back-end

HTTP & Forms

lab-3a

Stand-up!

Show what you did

today

1. ~~Stand-up~~
2. HTTP
3. Forms

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

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

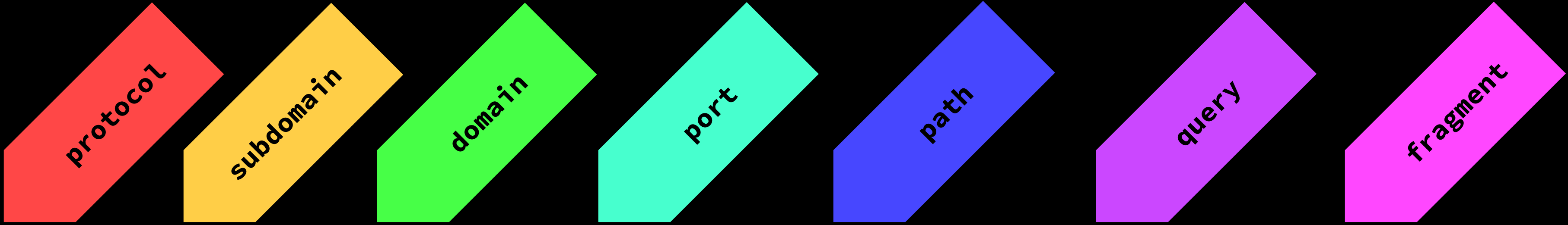
http

url

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

http

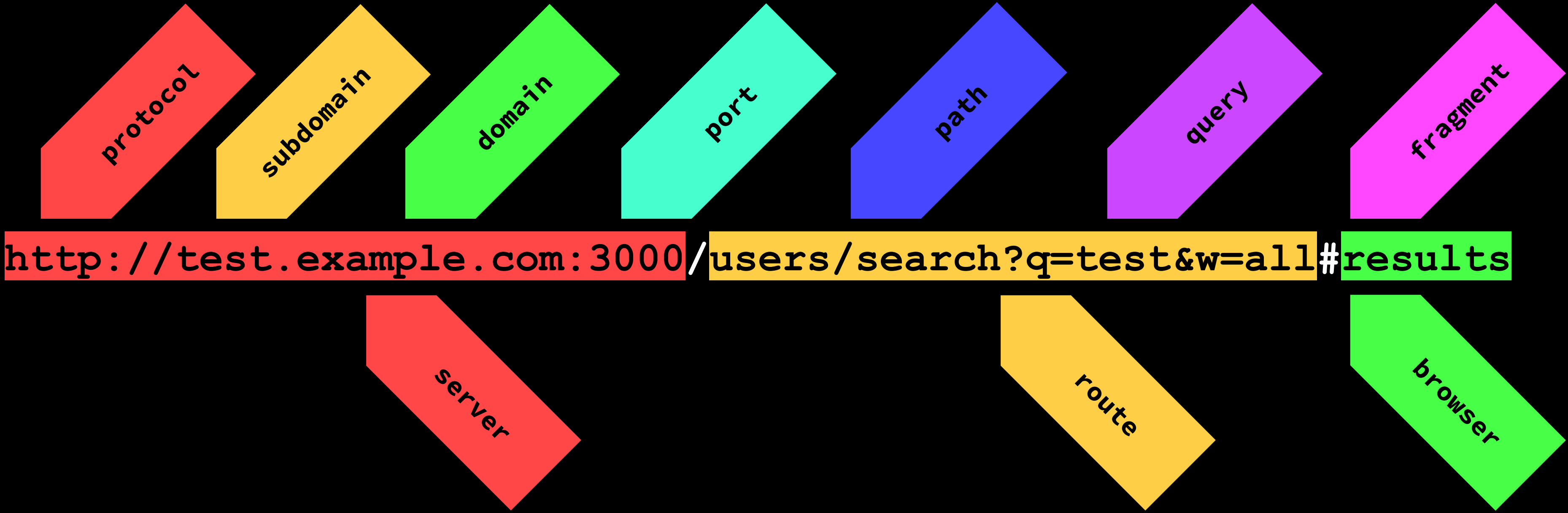
url



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

http

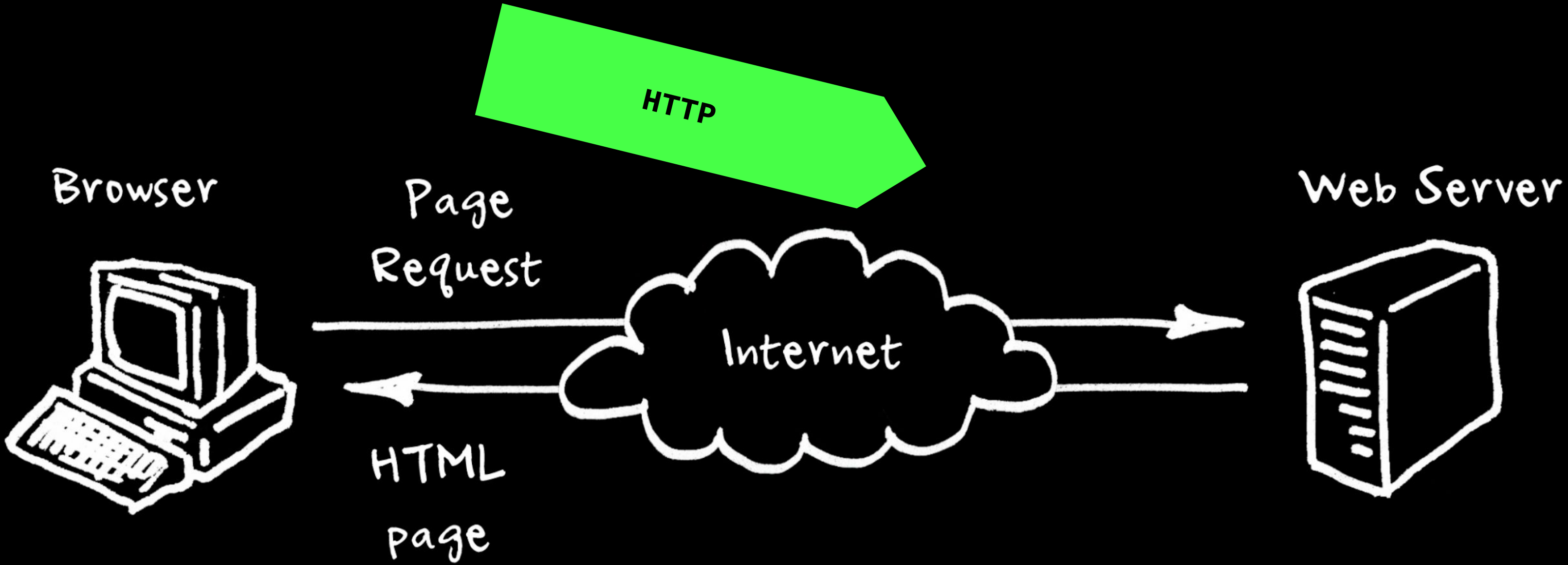
