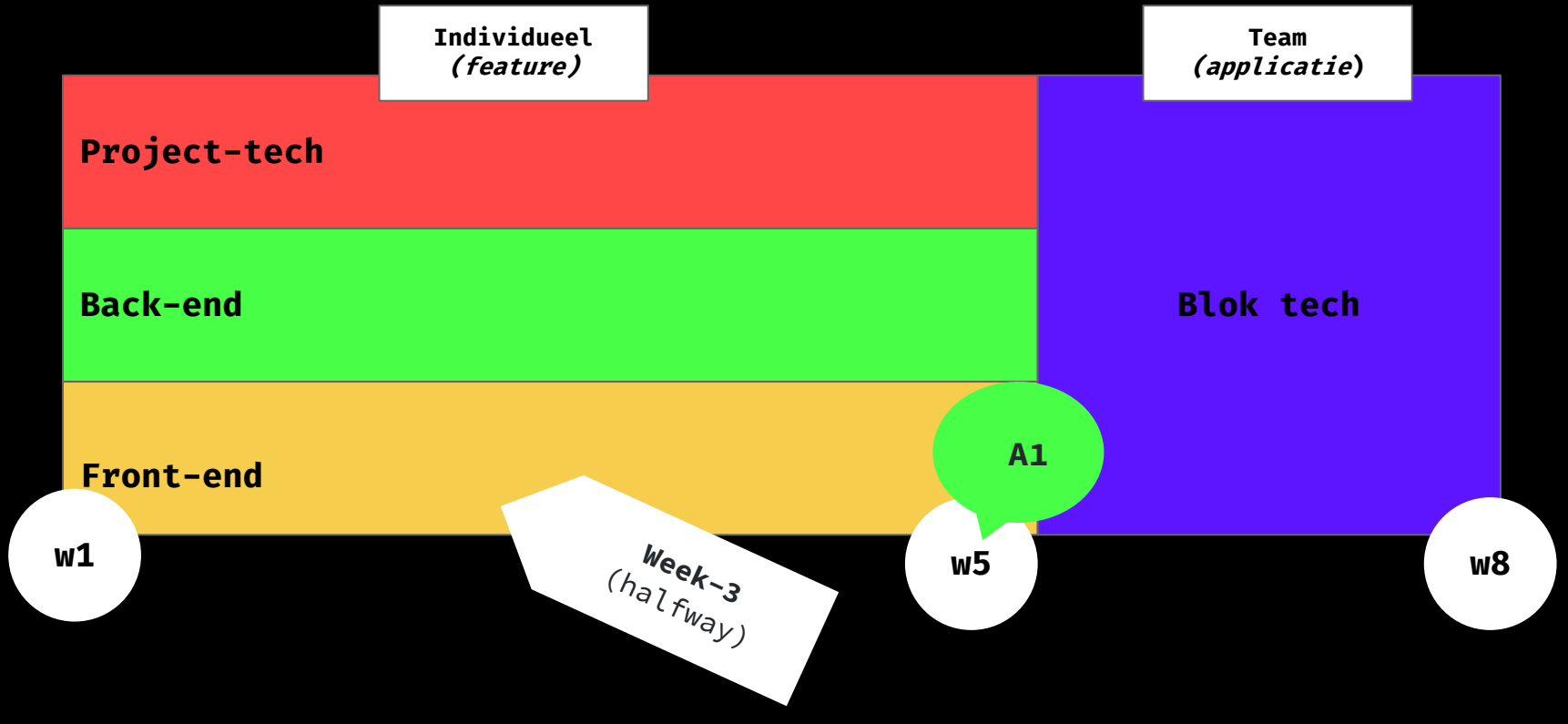
back-end

Send data to server
Connect database

lab-3

Stand-up!

Show what you did


Individueel
(feature)
Team
(applicatie)
Project-tech
Back-end
Front-end
Blok tech
A1
w1
Week-3
(halfway)
w5
w8

today

I. ~~Stand-up~~
II. Send data to server
    I. HTTP
    II. Forms
    III. Files
III. Connect to database

HTTP

http

?

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems. […]

Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text.

wikipedia.org

http

url

A Uniform Resource Locator (URL) […] is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.

