

# back-end

## Server & Express

lab-2a

**Stand-up!**

*Show what  
you did*

# today

~~I. Stand up~~

II. Request & Response

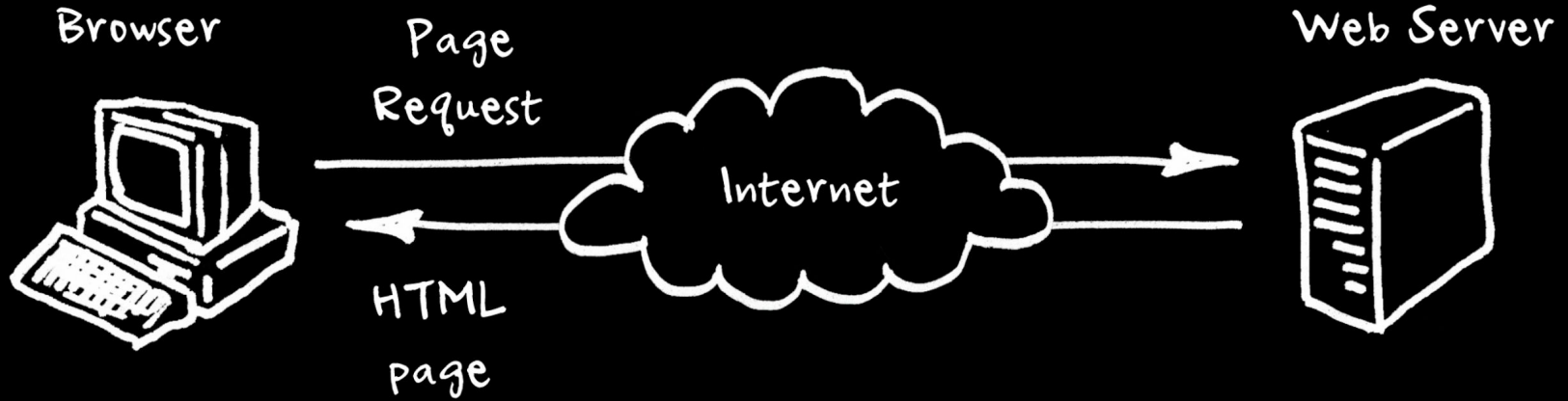
III. Express

IV. Routes & static

**Req & Res**

# Servers

flow



# Servers

client/server

## **Client** (request)

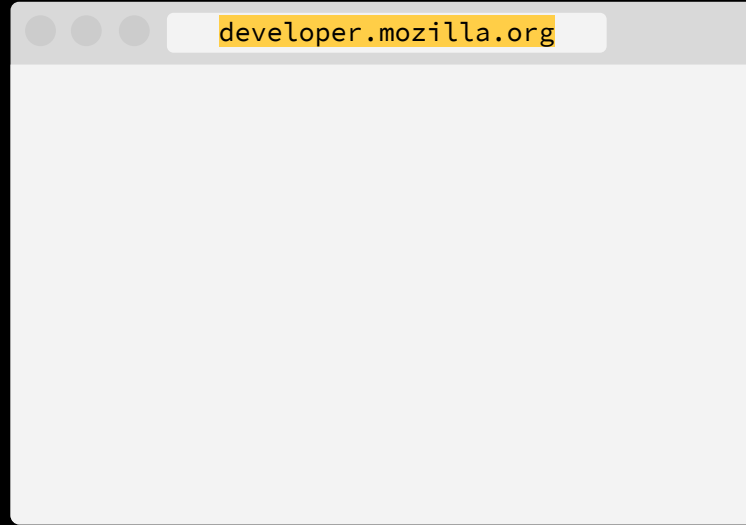
- ❖ Also known as user-agent: acts for the user
- ❖ Often a web browser
- ❖ Client asks for something: the request

## **Server** (response)

- ❖ Connected machine waiting for requests
- ❖ Can be one machine, multiple machines, or multiple servers per machine
- ❖ Server sends something back: the response