url



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
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)
```

status code & status message

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

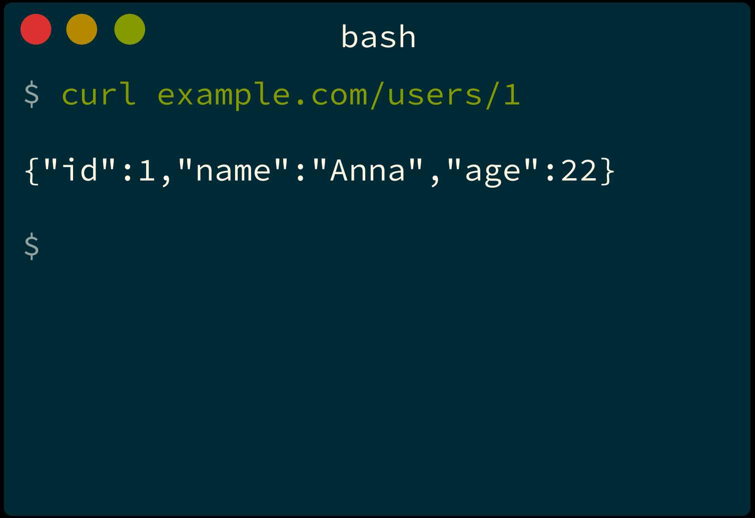
methods

- **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}
```



bash

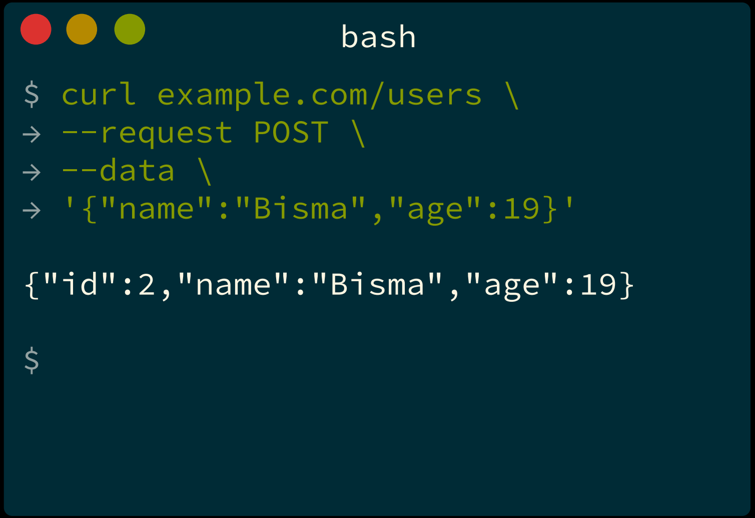
```
$ curl example.com/users/1

{"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}
```