wikipedia.org

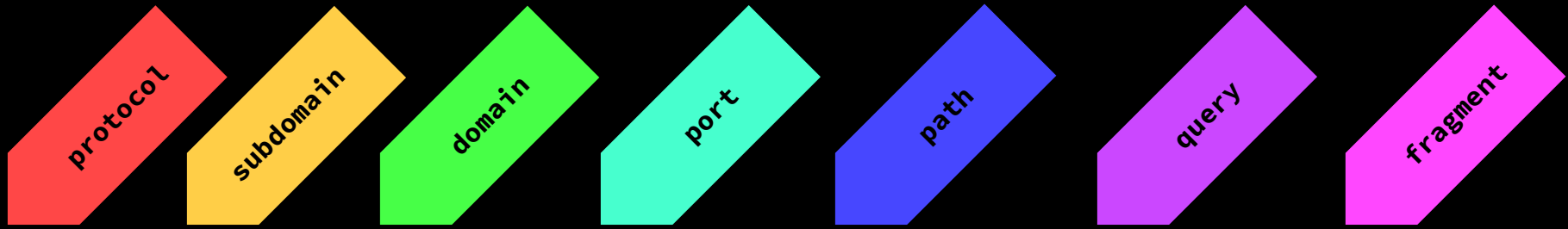
http

url

```
http://test.example.com:3000/users/search?q=test&w=all#results
```

http

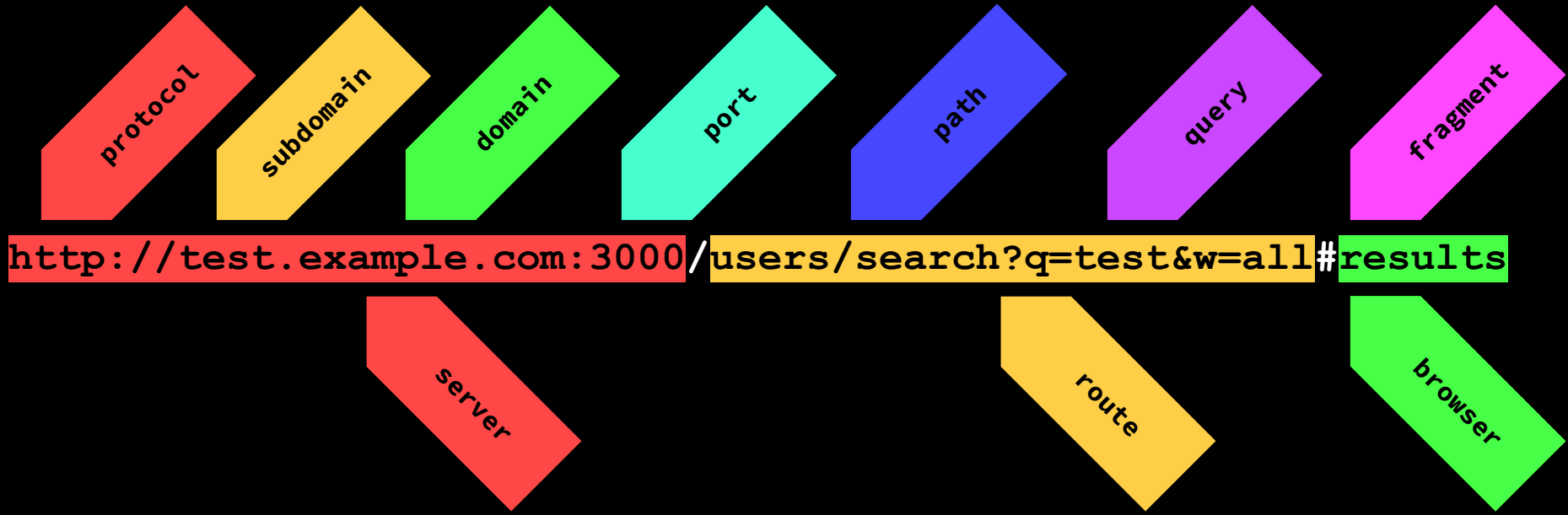
url



http://test.example.com:3000/users/search?q=test&w=all#results

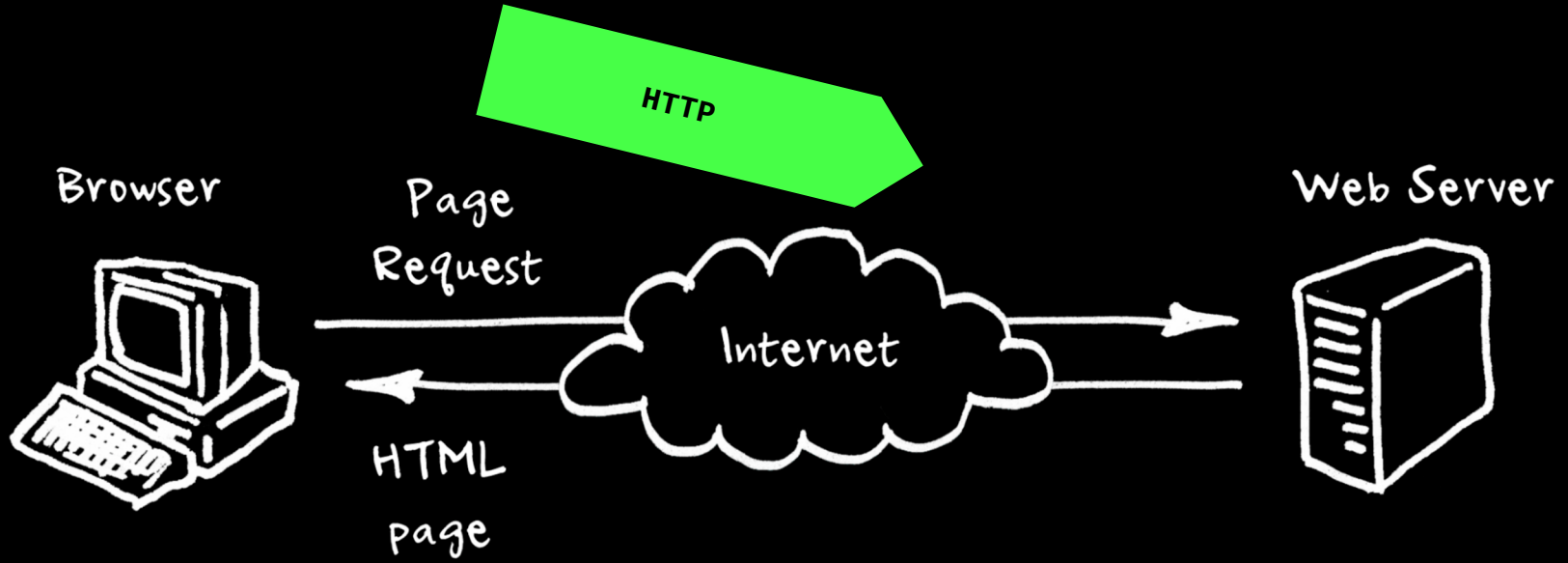
http

url


protocol
subdomain
domain
port
path
query
fragment
http://test.example.com:3000/users/search?q=test&w=all#results
server
route
browser

http

req/res



http

response

HTTP/1.1 200 OK

status code & status message

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)

http

status

Category	Range	Example
Information	**1••**	101 Switching Protocols
Success	**2••**	200 OK, 201 Created
Redirect	**3••**	301 Moved Permanently
Client error	**4••**	400 Bad Request, 404 Not Found
Server Error	**5••**	500 Internal Server Error

http

methods

- **Create:** PUT, POST
- **Read:** GET
- **Update:** PATCH
- **Delete:** DELETE

http

get

```
> GET /users/1 HTTP/1.1
> Host: example.com
>
< HTTP/1.1 200 OK
<
< {"id":1,"name":"Anna","age":22}
```

Request a resource

http

post

```
> POST /users HTTP/1.1
> Host: example.com
>
> {"name":"Bisma","age":19}
>
< HTTP/1.1 201 Created
< Location: /users/2
<
< {"id":2,"name":"Bisma","age":19}
```

Submit a resource

http

get/post

http

post

- **Text** text/plain, text/html, text/javascript, …
- **Image** image/jpeg, image/png, …
- **Audio** audio/webm, audio/ogg, …
