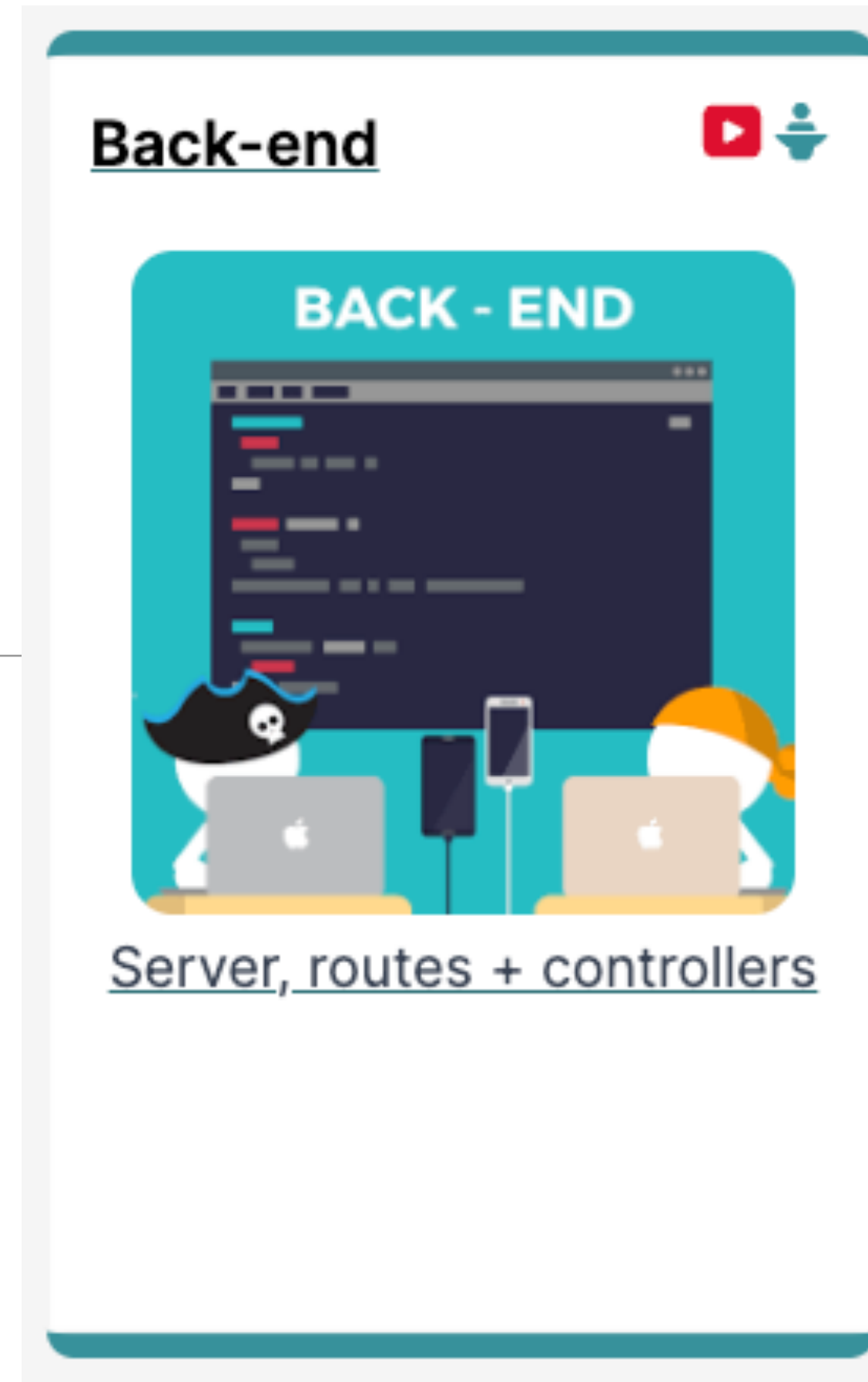
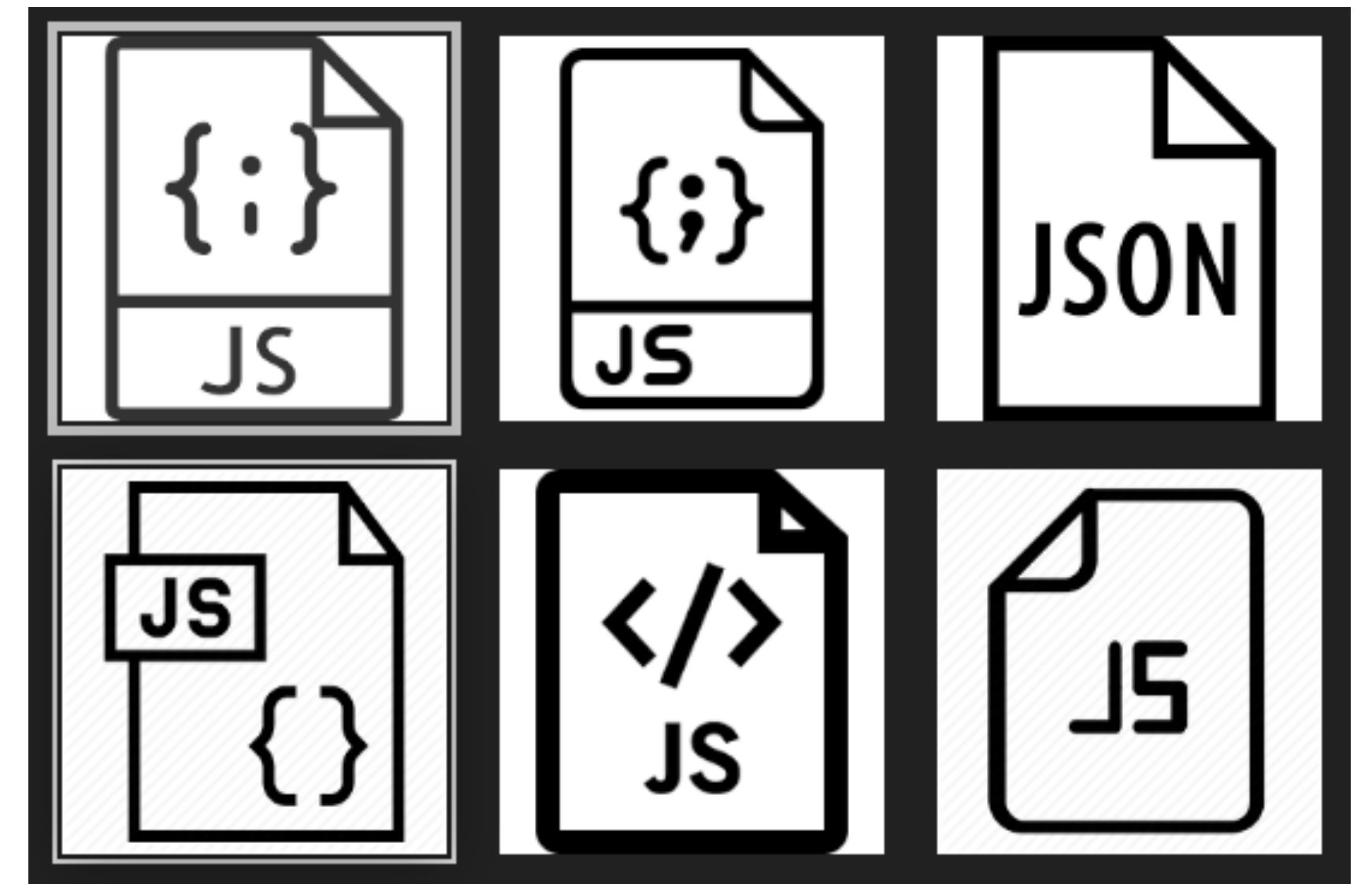