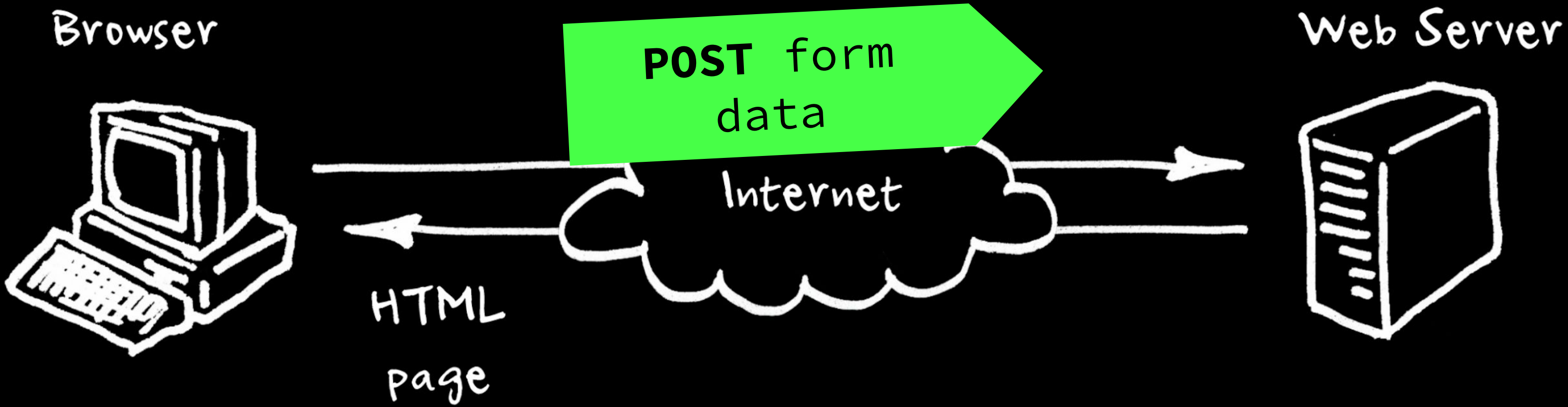


```
$ curl example.com/users \
→ --request POST \
→ --data \
→ '{"name":"Bisma","age":19}'

{"id":2,"name":"Bisma","age":19}

$
```

Submit a resource

http

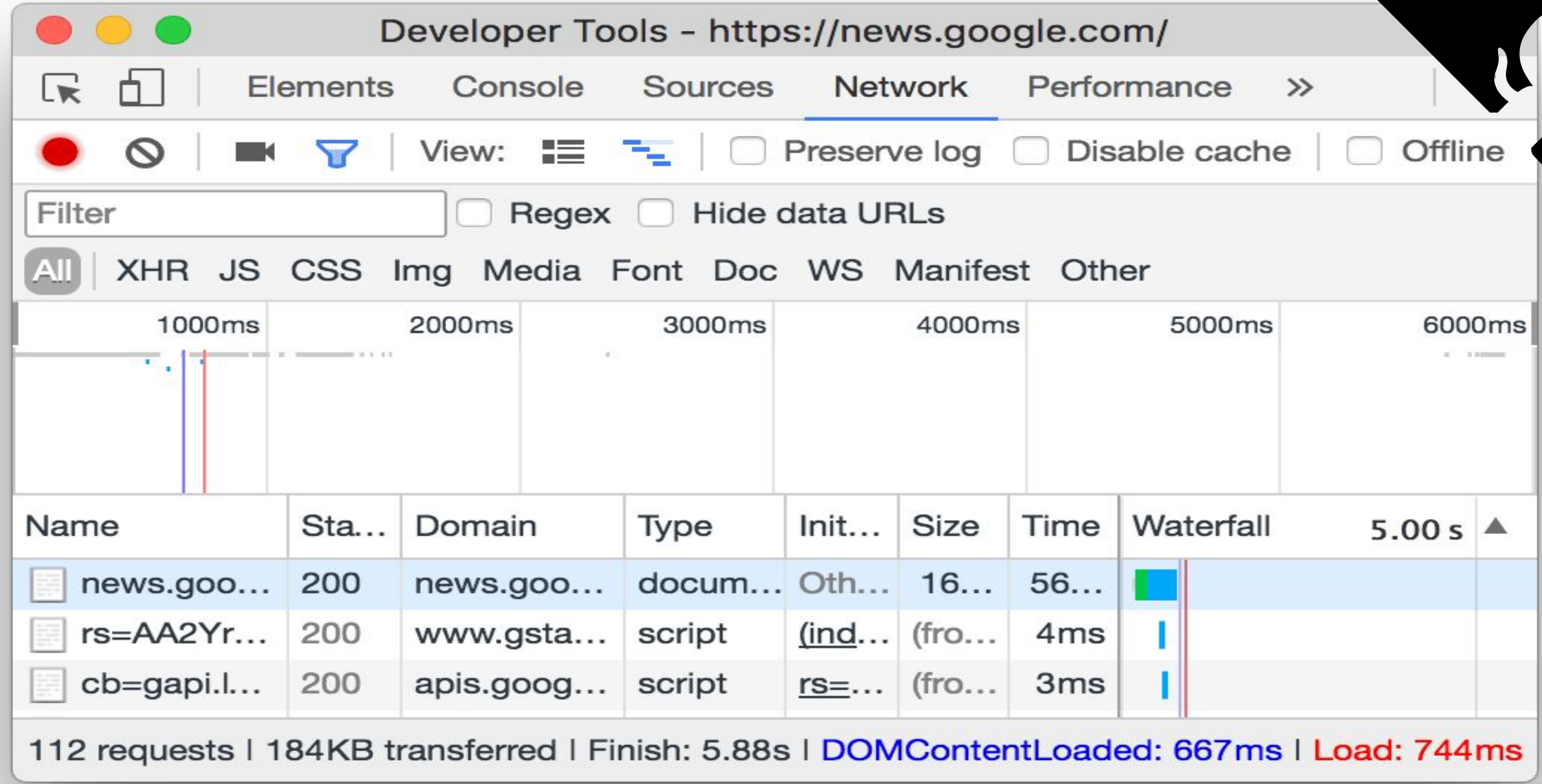
get/post



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,…


Developer Tools - https://news.google.com/

Elements Console Sources Network Performance »

