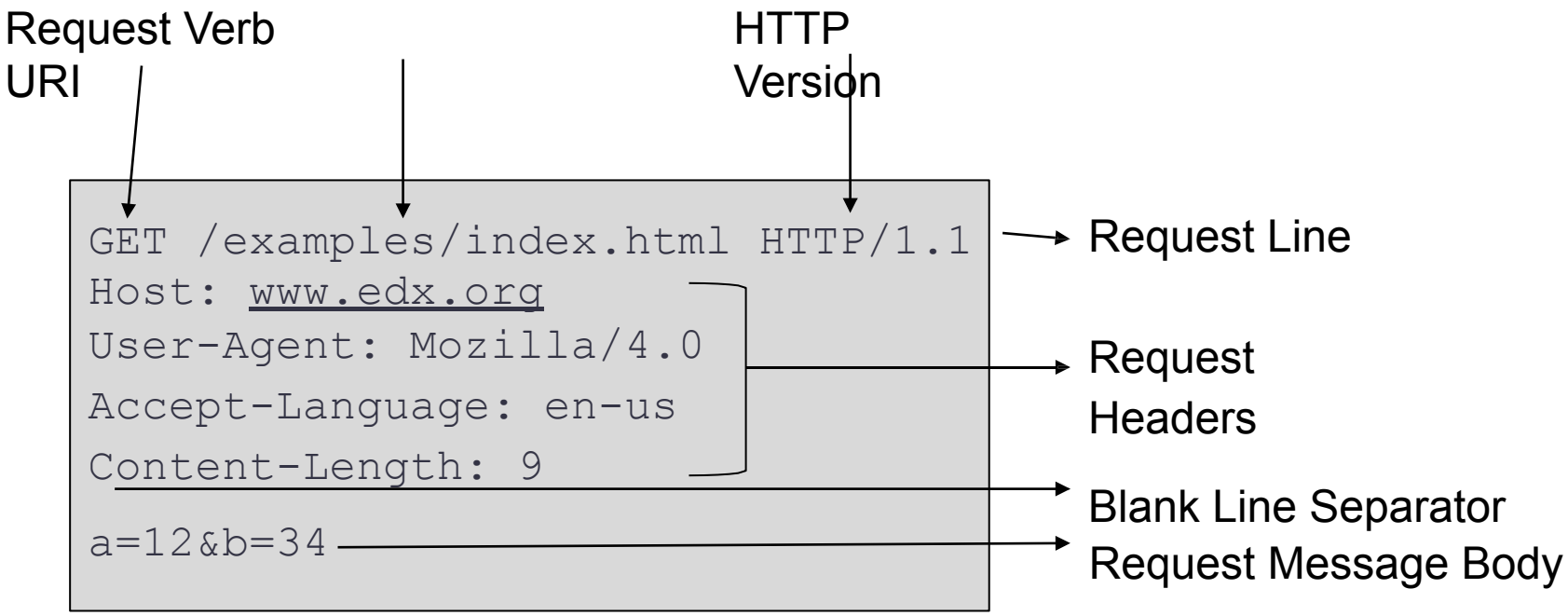
- Web browsers communicate with Web servers via HTTP requests and responses

- Node.js and Express simplify the development of Web servers to handle HTTP requests and create and return HTTP responses

Commands

```
> mkdir app_one

> cd app_one

> npm init

> npm install express --save

> touch index.js
make a file - index.js
write the backend content
> node index.js

open http://localhost:3000/
```

Anatomy of an HTTP Request



Node.js/Express Request Objects

- An HTTP Request is represented as an object in the Express app
- The object is passed as a parameter to the callback function/event handler

```
var express = require('express');
var app = express();

app.use('/', (req, res) => {
  res.send('Hello World!');
});
```

Write this code in index.js file, compile it with *node index.js*

Node.js/Express Request Objects

- An HTTP Request is represented as an object in the Express app
- The object is passed as a parameter to the callback function/event handler

```
var express = require('express');
var app = express();

app.use('/', (req, res) => {
  res.send('Hello World!');
});
```

Node.js/Express Request Objects

- An HTTP Request is represented as an object in the Express app
- The object is passed as a parameter to the callback function/event handler

```
var express = require('express');
var app = express();

app.use('/', (req, res) => {
  res.send('Hello World!');
});
```

Node.js/Express Request Objects

- An HTTP Request is represented as an object in the Express app
- The object is passed as a parameter to the callback function/event handler

```
var express = require('express');
var app = express();

app.use('/', (req, res) => {
  res.send('Hello World!');
});
```