# Servers

request url



# Servers

response





# Servers

headers

HTTP/1.1 200 OK

Date: Mon, 19 Feb 2018 15:40:02 GMT

Last-Modified: Tue, 13 Feb 2018 20:18:22 GMT

Content-Length: 29769

Content-Type: text/html



<!DOCTYPE html... (here comes the 29769 bytes of the requested web page)

# Servers

response

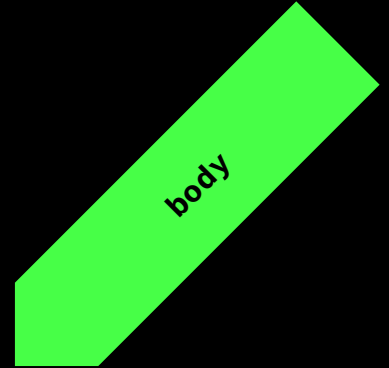
HTTP/1.1 200 OK

Date: Mon, 19 Feb 2018 15:40:02 GMT

Last-Modified: Tue, 13 Feb 2018 20:18:22 GMT

Content-Length: 29769

Content-Type: text/html



<!DOCTYPE html... (here comes the 29769 bytes of the requested web page)

index.js

```
var http = require('http')

http.createServer(onrequest).listen(8000)

function onrequest(req, res) {
  res.statusCode = 200
  res.setHeader('Content-Type', 'text/html')
  res.end('Hello World!\n')
}
```

Handle each request by  
sending "Hello World"

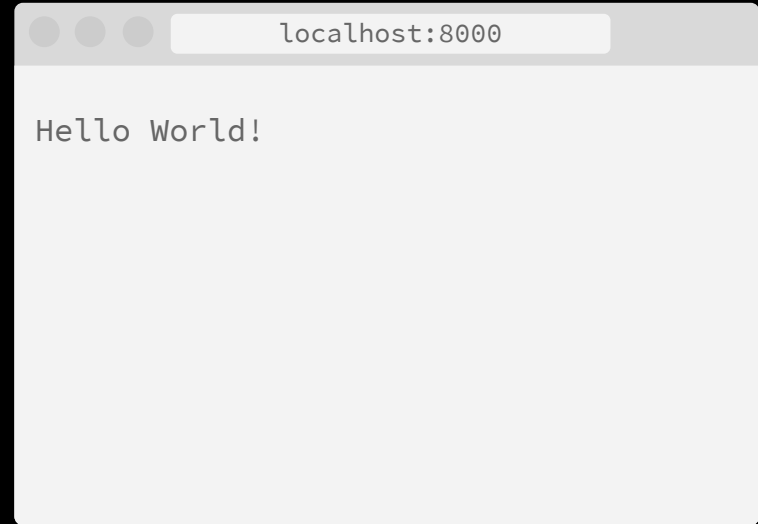


A terminal window with a dark blue background. At the top left are three colored window control buttons (red, yellow, green). The title bar shows the text "bash". The prompt "\$" is followed by the command "node index.js" in a yellow monospace font.

```
bash
$ node index.js
```



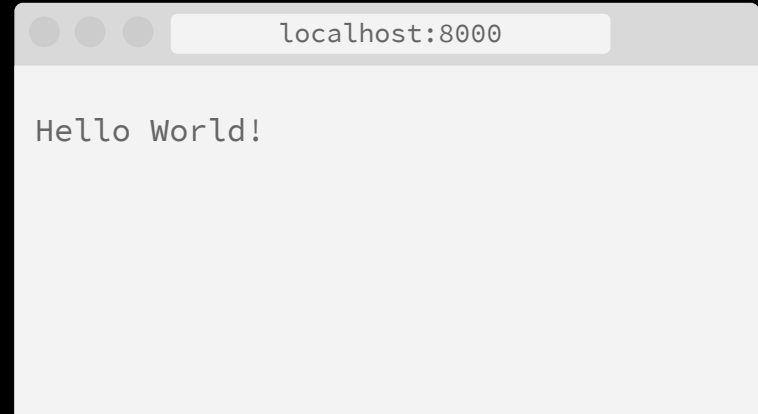
Now we have a server!





```
bash
$ node index.js
```

Now we have a server!