- **Video** video/webm, video/ogg, …
- **Application** application/pdf, application/octet-stream,…

Forms


localhost:8000/add
Add a new movie
Title
Plot
Description
add
Add a movie
Express

view/add.ejs

```
<% include head.ejs %>
<title>Add a movie - My movie website</title>
<h1>Add a new movie</h1>
<form action=/ method=post>
  <label>Title <input name=tit
  <label>
    Plot (short)
    <input name=plot>
  </label>
  <label>
    Description (long)
    <textarea
      name=description
      rows=5
    ></textarea>
  </label>
  <button>Add</button>
</form>
```

Send a **POST** request…

Express

view/add.ejs

```
<% include head.ejs %>
<title>Add a movie - My movie website</title>
<h1>Add a new movie</h1>
<form action=/ method=post>
  <label>Title <input name=tit
  <label>
    Plot (short)
    <input name=plot>
  </label>
  <label>
    Description (long)
    <textarea
      name=description
      rows=5
    ></textarea>
  </label>
  <button>Add</button>
</form>
```

…to /, and…

Express

view/add.ejs

```
<% include head.ejs %>
<title>Add a movie - My movie website</title>
<h1>Add a new movie</h1>
<form action=/ method=post>
  <label>Title <input name=title></label>
  <label>
    Plot (short)
    <input name=plot>
  </label>
  <label>
    Description (long)
    <textarea
      name=description
      rows=5
    ></textarea>
  </label>
  <button>Add</button>
</form>
```