Request Object Properties/Functions

```
app.use('/', (req, res) => {

  var method = req.method;
  var url = req.url;
  var agent = req.headers['user-agent'];
  agent = req.get('User-Agent');
```

Request Object Properties/Functions

```
app.use('/', (req, res) => {

  var method = req.method;
  var url = req.url;
  var agent = req.headers['user-agent'];
  agent = req.get('User-Agent');
```

Request Object Properties/Functions

- **`method`**: the HTTP Request verb/action

```
app.use('/', (req, res) => {

  var method = req.method;
  var url = req.url;
  var agent = req.headers['user-agent'];
  agent = req.get('User-Agent');
```

Request Object Properties/Functions

- **`method`**: the HTTP Request verb/action
- **`url`**: the resource that was requested

```
app.use('/', (req, res) => {

  var method = req.method;
  var url = req.url;
  var agent = req.headers['user-agent'];
  agent = req.get('User-Agent');
```

Request Object Properties/Functions

- **`method`**: the HTTP Request verb/action
- **`url`**: the resource that was requested
- **`headers`**: object containing all headers

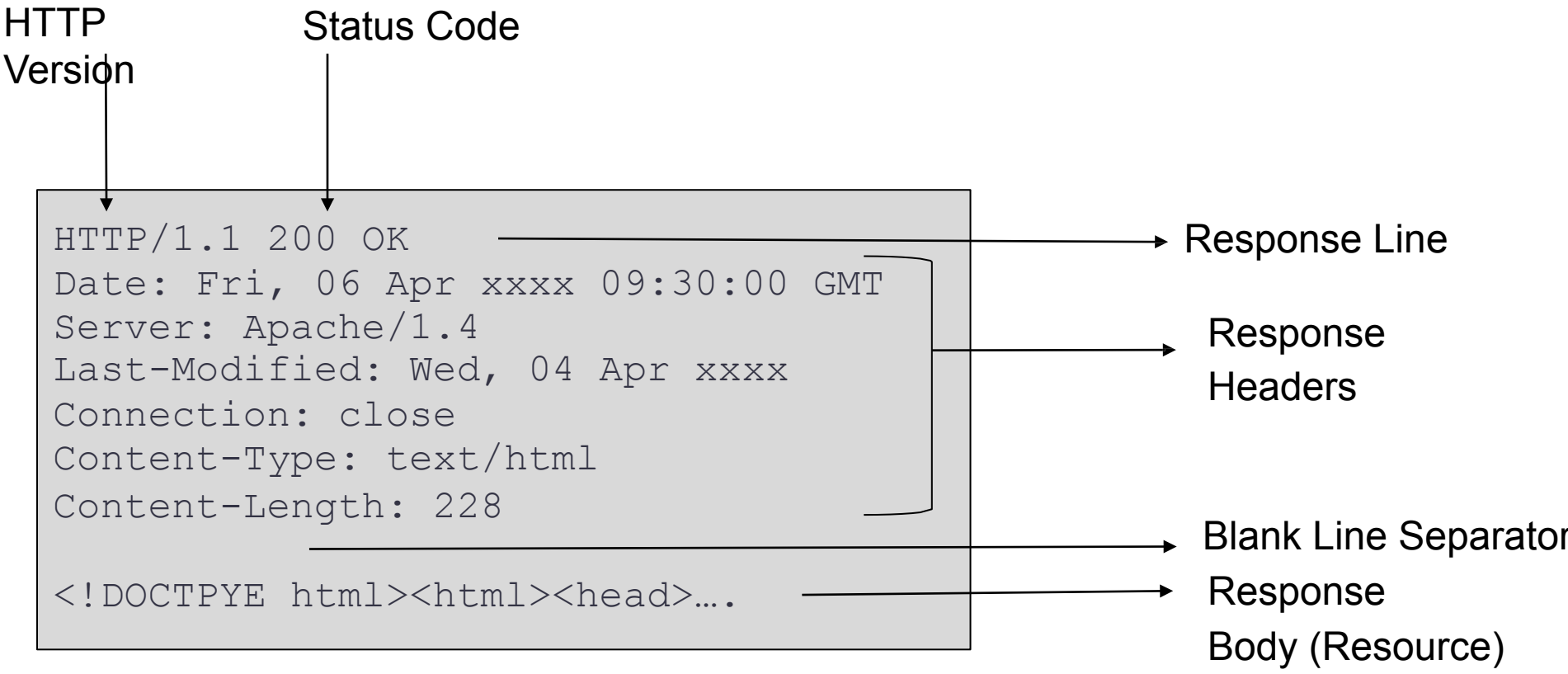
```
app.use('/', (req, res) => {

  var method = req.method;
  var url = req.url;
  var agent = req.headers['user-agent'];
  agent = req.get('User-Agent');
```