**Note:** this is a pretty simple server that actually misses a lot of features: routes, static files, mime types, streaming etc.

The background is a solid black field. It is populated with numerous small, light green geometric shapes and wavy lines. These elements are scattered across the entire frame, creating a textured, abstract pattern. The shapes include triangles, squares, and circles, some of which are solid, while others are outlines. The wavy lines are thin and fluid, resembling stylized waves or calligraphic strokes. The overall effect is a dynamic and modern aesthetic.

# Express

expressjs.com


Express

Home Getting started Guide API reference Advanced topics Resources

# Express 4.16.1

Fast, unopinionated,  
minimalist web  
framework for **Node.js**

```
$ npm install express --save
```

 **Express 4.16.0 contains important security updates.**  
For more information on what was added in this release, see the [4.16.0 changelog](#).

## Web Applications

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

## APIs


With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.

## Performance

Express provides a thin layer of fundamental web application features, without obscuring Node.js features that you know and love.

## Frameworks

Many [popular frameworks](#) are based on Express.



[expressjs.com](https://expressjs.com)

?

# Express

?

- ❖ **Web Applications:** Express is a minimal and flexible Node web application framework that provides a robust set of features for web and mobile applications
- ❖ **APIs:** With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy

[expressjs.com](https://expressjs.com)



index.js

```
const express = require('express')

express()
  .get('/', onhome)
  .listen(1900)

function onhome(req, res) {
  res.send('<h1>Hello Client</h1>\n')
}
```

Express

index.js

```
const express = require('express')
```

Require **express** package

```
express()
```

```
  .get('/', onhome)
```

```
  .listen(1900)
```

```
function onhome(req, res) {
```

```
  res.send('<h1>Hello Client</h1>\n')
```

```
}
```

Express

index.js

```
const express = require('express')

express()
  .get('/', onhome)
  .listen(1900)

function onhome(req, res) {
  res.send('<h1>Hello Client</h1>\n')
}
```

Handle each request by  
**sending "Hello World"**

Express

# express

middleware

Middleware functions are functions that have **access to the request and the response object**. [...] Middleware can **make changes** to the request and response objects.

Express

[expressjs.com](https://expressjs.com)

index.js

```
const express = require('express')
const compression = require('compression')

express()
  .get('/', onhome)
  .listen(1900)
  .use(compression)

function onhome(req, res) {
  res.send('<h1>Hello Client</h1>\n')
}
```

For example **compression**  
is middleware

Express

# Routes & Static

# express

routing

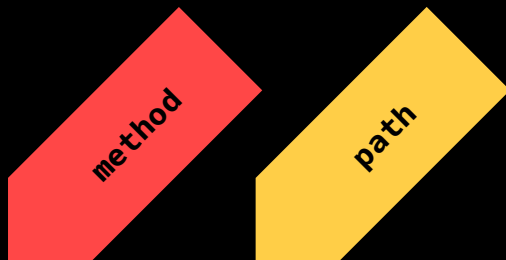
Routing refers to determining **how an application responds to a client request to a particular endpoint**, which is a URI (or path) and a specific HTTP request method (GET, POST).

Express

[expressjs.com](https://expressjs.com)