…send **title**, **plot**, and **description**

Express

form

slug

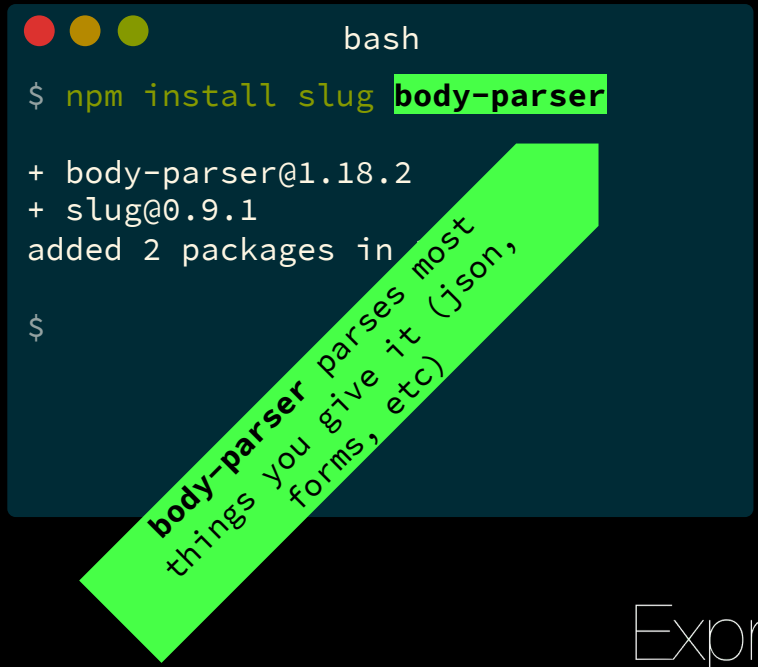
```
// Files
express-server/
├─ node_modules/
├─ static/
│  └─ index.css
├─ view/
│  ├─ add.ejs
│  ├─ detail.ejs
│  ├─ head.ejs
│  ├─ list.ejs
│  └─ not-found.ejs
├─ index.js
└─ package.json
```

Express

form

body

```
// Files
express-server/
├─ node_modules/
├─ static/
│  └─ index.css
├─ view/
│  ├─ add.ejs
│  ├─ detail.ejs
│  ├─ head.ejs
│  ├─ list.ejs
│  └─ not-found.ejs
├─ index.js
└─ package.json
```



Express

index.js

```
var express = require('express')
var find = require('array-find
var slug = require('slug')
var bodyParser = require('       ser')

…

express()
  .use(express.s          tatic'))
  .use(bodyPar          ncoded({extended: true}))
  .set('view          , 'ejs')
  .set('views          iew')
  .get('/', movies)
  .post('/', add)
  .get('/add', form)
  .get('/:id', movie)
  .use(notFound)
  .listen(8000)

…
```

Handle a **post** request to /

Express

index.js

```
…

function form(req, res) {
  res.render('add.ejs')
}

function add(req, res) {
  var id = slug(req.body.title).toLowerCase()

  data.push({
    id: id,
    title: req.body.title,
    plot: req.body.plot,
    description: req.body.description
  })

  res.redirect('/' + id)
}

…
```

body-parser parses the data and stores it in **req.body**

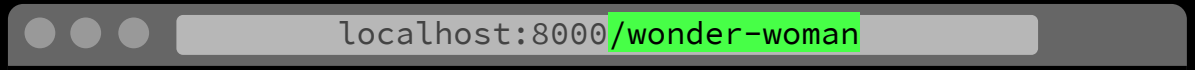
Express

index.js

```
…

function form(req, res) {
  res.render('add.ejs')
}

function add(req, res) {
  var id = slug(req.body.title).toLowerCase()

  data.push({
    id: id,
    title: req.body.title,
    plot: req.body.plot,
    description: req.body.description
  })

  res.redirect('/' + id)
}

…
```