Back-end



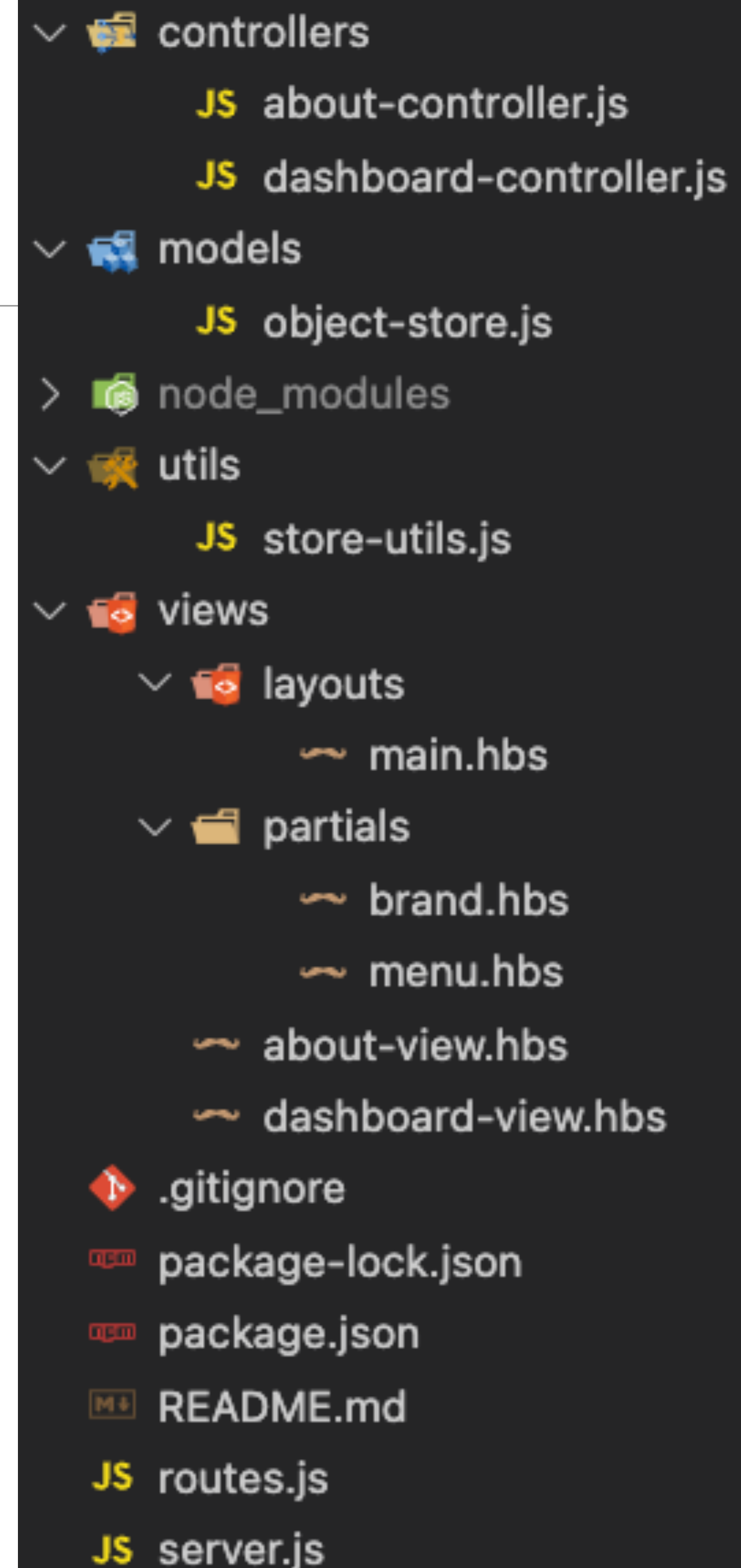
Javascript Modules

- To structure an application coherently, the backend consists of separate Javascript files.
- Objects declared in these files must be
 - exported by one file
 - imported by another
- In order to keep each module focused on a specific responsibility



Application Structure

- App implements Routes + Model/View/Controller Architecture
- These objects collaborate to support structured, predictable application workflow



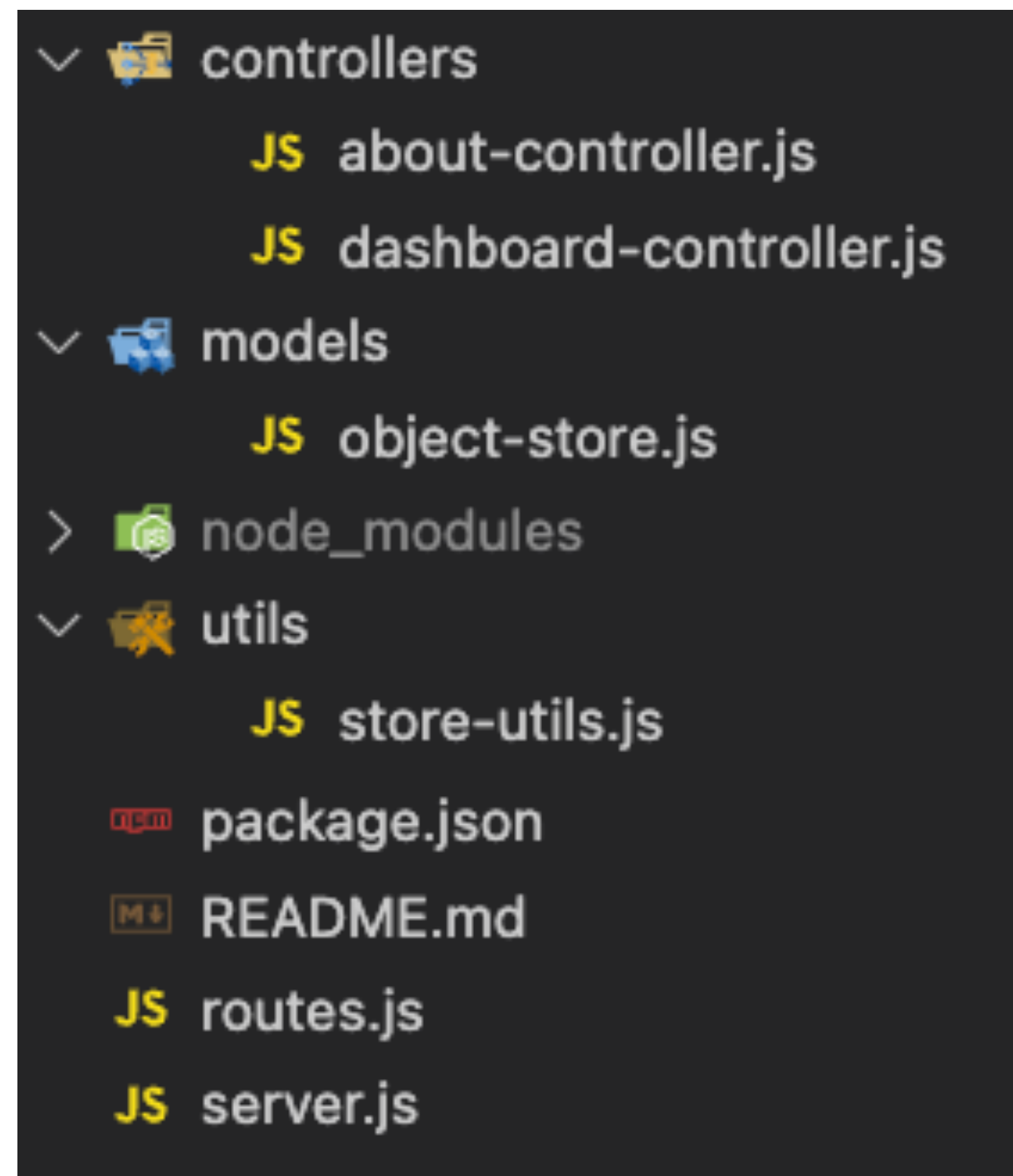
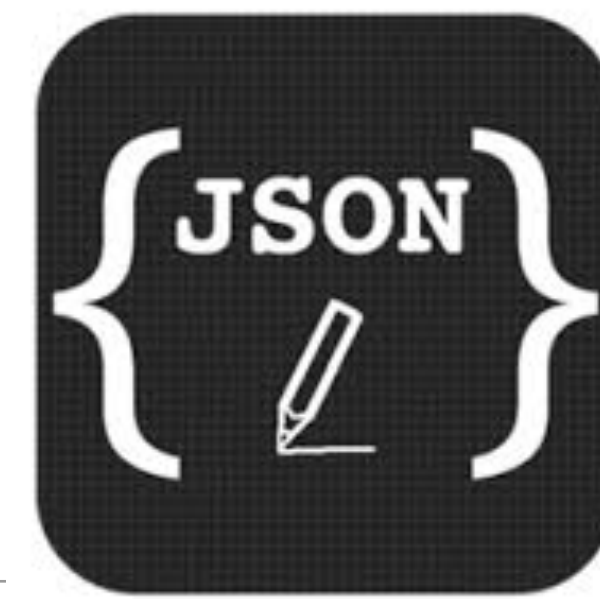
```

  controllers
    JS about-controller.js
    JS dashboard-controller.js
  models
    JS object-store.js
  > node_modules
  utils
    JS store-utils.js
  views
    layouts
      main.hbs
    partials
      brand.hbs
      menu.hbs
      about-view.hbs
      dashboard-view.hbs
  .gitignore
  package-lock.json
  package.json
  README.md
  JS routes.js
  JS server.js

