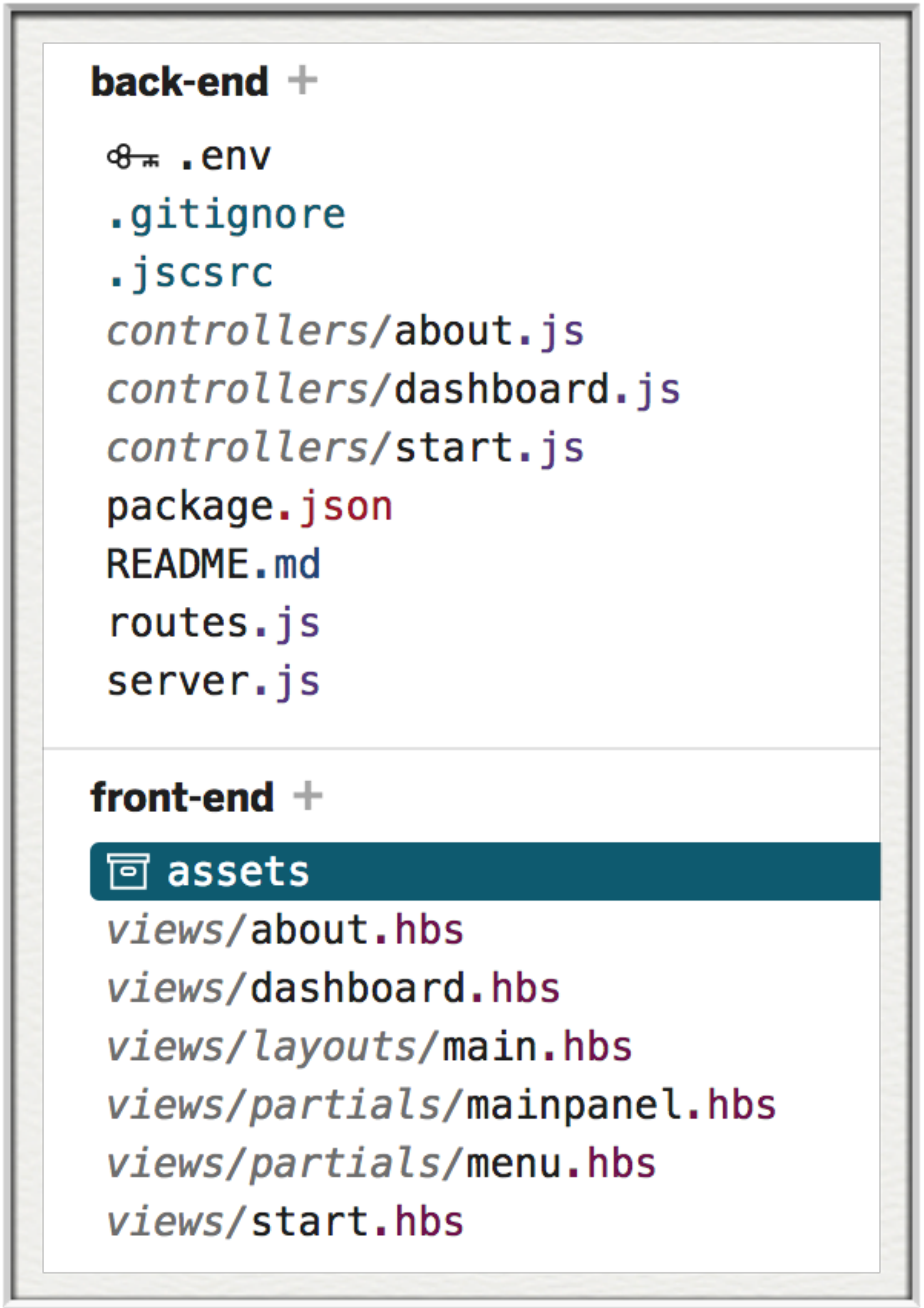
Back-end

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



back-end +

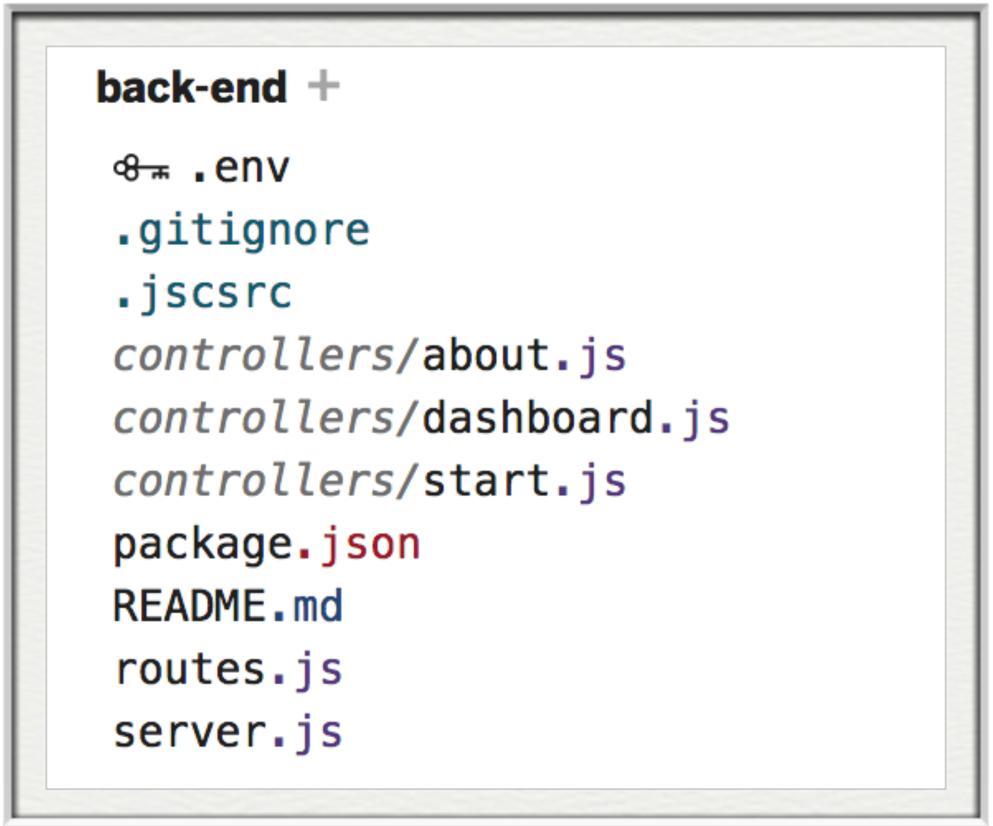
```
.env
.gitignore
.jscsrc
controllers/about.js
controllers/dashboard.js
controllers/start.js
package.json
README.md
routes.js
server.js
```

front-end +

```
assets
views/about.hbs
views/dashboard.hbs
views/layouts/main.hbs
views/partials/mainpanel.hbs
views/partials/menu.hbs
views/start.hbs
```

- App implements Routes + Model/View/Controller Architecture
- Back-end + Front-end collaborate to support structured, predictable application workflow