title, **plot**, and **description** come from **name** attributes on inputs

Express

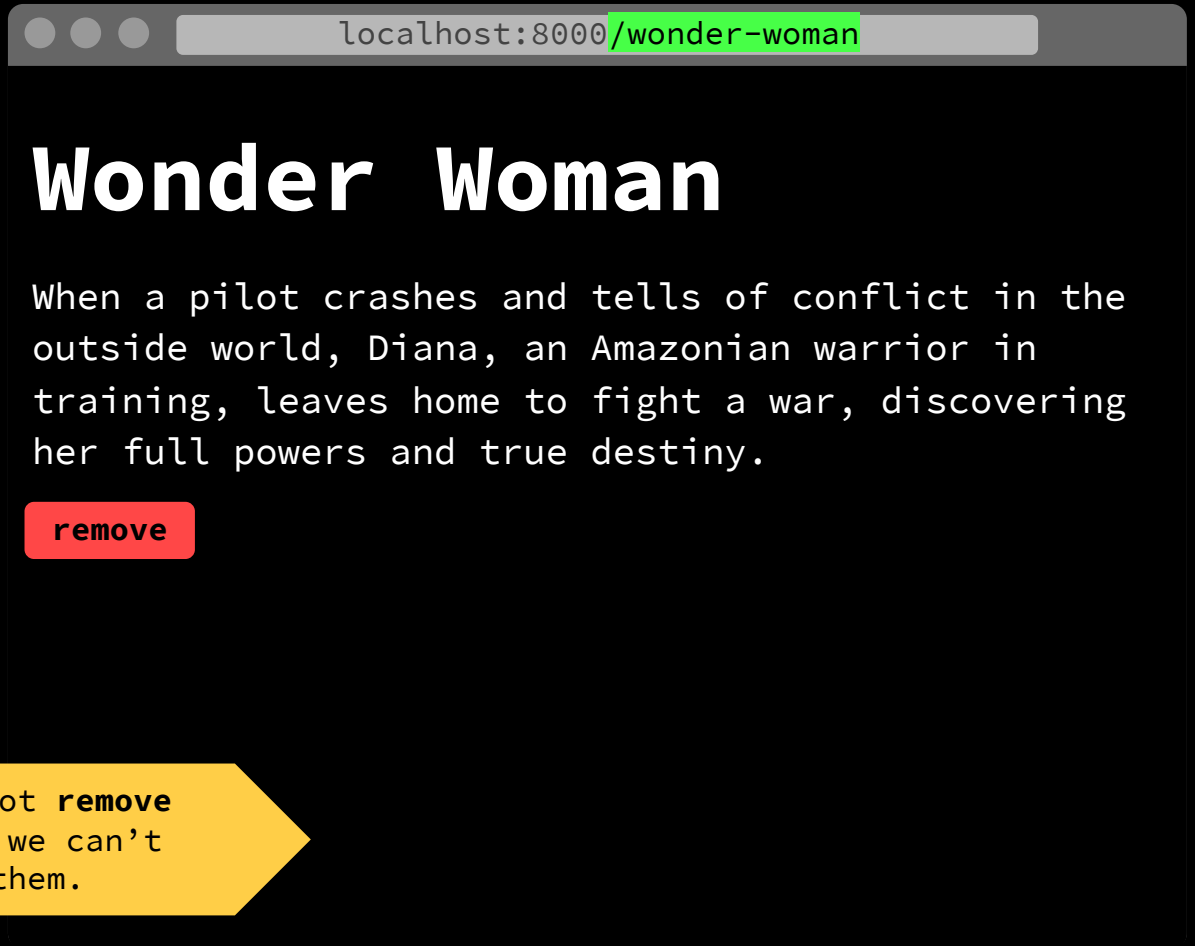


Wonder Woman

When a pilot crashes and tells of conflict in the outside world, Diana, an Amazonian warrior in training, leaves home to fight a war, discovering her full powers and true destiny.

…and we can submit the form!

Express



…but we cannot **remove** movies? And we can't **update** them.

Express

Input

input

> Receive input from users on the server and manipulate that data for your own feature using HTTP request methods.

Synopsis

- **Time**: 10:00h
- **Goals**: subgoal 4, subgoal 5, subgoal 6
- **Due**: before week 4

Description

So far we only send data (response) to the client with our server. A one-sided conversation. Now the fun starts, it's time to actually start receiving data from users. For example; users can enter something into an input field or submit whole forms with file uploads.

The description of this assignment is quite vague since the end result will be very specific to your Job Story. Make sure you at least spend the

work on **input**

Input

input

> Receive input from users on the server and manipulate that data for your own feature using HTTP request methods.

Synopsis

- **Time**: 10:00h
- **Goals**: subgoal 4, subgoal 5, subgoal 6
- **Due**: before week 4

Description

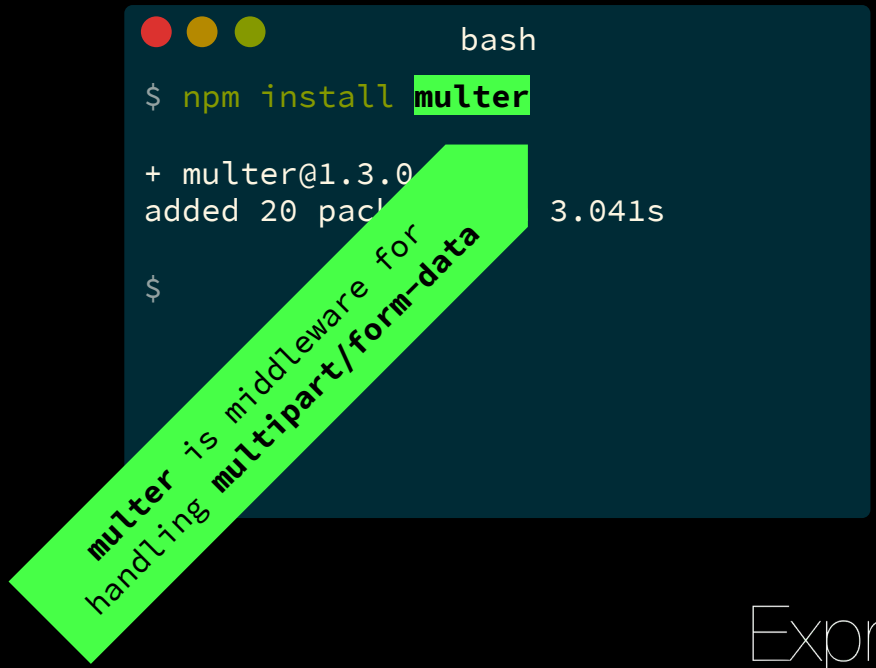
Note: *Input* is quite a **‘large’** and **‘vague’** assignment since the end result will be very specific to your Job Story.