```

A screenshot of a file explorer interface with a dark background. It shows a hierarchical file structure for a web application. The 'controllers' folder contains 'about-controller.js' and 'dashboard-controller.js'. The 'models' folder contains 'object-store.js'. The 'node_modules' folder is expanded with a right-pointing chevron. The 'utils' folder contains 'store-utils.js'. The 'views' folder contains a 'layouts' subfolder with 'main.hbs' and a 'partials' subfolder with 'brand.hbs', 'menu.hbs', 'about-view.hbs', and 'dashboard-view.hbs'. At the bottom are several files: '.gitignore', 'package-lock.json', 'package.json', 'README.md', 'routes.js', and 'server.js'. Each file or folder has a small icon to its left representing its type.

Back-end

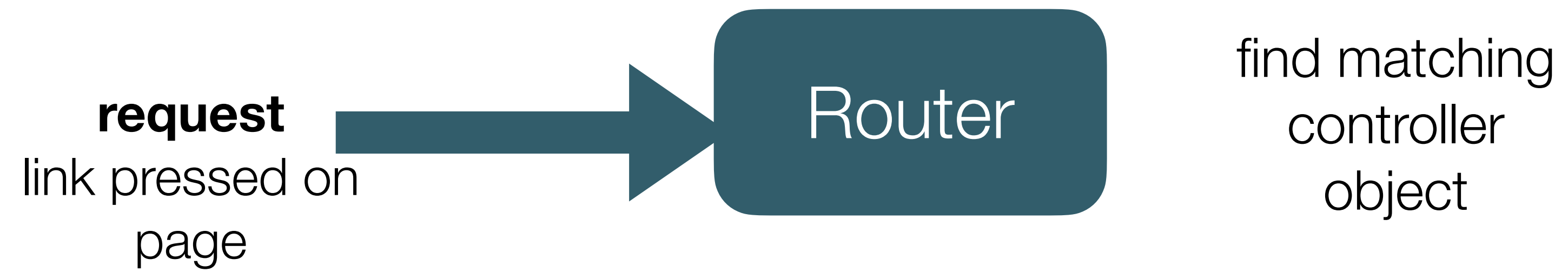
A yellow square with the letters 'JS' in a bold, black, sans-serif font.

- All written in Javascript + JSON
- Consists of:
 - **Server** - main entry point
 - **Routes** - supported urls
 - **Controllers** - objects to handle the routes
 - **Models** - Core information models represented by the application
 - **Config** - package.json

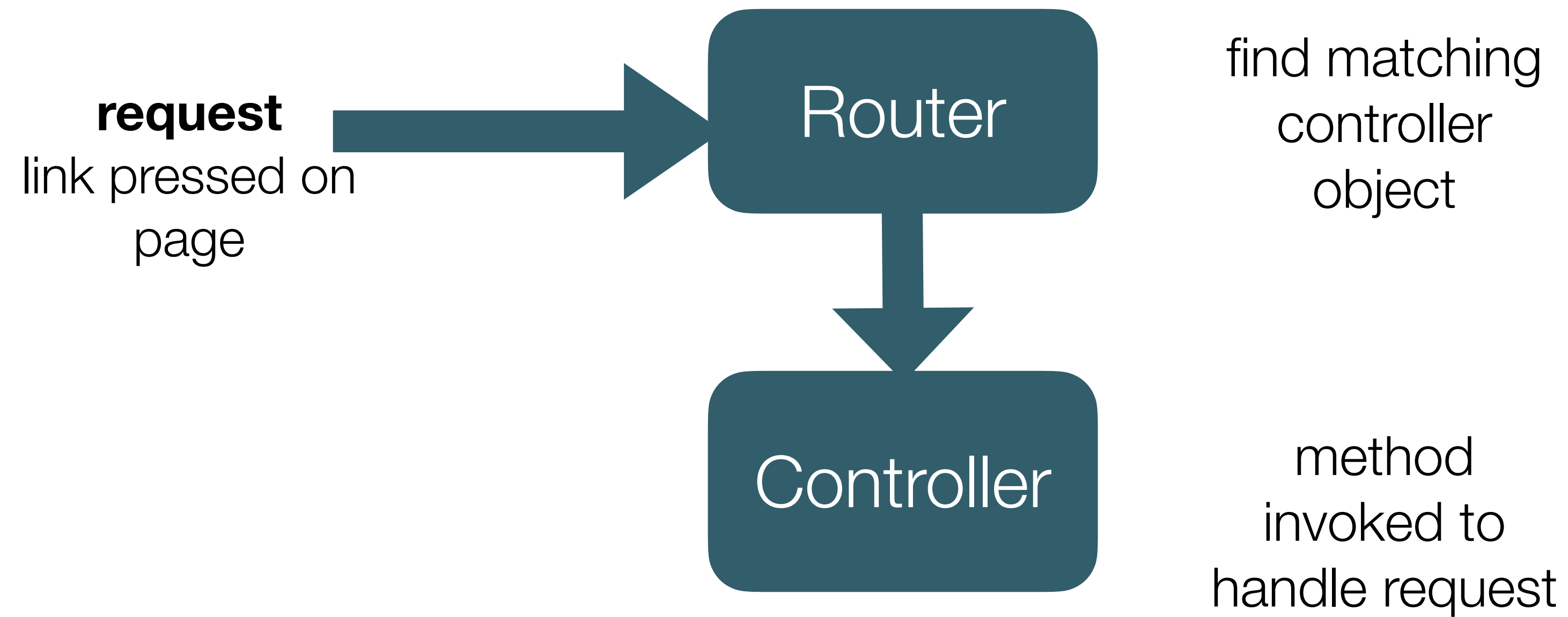
Request/Response Lifecycle

1. **Request** - link pressed on page
2. **Router** - find matching controller object
3. **Controller** - method invoked to handle request
4. **View** - data sent from controller to view to construct response
5. **Response** - complete page rendered into browser