Back-end



back-end +

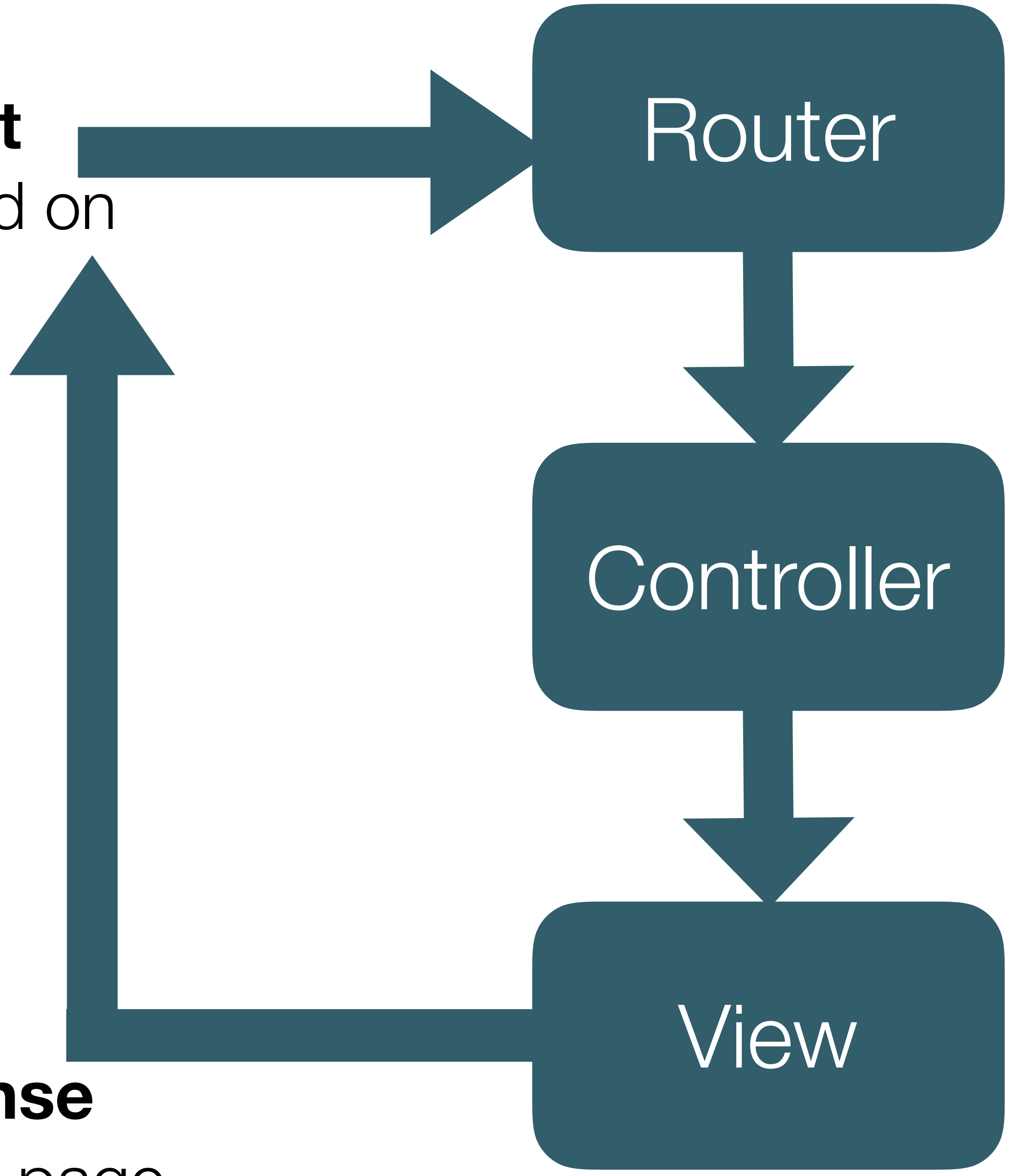
```
🔑 .env
.gitignore
.jscsrc
controllers/about.js
controllers/dashboard.js
controllers/start.js
package.json
README.md
routes.js
server.js
```

- All written in Javascript + JSON
- Consists of:
  - **Server** - main entry point
  - **Routes** - supported urls
  - **Controllers** - objects to handle the routes
  - **Config** - .gitignore, .jscsrc, env, package.json, readme.md
- Will include **Models** later…