Files

files

multer

```
// Files
express-server/
├─ node_modules/
├─ static/
│  ├─ index.css
│  ├─ index.js
│  └─ upload/
├─ view/
│  ├─ add.ejs
│  ├─ detail.ejs
│  ├─ head.ejs
│  ├─ list.ejs
│  ├─ not-found.ejs
│  └─ tail.ejs
├─ index.js
└─ package.json
```



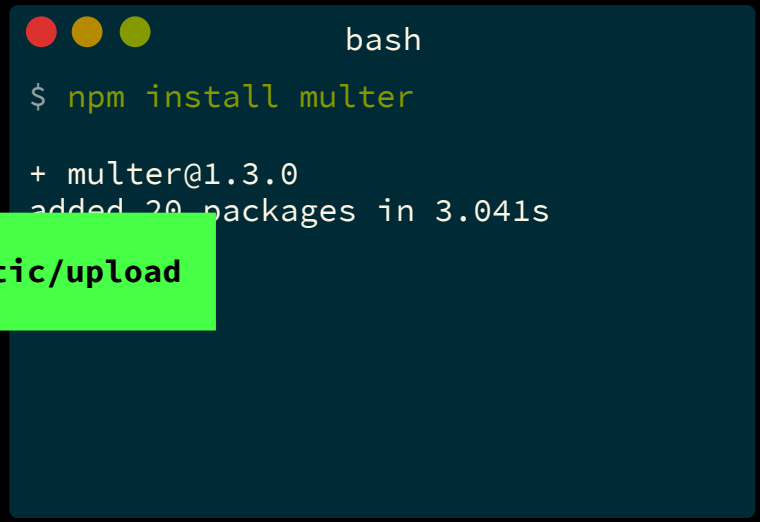
Express

files

folder

```
// Files
express-server/
├─ node_modules/
├─ static/
│  ├─ index.css
│  ├─ index.js
│  └─ upload/
├─ view/
│  ├─ add.ejs
│  ├─ detail.ejs
│  ├─ head.ejs
│  ├─ list.ejs
│  ├─ not-found.ejs
│  └─ tail.ejs
├─ index.js
└─ package.json
```

We'll upload files to **static/upload**



Express

static/index.css

```
…

img {
  max-width: 100%;
  max-height: 15em
}
```

view/detail.ejs

```
…
<h1><%= data.title %></h1>
<% if (data.cover) { %>
<img
  alt
  src="/upload/<%= data.cover %>"
>
<% } %>
<p><%= data.description %></p>
…
```

Express

view/add.ejs

```
<% include head.ejs %>
<title>Add a movie - My movie website</title>
<h1>Add a new movie</h1>
<form
  action=/
  method=post
  enctype=multipart/form-data
>
  <label>Title <input name=title></label>
  <label>
    Cover
    <input name=cover type=file accept=image/*>
  </label>
  <label>
    Plot (short)
    <input name=plot>
  </label>
  …
  <button>Add</button>
</form>
<% include tail.ejs %>
```

Needed for **input[type=file]**

Express

view/add.ejs

```
<% include head.ejs %>
<title>Add a movie - My movie website</title>
<h1>Add a new movie</h1>
<form
  action=/
  method=post
  enctype=multipart/form-data
>
  <label>Title <input name=title></label>
  <label>
    Cover
    <input name=cover type=file accept=image/*>
  </label>
  <label>
    Plot (short)
    <input name=plot>
  </label>
  …
  <button>Add</button>
</form>
<% include tail.ejs %>
```

Accept only images

Express

view/add.ejs

```
<% include head.ejs %>
<title>Add a movie - My movie website</title>
<h1>Add a new movie</h1>
<form
  action=/
  method=post
  enctype=multipart/form-data
>
  <label>Title <input name=title></label>
  <label>
    Cover
    <input name=cover type=file accept=image/*>
  </label>
  <label>
    Plot (short)
    <input name=plot>
  </label>
  …
  <button>Add</button>
</form>
<% include tail.ejs %>
```

Accept only images

Express

index.ejs

```
…
var multer = require('multer')

…

var upload = multer({dest: 'static/upload/'})

express()
  …
  .post('/', upload.single('cover'), add)
  …

function add(req, res) {
  …
  data.push({
    …
    cover: req.file ? req.file
    …
  })
}

…
```