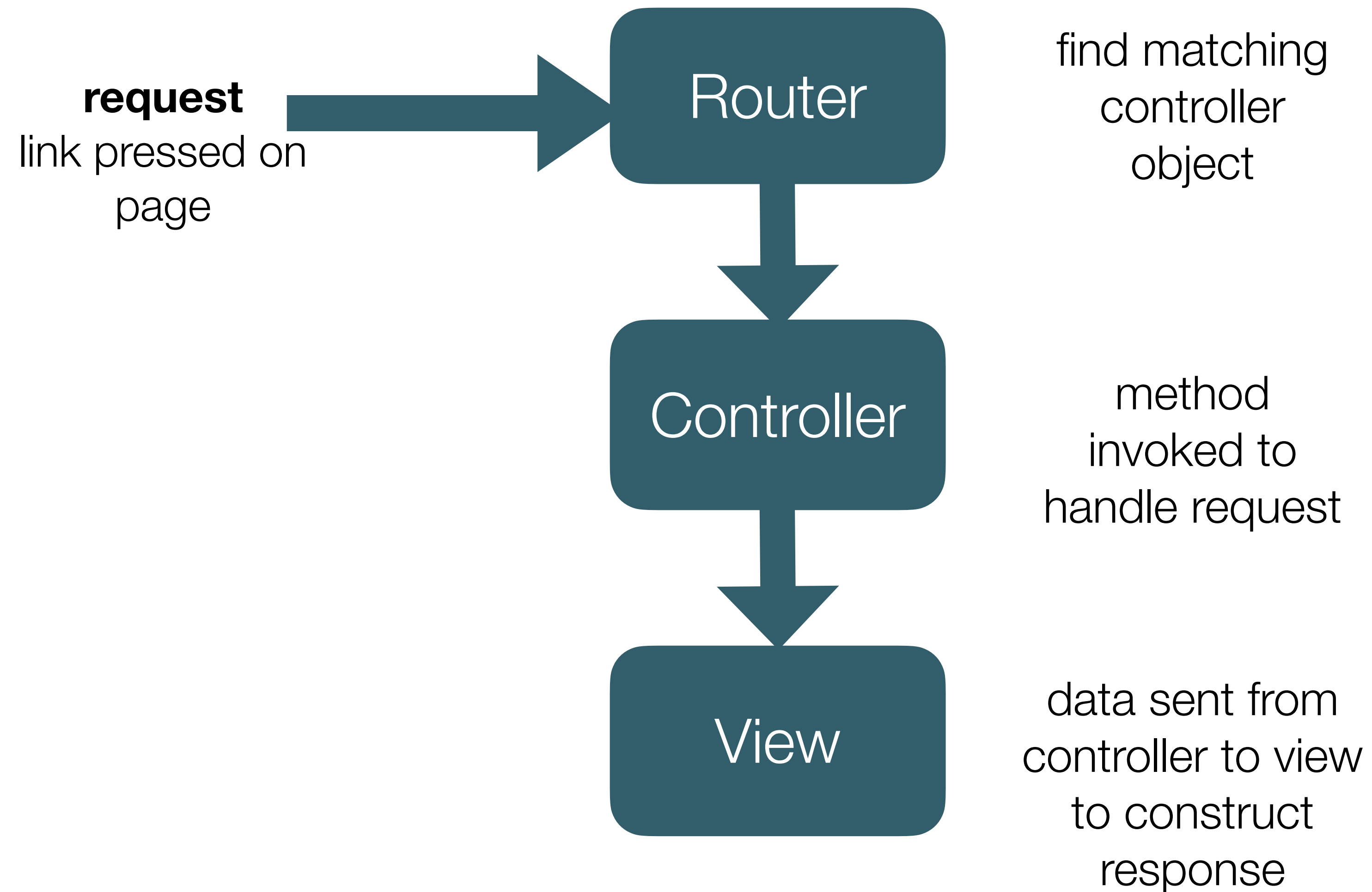
Router/Controller/View



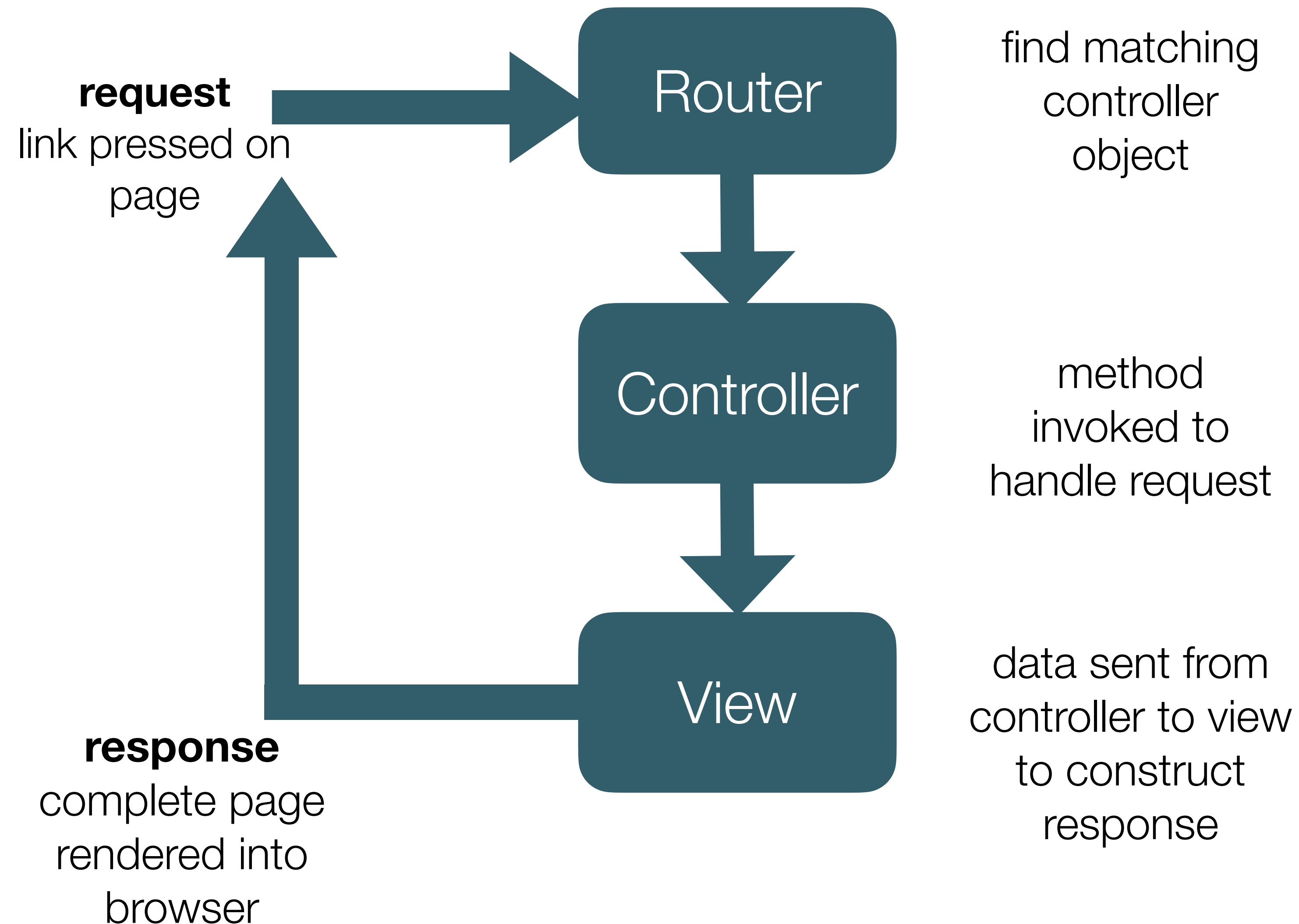
Router/Controller/View



Router/Controller/View

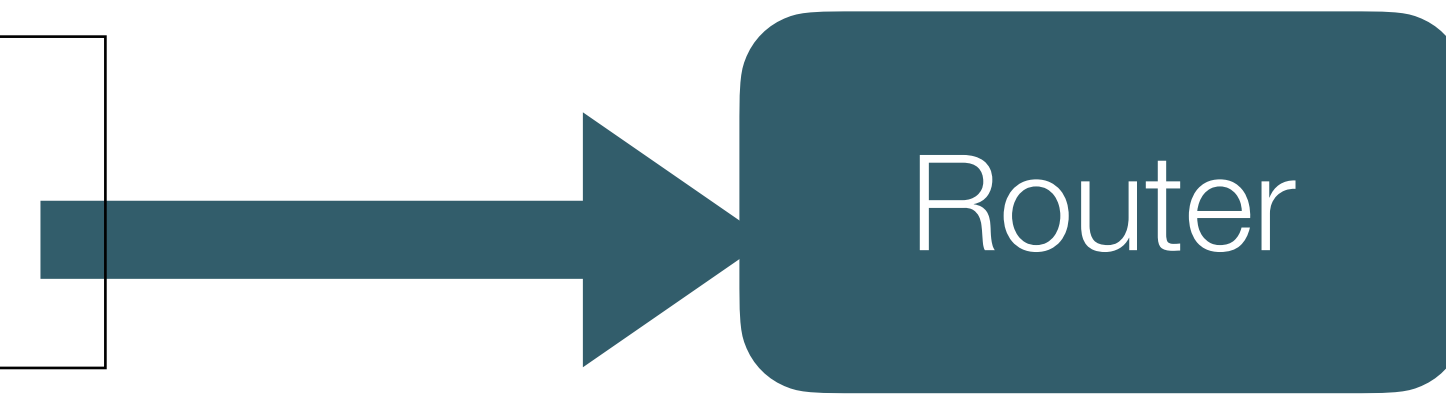


Router/Controller/View



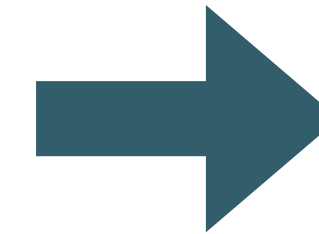
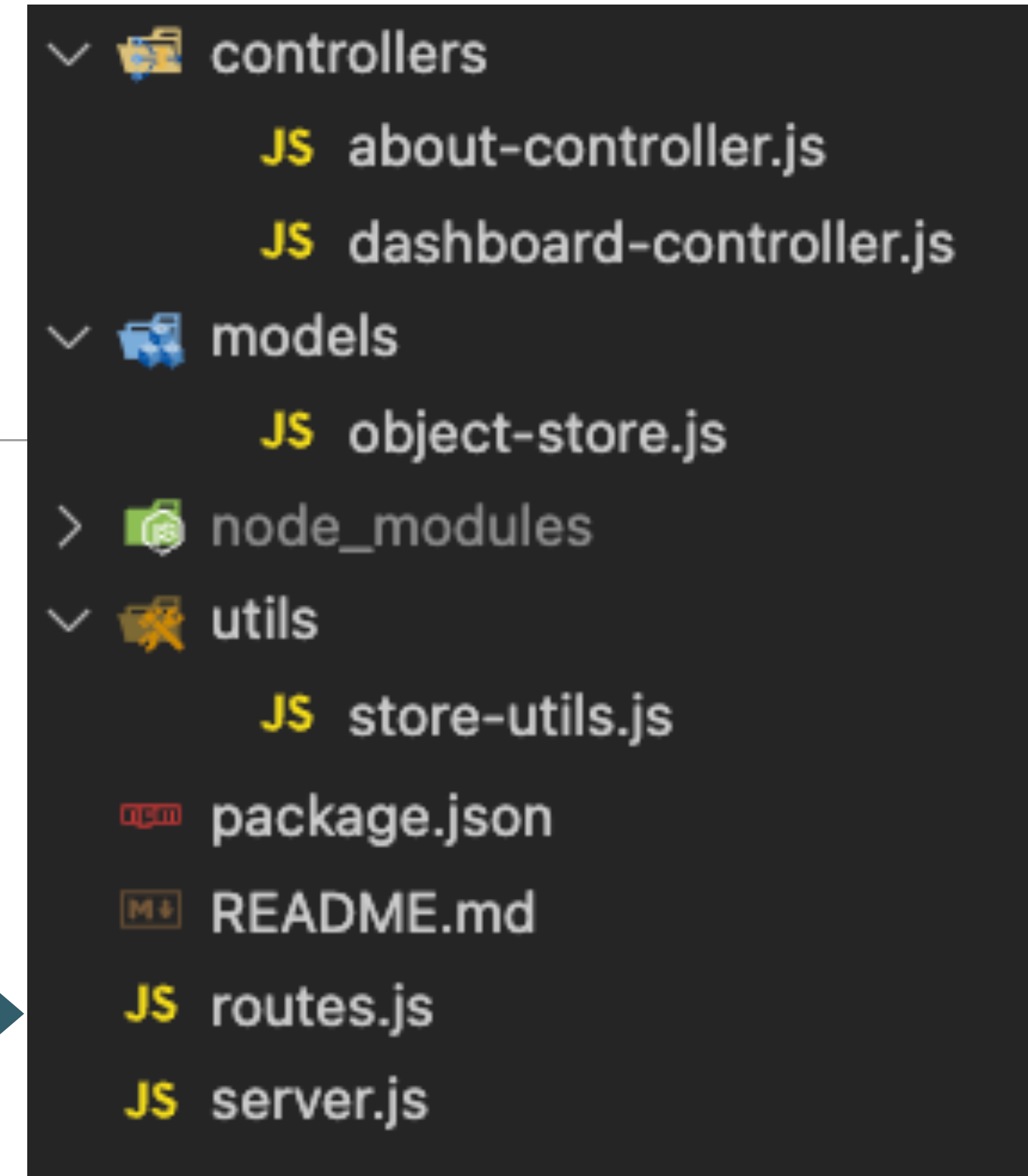
Request - link pressed on page

```
...  
<a id="dashboard" class="button" href="/dashboard"> Dashboard </a>  
<a id="about" class="button" href="/about"> About </a>  
...
```



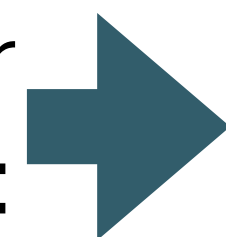
- Requests defined in links in views:
 - href in <a> tags
 - href in Menus
 - href in Buttons
 - action links in forms

Router - find matching controller object

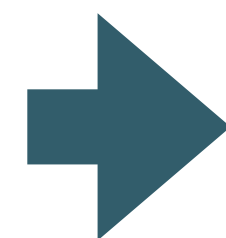


routes.js

Import controller
objects:



Match these
two objects with
each of these
'links'



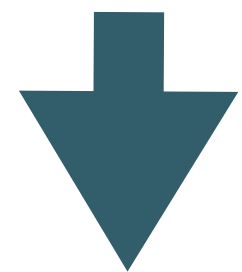
```
import express from "express";
import { dashboardController } from "../controllers/dashboard-controller.js";
import { aboutController } from "../controllers/about-controller.js";

export const router = express.Router();

router.get("/", dashboardController.index);
router.get("/dashboard", dashboardController.index);
router.get("/about", aboutController.index);
```

Router Behaviour

If user selects these links...