Preserve log Disable cache Offline

Filter Regex Hide data URLs

All XHR JS CSS Img Media Font Doc WS Manifest Other

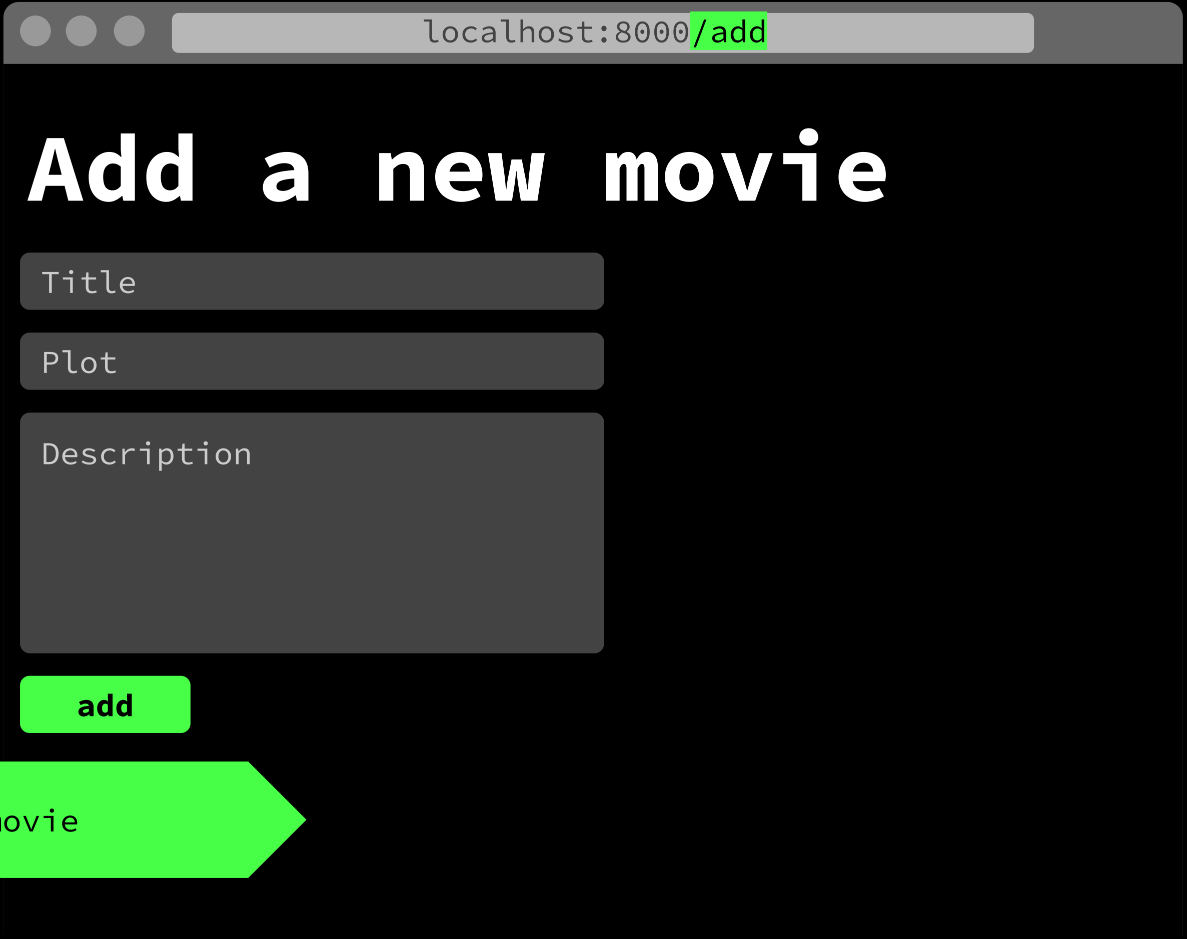
1000ms 2000ms 3000ms 4000ms 5000ms 6000ms

Name	Sta...	Domain	Type	Init...	Size	Time	Waterfall 5.00 s
news.goo...	200	news.goo...	docum...	Oth...	16...	56...	
rs=AA2Yr...	200	www.gsta...	script	(ind...	(fro...	4ms	
cb=gapi.l...	200	apis.goog...	script	rs=...	(fro...	3ms	

112 requests | 184KB transferred | Finish: 5.88s | DOMContentLoaded: 667ms | Load: 744ms