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



request
link pressed on
page

Router

find matching
controller
object

Controller

method
invoked to
handle request

View

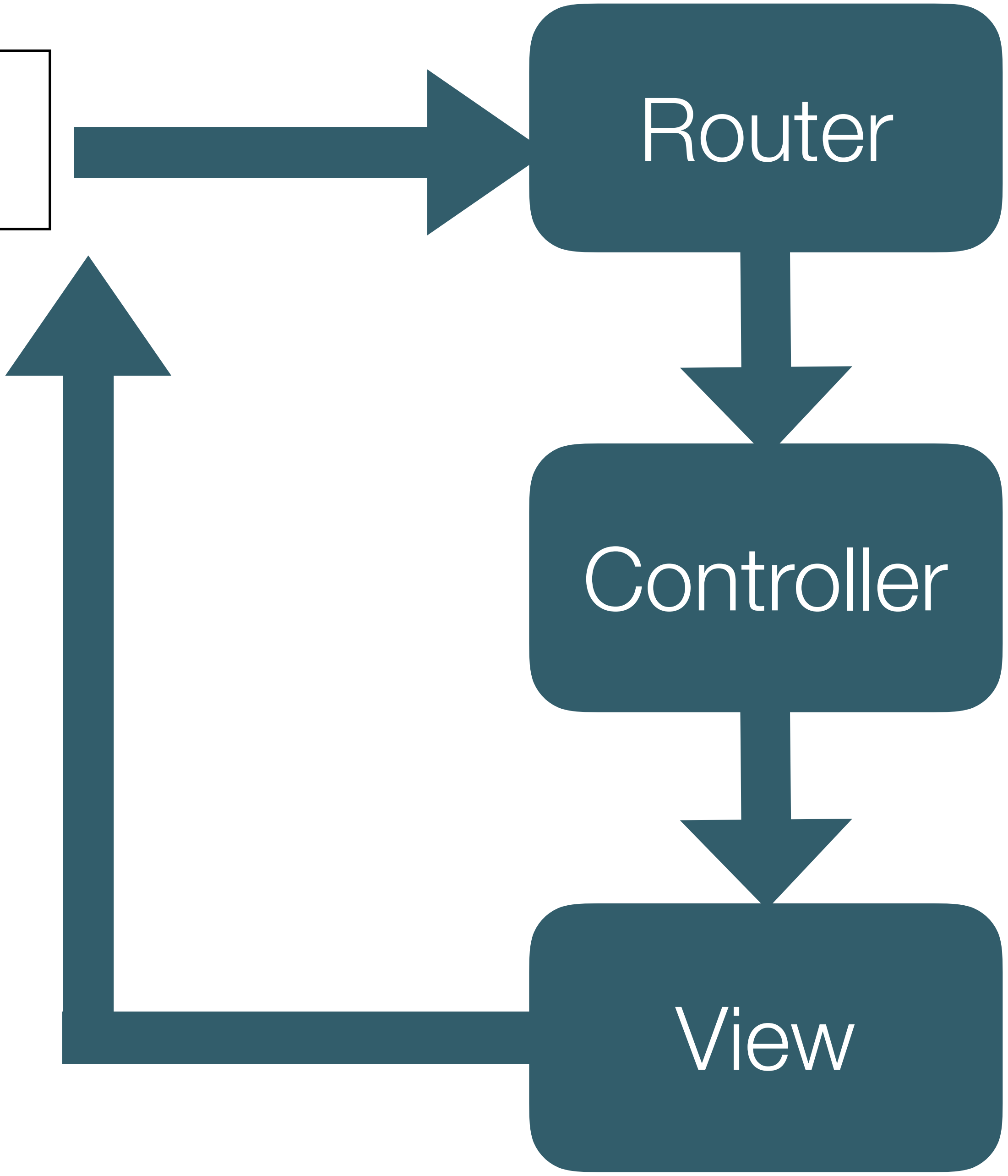
data sent from
controller to view
to construct
response

response
complete page
rendered into
browser

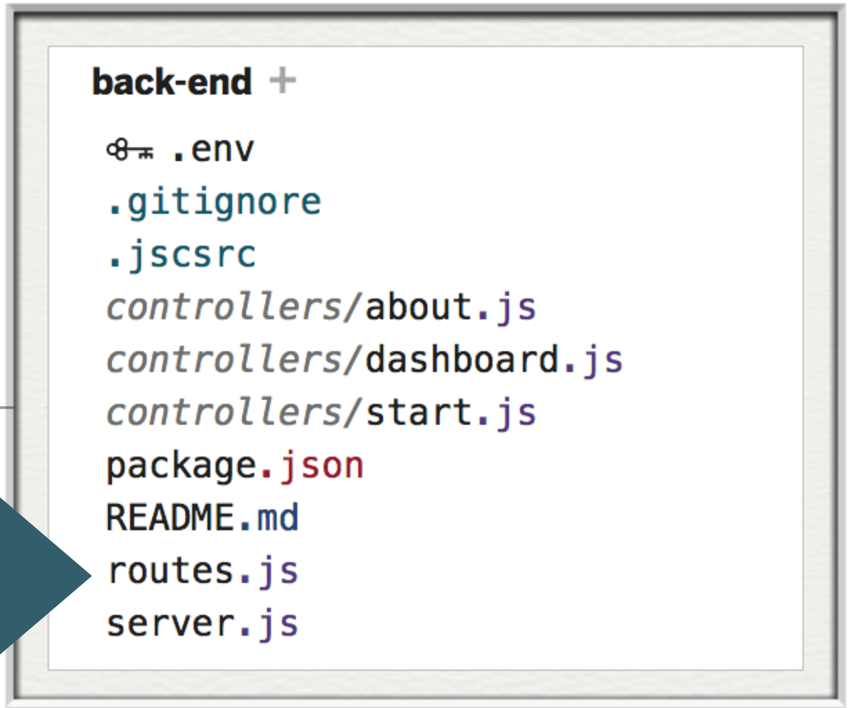
Request - link pressed on page

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

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



Router - find matching controller object



```
back-end +
.env
.gitignore
.jscsrc
controllers/about.js
controllers/dashboard.js
controllers/start.js
package.json
README.md
routes.js
server.js
```

routes.js

Import 3 objects:

```
...
const start = require('./controllers/start');
const dashboard = require('./controllers/dashboard.js');
const about = require('./controllers/about.js');

router.get('/', start.index);
router.get('/dashboard', dashboard.index);
router.get('/about', about.index);
...
```