'/'



dashboardController.index

/dashboard'



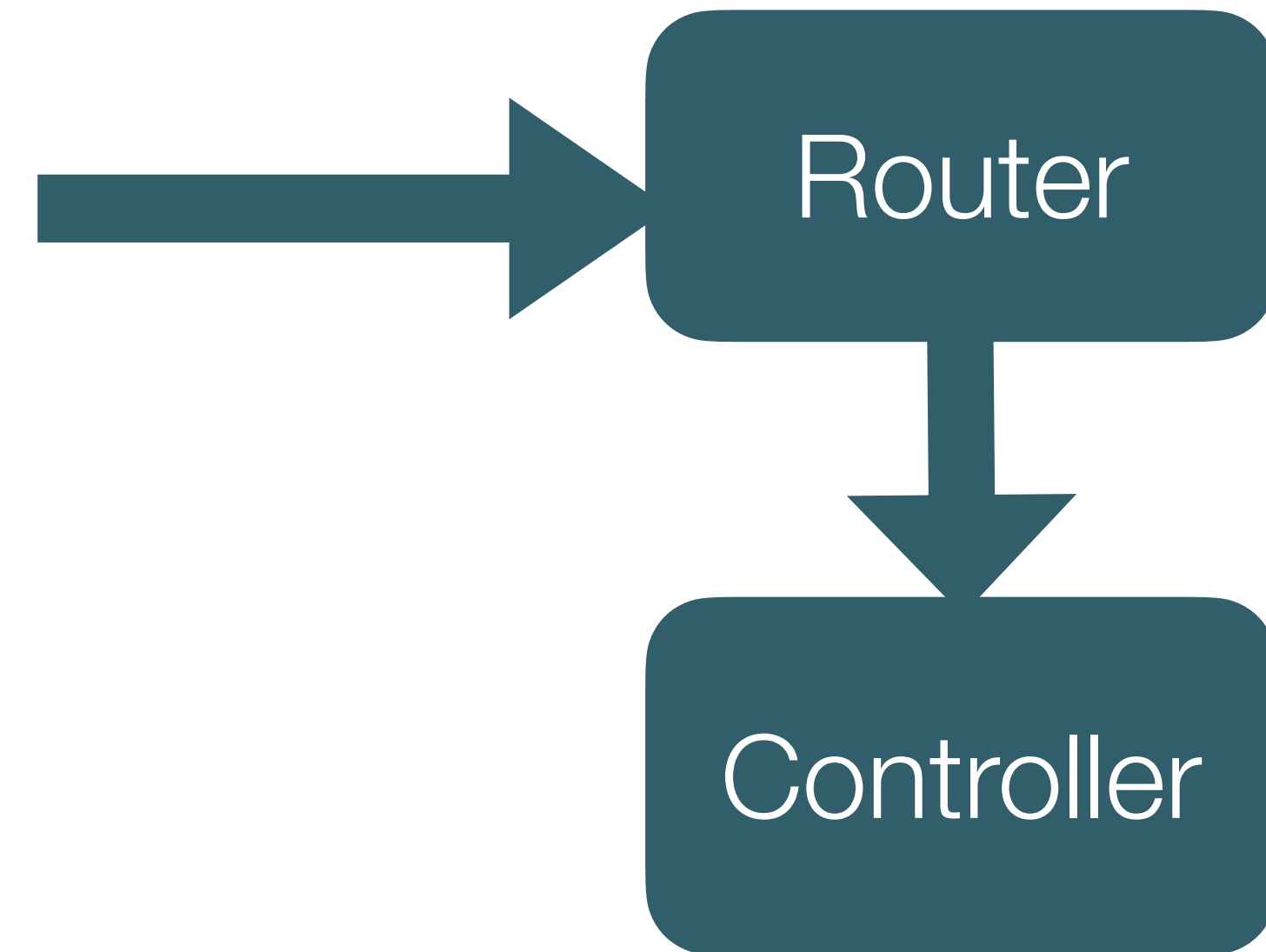
dashboardController.index

/about'



aboutController.index

...then call the corresponding controller methods

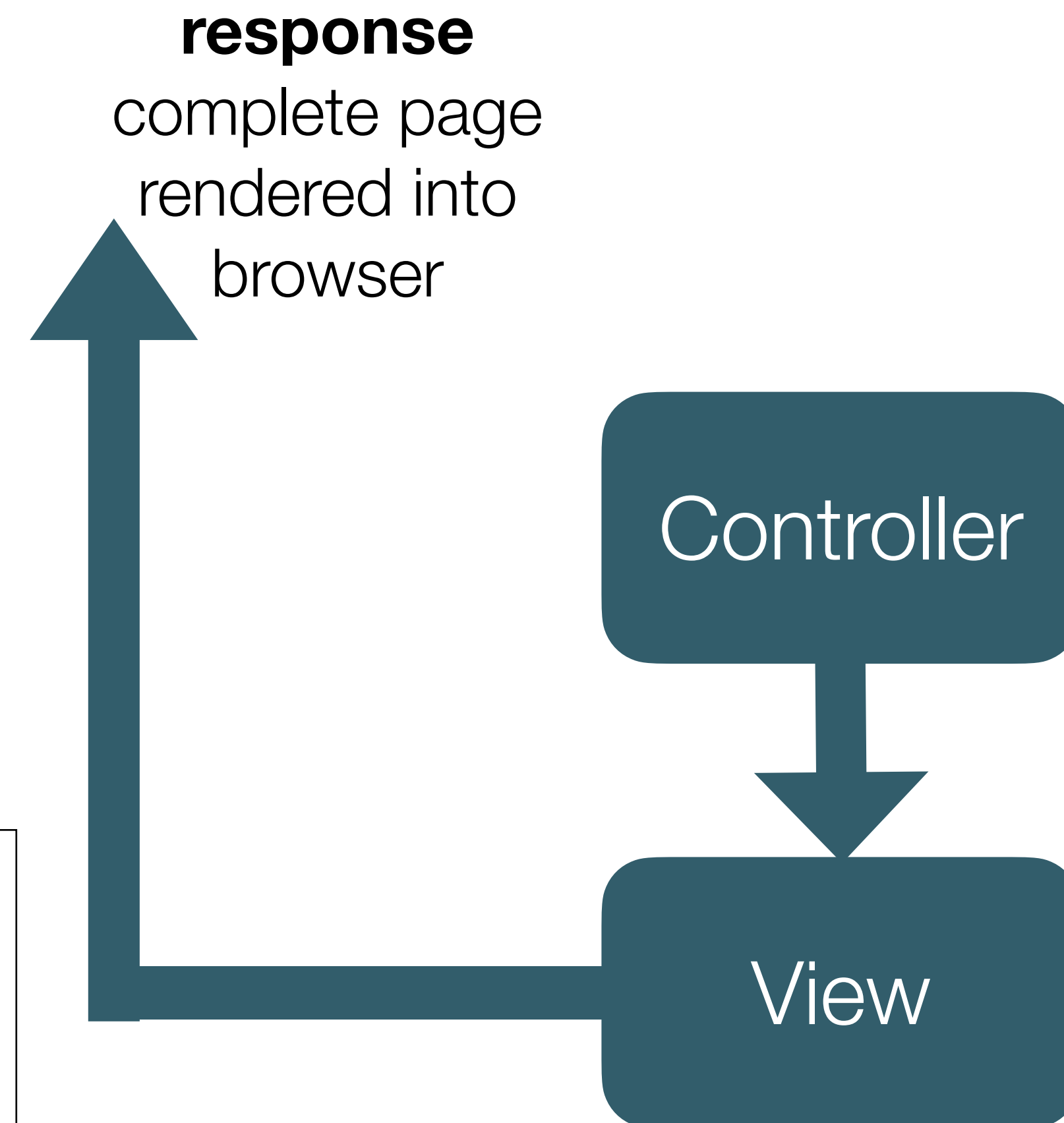
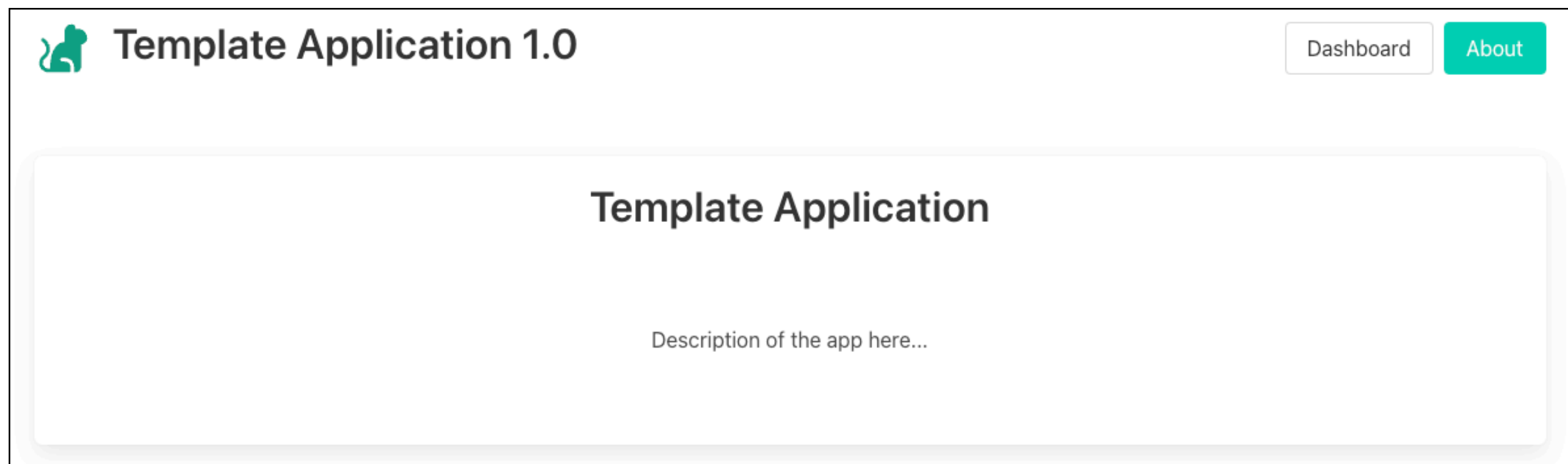