Request Object Properties/Functions

- **`method`**: the HTTP Request verb/action
- **`url`**: the resource that was requested
- **`headers`**: object containing all headers
- **`get(field)`**:request header field

```
app.use('/', (req, res) => {

  var method = req.method;
  var url = req.url;
  var agent = req.headers['user-agent'];
  agent = req.get('User-Agent');
```

Anatomy of an HTTP Response



Node.js/Express Response Objects

- An HTTP Response is also represented as an object in the Express app
- The object is passed as a parameter to the callback function/event handler

```
var express = require('express');
var app = express();

app.use('/', (req, res) => {
  res.send('Hello World!');
});
```

Node.js/Express Response Objects

- An HTTP Response is also represented as an object in the Express app
- The object is passed as a parameter to the callback function/event handler

```
var express = require('express');
var app = express();

app.use('/', (req, res) => {
  res.send('Hello World!');
});
```

Response Object Functions

```
app.use('/', (req, res) => {

  res.status(200);
  res.type('html');
  res.write('Hello world!');
  res.write('<p>');
  res.write('<b>Have a nice day</b>');
  res.end();
```

Response Object Functions

```
app.use('/', (req, res) => {

  res.status(200);
  res.type('html');
  res.write('Hello world!');
  res.write('<p>');
  res.write('<b>Have a nice day</b>');
  res.end();
```

Response Object Functions

- **`status`**: set the HTTP status code

```
app.use('/', (req, res) => {

  res.status(200);
  res.type('html');
  res.write('Hello world!');
  res.write('<p>');
  res.write('<b>Have a nice day</b>');
  res.end();
```

Response Object Functions

- **`status`**: set the HTTP status code
- **`type`**: set the HTTP content type

```
app.use('/', (req, res) => {

  res.status(200);
  res.type('html');
  res.write('Hello world!');
  res.write('<p>');
  res.write('<b>Have a nice day</b>');
  res.end();
```

Response Object Functions

- **`status`**: set the HTTP status code
- **`type`**: set the HTTP content type
- **`write`**: add content to the body of the response

```
app.use('/', (req, res) => {

  res.status(200);
  res.type('html');
  res.write('Hello world!');
  res.write('<p>');
  res.write('<b>Have a nice day</b>');
  res.end();
```

Response Object Functions

- **`status`**: set the HTTP status code
- **`type`**: set the HTTP content type
- **`write`**: add content to the body of the response

```
app.use('/', (req, res) => {

  res.status(200);
  res.type('html');
  res.write('Hello world!');
  res.write('<p>');
  res.write('<b>Have a nice day</b>');
  res.end();
```

Response Object Functions

- **`status`**: set the HTTP status code
- **`type`**: set the HTTP content type
- **`write`**: add content to the body of the response

```
app.use('/', (req, res) => {

  res.status(200);
  res.type('html');
  res.write('Hello world!');
  res.write('<p>');
  res.write('<b>Have a nice day</b>');
  res.end();
```

Response Object Functions

- **`status`**: set the HTTP status code
- **`type`**: set the HTTP content type
- **`write`**: add content to the body of the response
- **`end`**: send the response and close the connection

```
app.use('/', (req, res) => {

  res.status(200);
  res.type('html');
  res.write('Hello world!');
  res.write('<p>');
  res.write('<b>Have a nice day</b>');
  res.end();
```

Note that all codes in these examples are in the server-side.

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh

  res.status(200).type('html');

  if (name) {
   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Combining Requests and Responses

```
app.use('/', (req, res) => {

  var name = req.query.name; // e.g. /?name=devesh
  res.status(200).type('html');

  if (name) {

   res.write('Hi, ' + name + "it's nice to see you.");
  }
  else {
   res.write('Welcome, guest!');
  }

  res.end();
});
```

Now, try this on your browser:

http://localhost:3000/?name=Mina

Note. use
> node index.js

We're just starting to scratch the surface
of how we can develop a web application using Node
and Express.

Summary

- Node.js and Express represent HTTP requests and responses using JavaScript objects

- We can use these objects' properties and functions to dynamically generate the content that is sent in response to a request