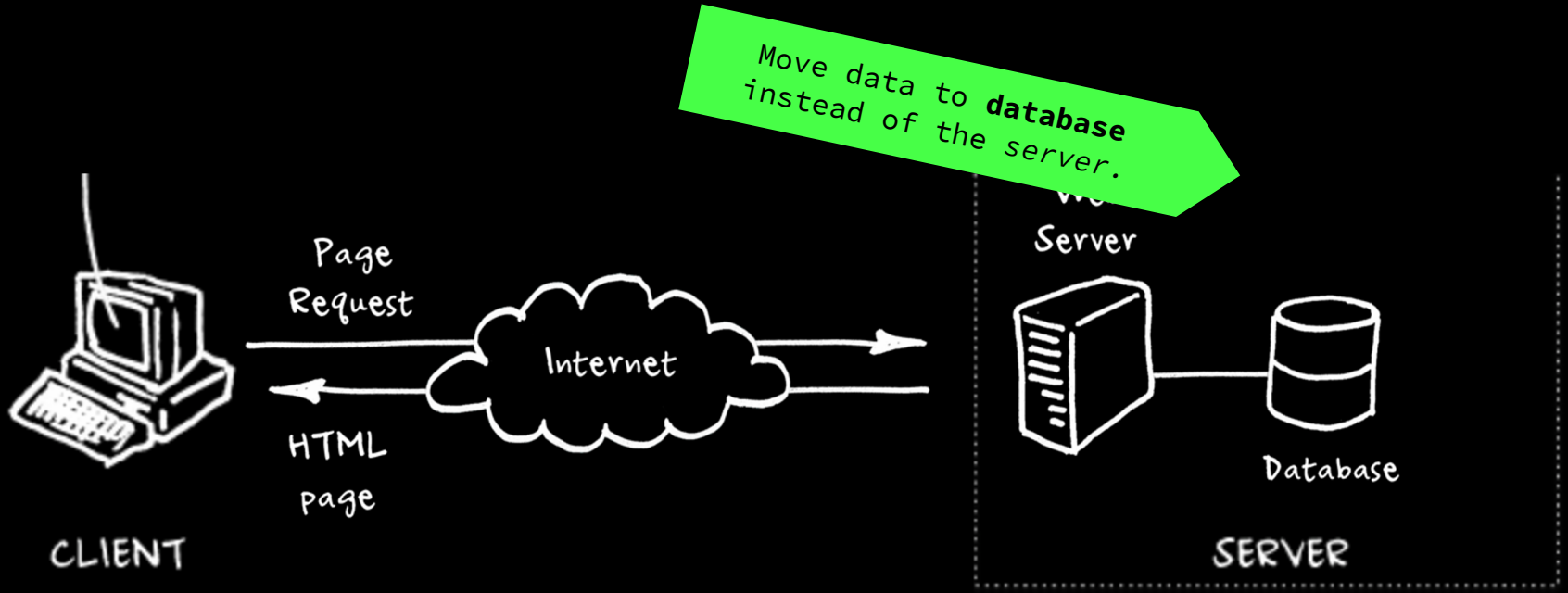
multer sets **req.file**

Express


localhost:8000/add
Add a new movie
Wonder Woman
Diana, an Amazonian warrior…
Cover
wonder-woman.jpg
When a pilot crashes and tells of conflict in the outside world, Diana, an Amazonian warrior in training, leaves home …
We can add files!

Express

Connect


Move data to **database** instead of the server.
Page Request
HTML page
Internet
Server
Database
CLIENT
SERVER

connect

mongodb

MongoDB (from hu**mong**ous) is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas. MongoDB is developed by MongoDB Inc. […]