Controller method invoked to handle request

about-controller.js

```
export const aboutController = {  
  index(request, response) {  
    const viewData = {  
      title: "About This Application",  
    };  
    console.log("about rendering");  
    response.render("about-view", viewData);  
  },  
};
```

The About controller



The 'About' controller object -

index method parameters

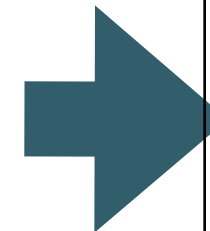
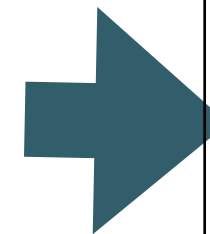
- Has a single method - index, which has 2 parameters:
- **request** : object containing details of the user request
- **response**: object to be used to send response back to browser

```
export const aboutController = {  
  index(request, response) {  
    const viewData = {  
      title: "About This Application",  
    };  
    console.log("about rendering");  
    response.render("about-view", viewData);  
  },  
};
```

The 'About' controller **index function body**

Create an object called **viewData**, containing a single property: **title**

logs a message to the console (glitch) console, not chrome console)



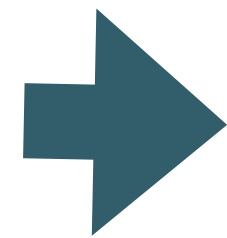
```
export const aboutController = {  
  index(request, response) {  
    const viewData = {  
      title: "About This Application",  
    };  
    console.log("about rendering");  
    response.render("about-view", viewData);  
  },  
};
```

Data sent from controller to view to construct response

Calls **render** method on
response with 2
parameters:

name of view to render
(about)

object to inject into the
view prior to rendering it
(viewData)



```
export const aboutController = {  
  index(request, response) {  
    const viewData = {  
      title: "About This Application",  
    };  
    console.log("about rendering");  
    response.render("about-view", viewData);  
  },  
};
```


Back-end + Front-End

about.js

```
export const aboutController = {
  index(request, response) {
    const viewData = {
      title: "About This Application",
    };
    console.log("about rendering");
    response.render("about-view", viewData);
  },
};
```

about-view.hbs

```
{{> menu active="about"}}

<section class="box is-3 has-text-centered">
  <h3 class="title">
    Template Application
  </h3>
  <div class="section">
    Description of the app here...
  </div>
</section>
```

menu.hbs

```
<nav class="navbar mb-6">
  <div class="navbar-brand">
    {{> brand}}
  </div>
  <div class="navbar-menu" id="navMenu">
    <div class="navbar-end">
      <div class="navbar-item">
        <div class="buttons">
          <a id="dashboard" class="button" href="/dashboard"> Dashboard </a>
          <a id="about" class="button" href="/about"> About </a>
        </div>
      </div>
    </div>
  </div>
</nav>

<script>
  document.getElementById("{{active}}").classList.add("is-primary");
</script>
```

main.hbs

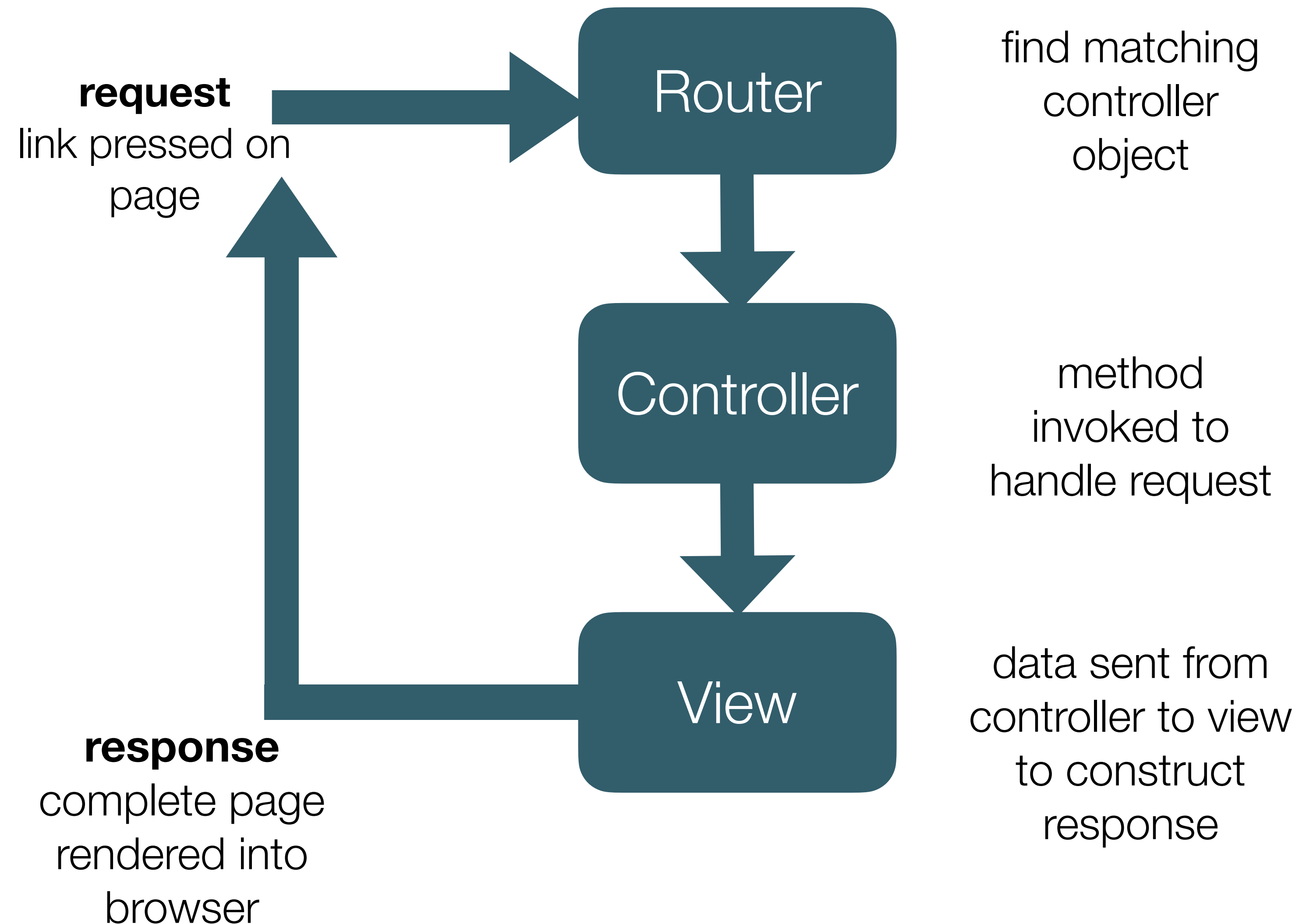
```
<!DOCTYPE html>
<html lang="en-IE">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>{{title}}</title>
  <link rel="stylesheet" href="https://cdn.jsdelivriv....bulma.min.css">
</head>

<body>
  <div class="container">
    {{{body}}}
  </div>
</body>

</html>
```