Live demo **network tab**

Forms



localhost:8000/add

Add a new movie

Title

Plot

Description

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>
```

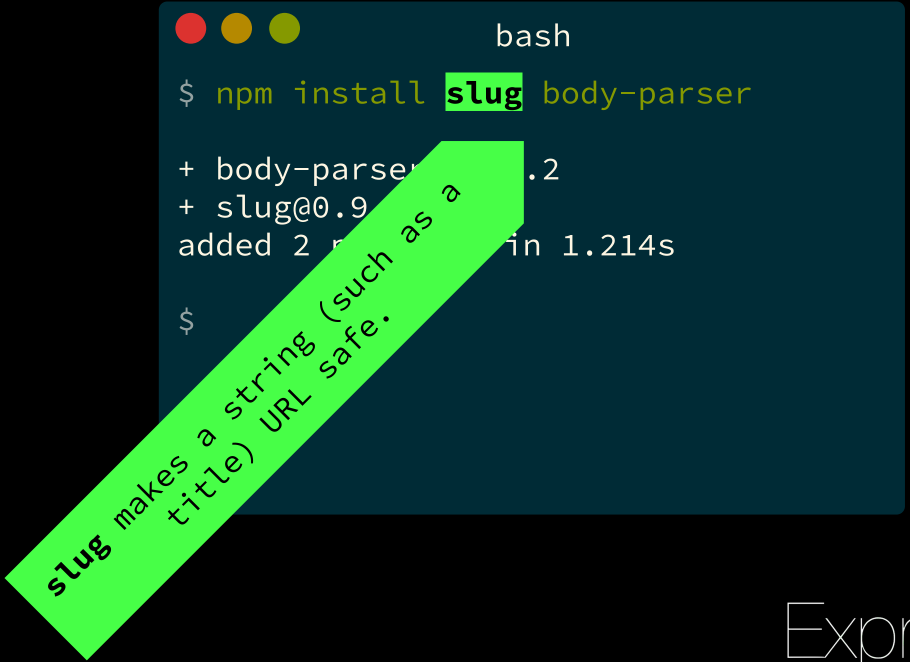
…send **title**, **plot**, and **description**

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
```



```
bash
$ npm install slug body-parser

+ body-parse       .2
+ slug@0.9
added 2            in 1.214s

$
```

slug makes a string (such as a title) URL safe.

Express

form

body

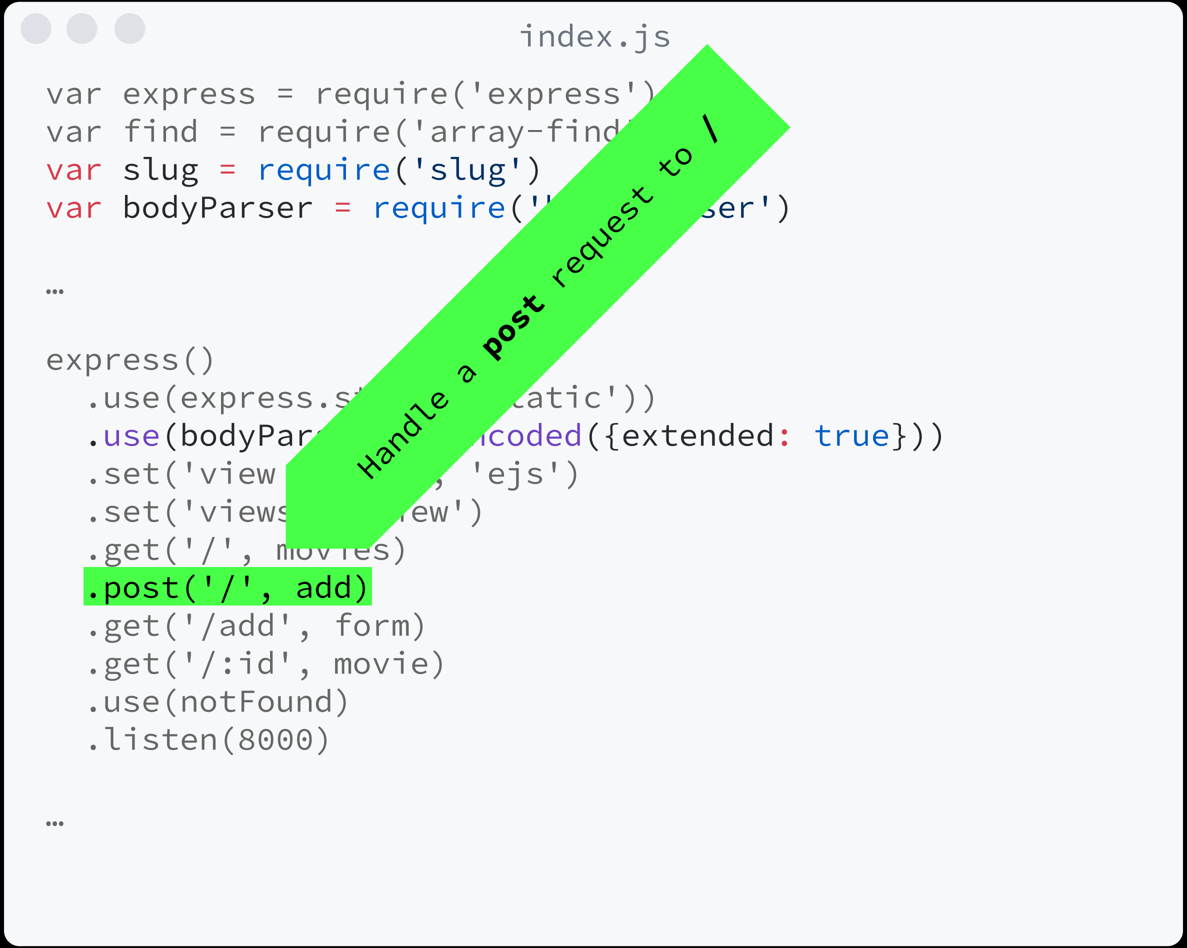
```
// Files
express-server/
├─ node_modules/
├─ static/
│  └─ index.css
├─ view/
│  ├─ add.ejs
│  ├─ detail.ejs
│  ├─ head.ejs
│  ├─ list.ejs
│  └─ not-found.ejs
├─ index.js
└─ package.json
```

bash

```
$ npm install slug body-parser

+ body-parser@1.18.2
+ slug@0.9.1
added 2 packages in 1

$
```

body-parser parses most things you give it (json, forms, etc)