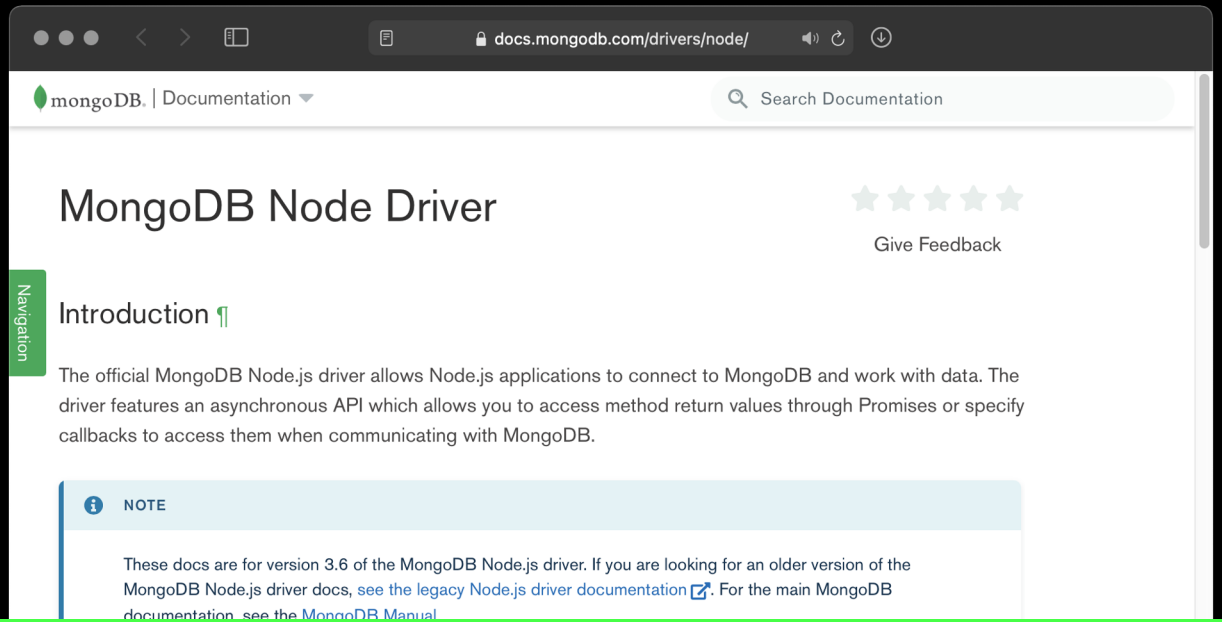
wikipedia.org

connect

mongodb

- JavaScript can be used in queries, aggregation function
- Map-reduce can be used for batch processing of data and aggregation operations
- Manage massive increases in new, rapidly changing data types

https://www.mongodb.com/compare/mongodb-mysql

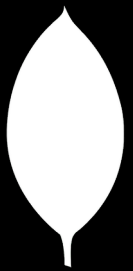


Note: there are a lot of small steps involved. Read the Mongo guides very carefully. If you miss a step everything will be broken.

bash

```
$ npm install mongodb

+ mongodb@2.2.33
+ dotenv@4.0.0
added 11 packages in 4.022s

$
```

mongodb wraps **MongoDB** for **Node**



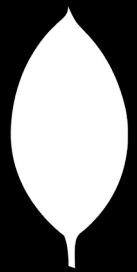
connect

mongodb

```
// Files
mongodb-server/
├─ node_modules/
├─ static/
│  ├─ index.css
│  ├─ index.js
│  └─ upload/
├─ view/
│  ├─ add.ejs
│  ├─ detail.ejs
│  ├─ head.ejs
│  ├─ list.ejs
│  ├─ not-found.ejs
│  └─ tail.ejs
├─ .env
├─ index.js
└─ package.json
```

.env

```
DB_HOST=localhost
DB_PORT=27017
DB_NAME=mymoviewebsite
DB_USERNAME=dandevri
```



connect

mongodb

```
// Files
mongodb-server/
├─ node_modules/
├─ static/
│  ├─ index.css
│  ├─ index.js
│  └─ upload/
├─ view/
│  ├─ add.ejs
│  ├─ detail.ejs
│  ├─ head.ejs
│  ├─ list.ejs
```

.env

```
DB_HOST=localhost
DB_PORT=27017
DB_NAME=mymoviewebsite
```

Note: Never ever put your **host and password in code or on GitHub**! People will be able to access your database!

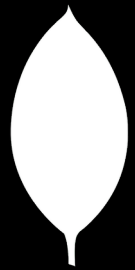
connect

mongodb

```
// Files
mongodb-server/
├─ node_modules/
├─ static/
│  ├─ index.css
│  ├─ index.js
│  └─ upload/
├─ view/
│  ├─ add.ejs
│  ├─ detail.ejs
│  ├─ head.ejs
│  ├─ list.ejs
│  ├─ not-found.ejs
│  └─ tail.ejs
├─ .env
├─ .gitignore
├─ index.js
└─ package.json
```

.gitignore

```
node_modules/
.DS_Store
.env
```

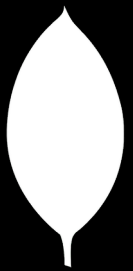


II MONG

MONGODB

index.js

```
…
var multer = require('multer')
var mongo = require('mongodb')

require('dotenv').config()

var db = null
var url = 'mongodb://' + process.env.DB_HOST + ':' +
process.env.DB_PORT

mongo.MongoClient.connect(url, function (err, client) {
  if (err) throw err
  db = client.db(process.env.DB_NAME)
})

…
```



connect

> Create a database and set-up your remote connection

Synopsis

- **Time**: 2:00h
- **Goals**: subgoal 7, subgoal 8
- **Due**: before week 4

Description

We'll cover actually storing the user input and learning about databases the next lesson but what you already do is create a MongoDB atlas account and set-up your mongodb connection.

All of your data is going to be stored in MongoDB. Take ± 2 hours to set-up your database and connect to it. There are roughly two options: go with a database as a service DBaaS or use MongoDB locally.

work on **connect**

connect

> Create a database and set-up your remote connection

Synopsis

- **Time**: 2:00h
- **Goals**: subgoal 7, subgoal 8
- **Due**: before week 4

Description

We'll cover actually storing the user input and learning about databases the next lesson but what you already do is create a MongoDB atlas

Note: If you are ready, **begin on week-4** because after the holiday we 'only' have one week left before A1.

exit;

see you in lab-4!