Router/Controller/View

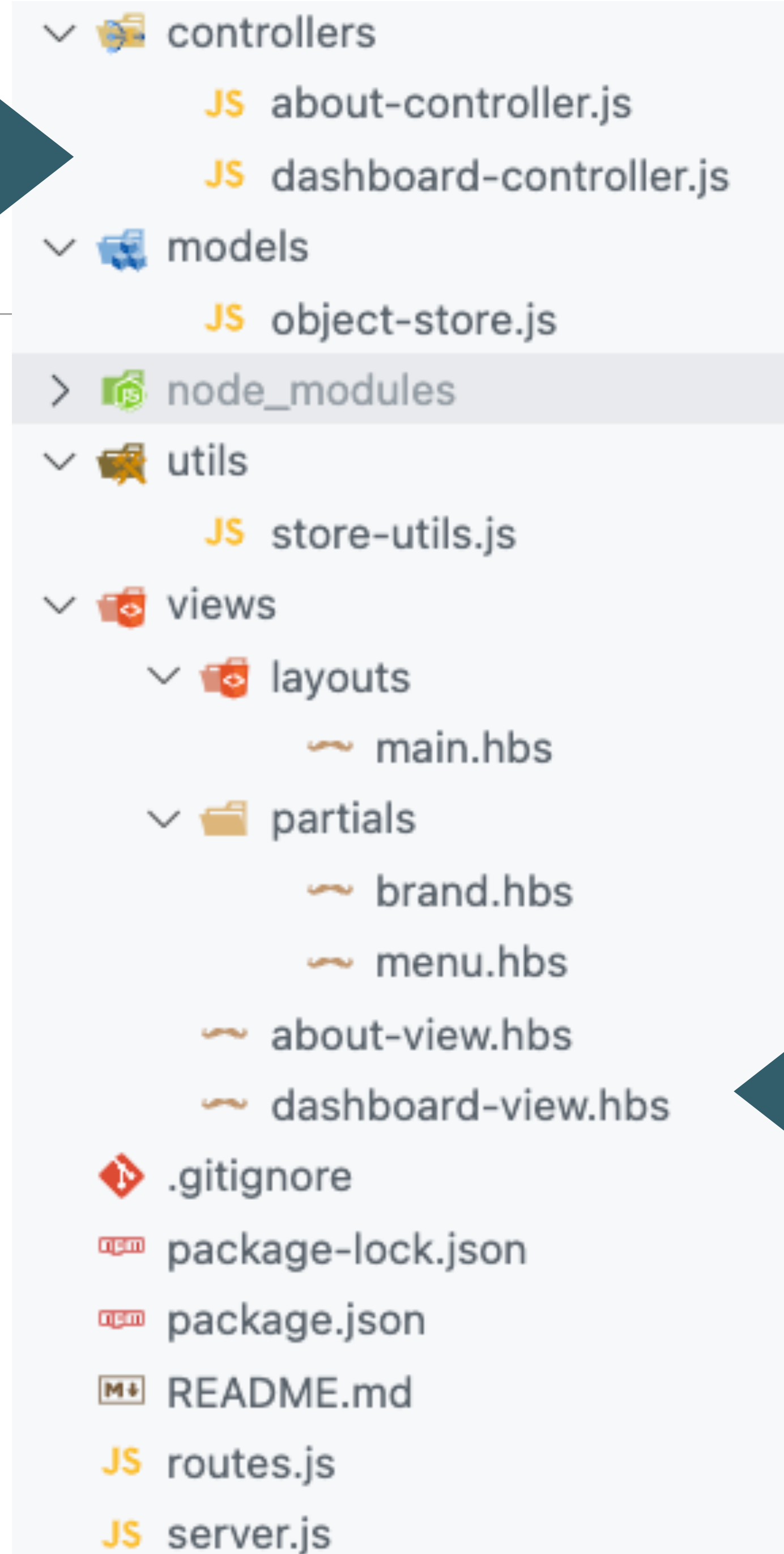
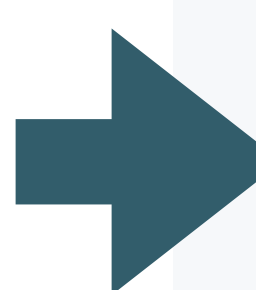


Dashboard Controller

```
export const dashboardController = {  
  async index(request, response) {  
    const viewData = {  
      title: "Template Application",  
    };  
    console.log("dashboard rendering");  
    response.render("dashboard-view", viewData);  
  },  
};
```

Application Structure

2 Controllers
which will render
2 matching views



- ▼ controllers
 - JS about-controller.js
 - JS dashboard-controller.js
- ▼ models
 - JS object-store.js
- > node_modules
- ▼ utils
 - JS store-utils.js
- ▼ views
 - ▼ layouts
 - main.hbs
 - ▼ partials
 - brand.hbs
 - menu.hbs
 - about-view.hbs
 - dashboard-view.hbs
- .gitignore
- package-lock.json
- package.json
- README.md
- JS routes.js
- JS server.js

```
export const dashboardController = {  
  
  async index(request, response) {  
    const viewData = {  
      title: "Template Application",  
    };  
    console.log("dashboard rendering");  
    response.render("dashboard-view", viewData);  
  },  
  
};
```

routes.js

```
router.get("/dashboard", dashboardController.index);  
router.get("/about", aboutController.index);
```

```
export const aboutController = {  
  
  index(request, response) {  
    const viewData = {  
      title: "About This Application",  
    };  
    console.log("about rendering");  
    response.render("about-view", viewData);  
  },  
  
};
```

controllers/views

dashboard-controller.js

```
export const dashboardController = {  
  
  async index(request, response) {  
    const viewData = {  
      title: "Template Application",  
    };  
    console.log("dashboard rendering");  
    response.render("dashboard-view", viewData);  
  },  
  
};
```

dashboard-view.hbs

```
{{> menu active="dashboard"}}  
  
<section class="box has-text-centered">  
  <h3 class="is-3 title">  
    Main Dashboard  
  </h3>  
  <div class="section">  
    <p>  
      This is the main dashboard view  
    </p>  
  </div>  
</section>
```

about-controller.js

```
export const aboutController = {  
  
  index(request, response) {  
    const viewData = {  
      title: "About This Application",  
    };  
    console.log("about rendering");  
    response.render("about-view", viewData);  
  },  
  
};
```

about-view.hbs

```
{{> menu active="about"}}  
  
<section class="box is-3 has-text-centered">  
  <h3 class="title">  
    Template Application  
  </h3>  
  <div class="section">  
    Description of the app here...  
  </div>  
</section>
```

Back-end