Express



index.js

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

…

express()
  .use(express.s          atic'))
  .use(bodyPar            ncoded({extended: true}))
  .set('view          , 'ejs')
  .set('views         ew')
  .get('/', mov  es)
  .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**

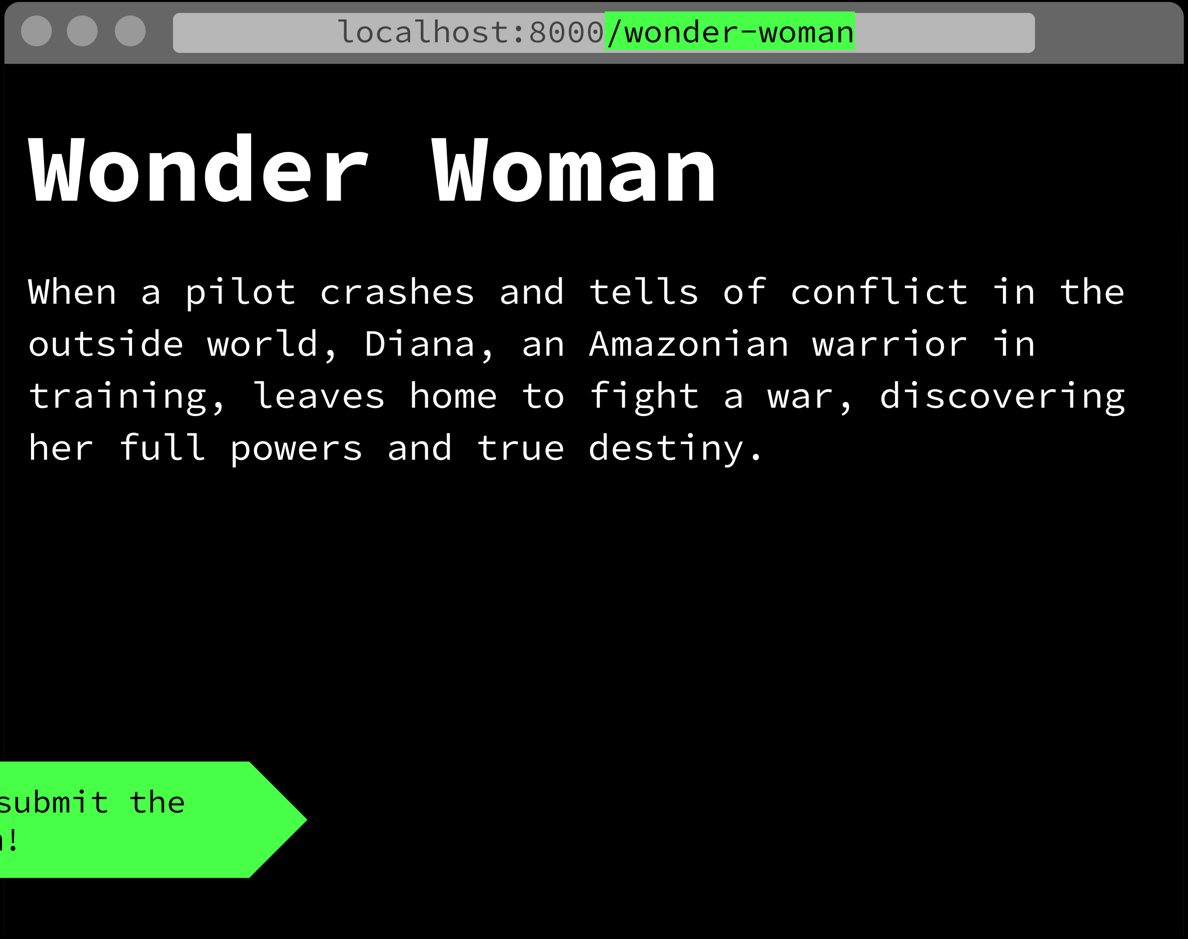
Express

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

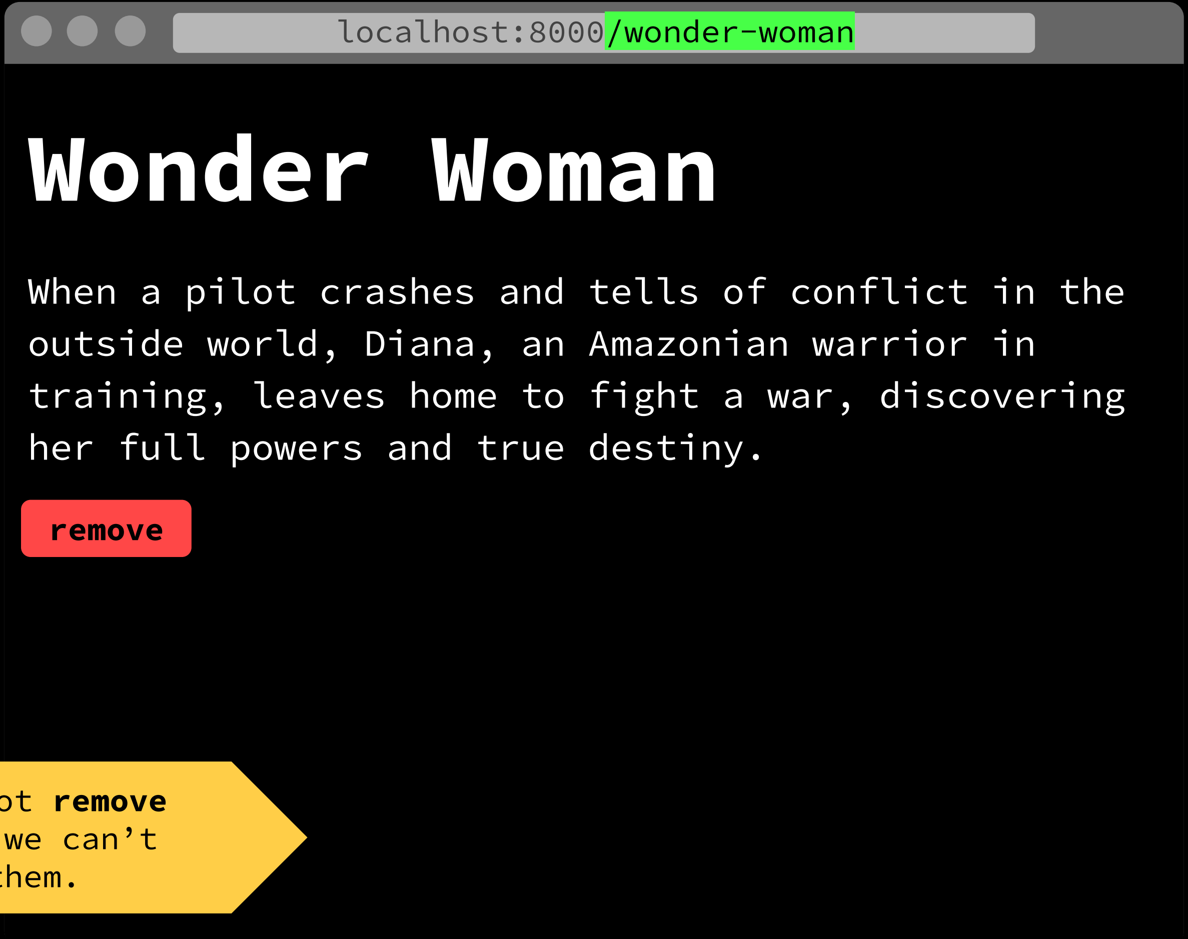


Wonder Woman

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

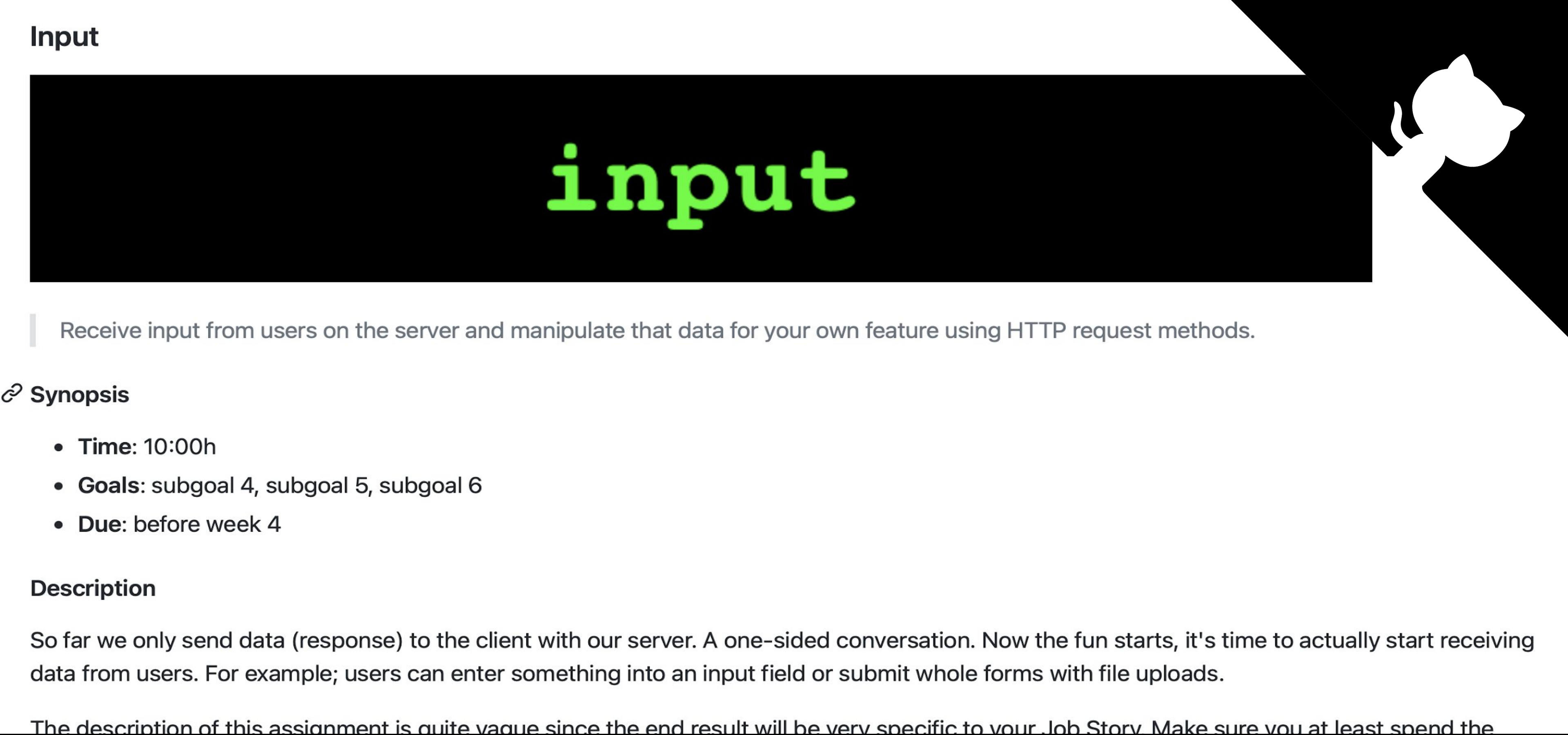


Wonder Woman

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.