# express

methods



```
app.get('/', callback)
```

Express



index.js

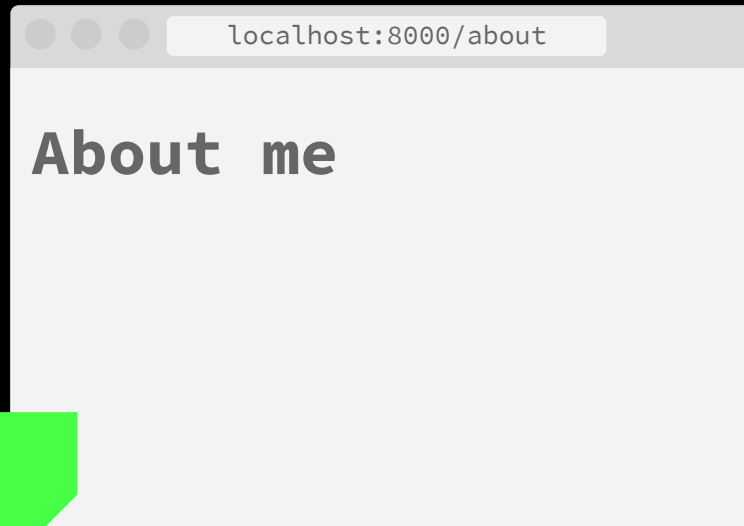
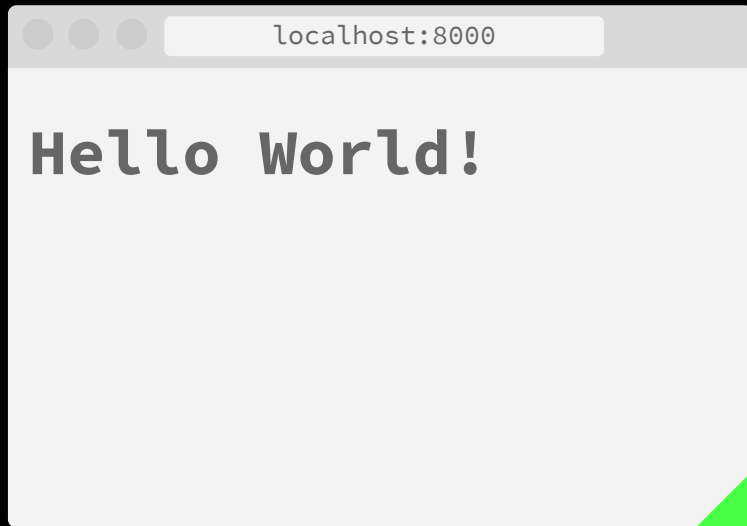
```
const express = require('express')

express()
  .get('/', onhome)
  .get('/about', onabout)
  .listen(1900)

function onhome(req, res) {
  res.send('<h1>Hello World!</h1>\n')
}

function onabout(req, res) {
  res.send('<h1>About me</h1>\n')
}
```

Express



We have a **route!**

# express

static

Static files are files that are **not dynamically generated**. For example; .css, images, fonts. Express provides static middleware to look up files relative to the static directory.

Express

[expressjs.com](https://expressjs.com)

# express

## folders

```
// Files
plain-server/
├─ index.js
├─ static/
│   └─ css
│       └─ style.css
│   └─ images
│       └─ kitten.jpg
```

# express

## folders

```
// Files
plain-server/
├─ index.js
├─ static/
│   └─ css
│       └─ style.css
│   └─ images
│       └─ kitten.jpg
```

**Note:** everything in the **static** folder will be public!

index.js

```
const express = require('express')

express()
  .use('/static', express.static('static'))
  .get('/', onhome)
  .get('/about', onabout)

  .listen(1900)

function onhome(req, res) {
  res.send('<h1>Hello World!</h1>\n')
}

function onabout(req, res) {
  res.send('<h1>About me</h1>\n')
}
```

Use **static** middleware

Express

localhost:8000

# Hello World!



404  
Not Found

CSS and  
Kittens!

~/Desktop

→ npm init -y

Wrote to /Users/deckard/Desktop/package.json:

```
{
  "name": "Desktop",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
  },
  "keywords": [],
  "author": "",
```

Live demo **express**



## Serve



# serve

In this assignment you'll build a static file server with a little help from Express.

### Synopsis

- **Time:** 4:00h
- **Goals:** subgoal 3, subgoal 4
- **Due:** before week 3

### Description

Create a server that handles routes and serves static files in Node.js. Use the `express` web framework. Make sure it does (atleast) the following three things:

1. **Basic routing:** Have a couple of different `routes` (e.g. `/about` `/login` ) that are useful for your matching-application.
2. **Error handling:** Respond with a `404 Not Found` if you go to a route that doesn't exist.
3. **Serve static files :** such CSS but also media files such as images, video's or audio files.

# work on **serve**



# exit;

see you in lab-2b!