…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

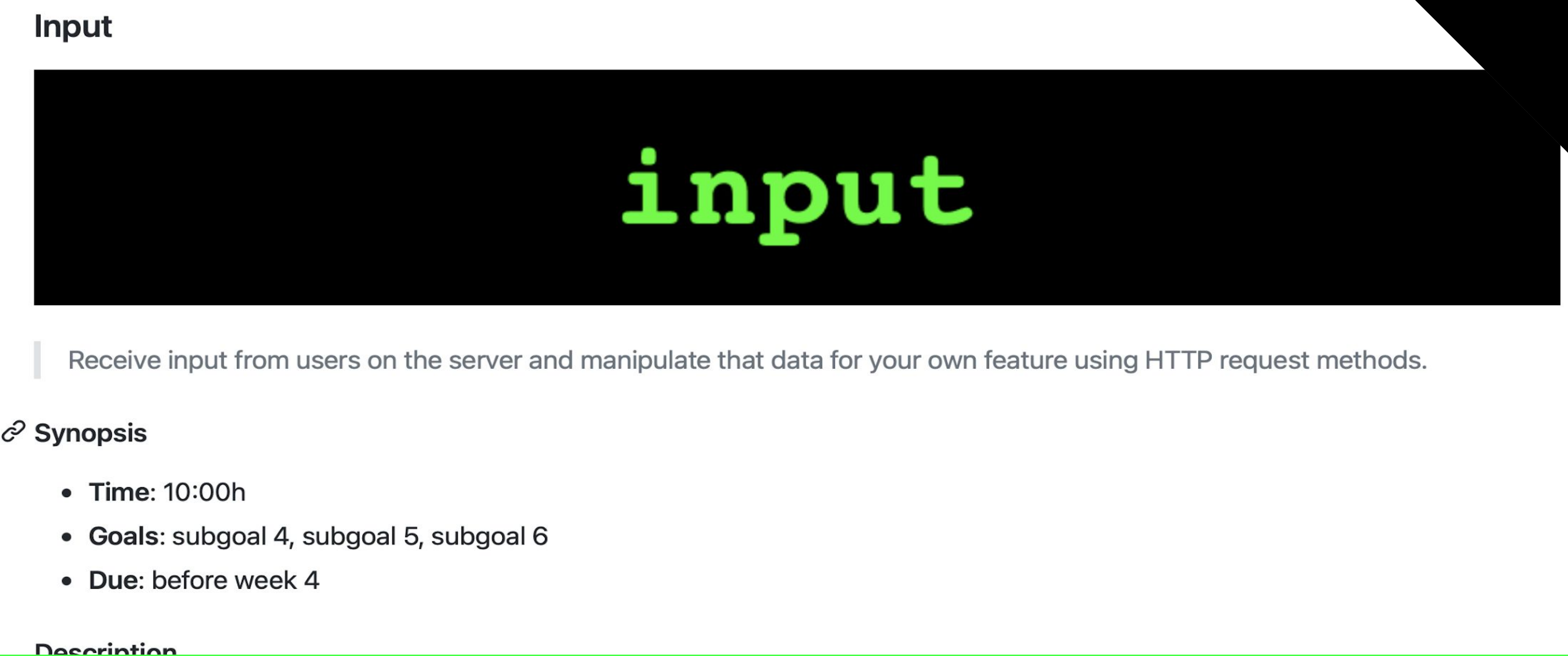
- **Time**: 10:00h
- **Goals**: subgoal 4, subgoal 5, subgoal 6
- **Due**: before week 4

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

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

exit;

see you in lab-3b!