Match these three objects with each of these ‘links’

Router Behaviour

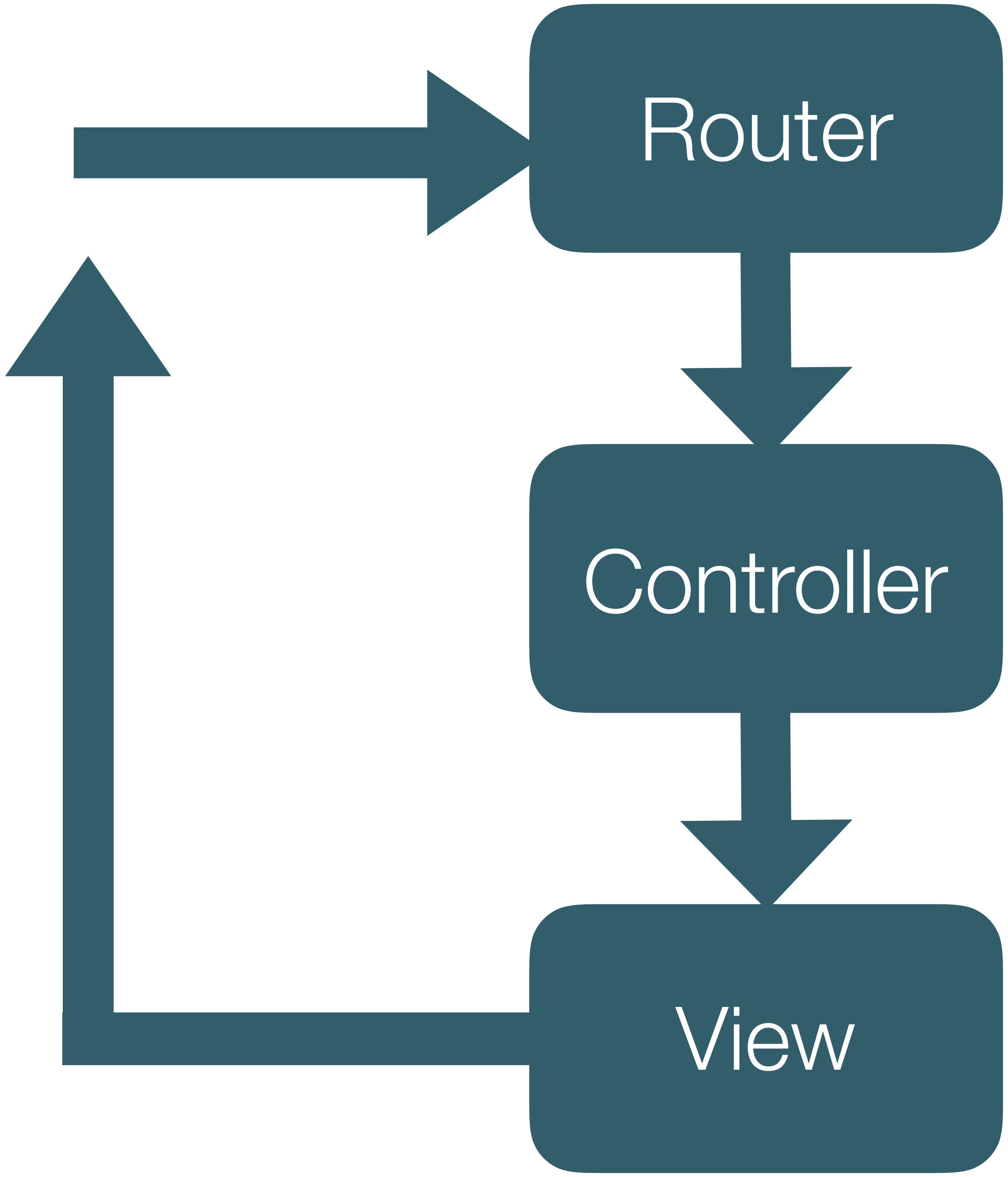
If user selects link page like these

Then call the corresponding controller methods

'/' → start.index

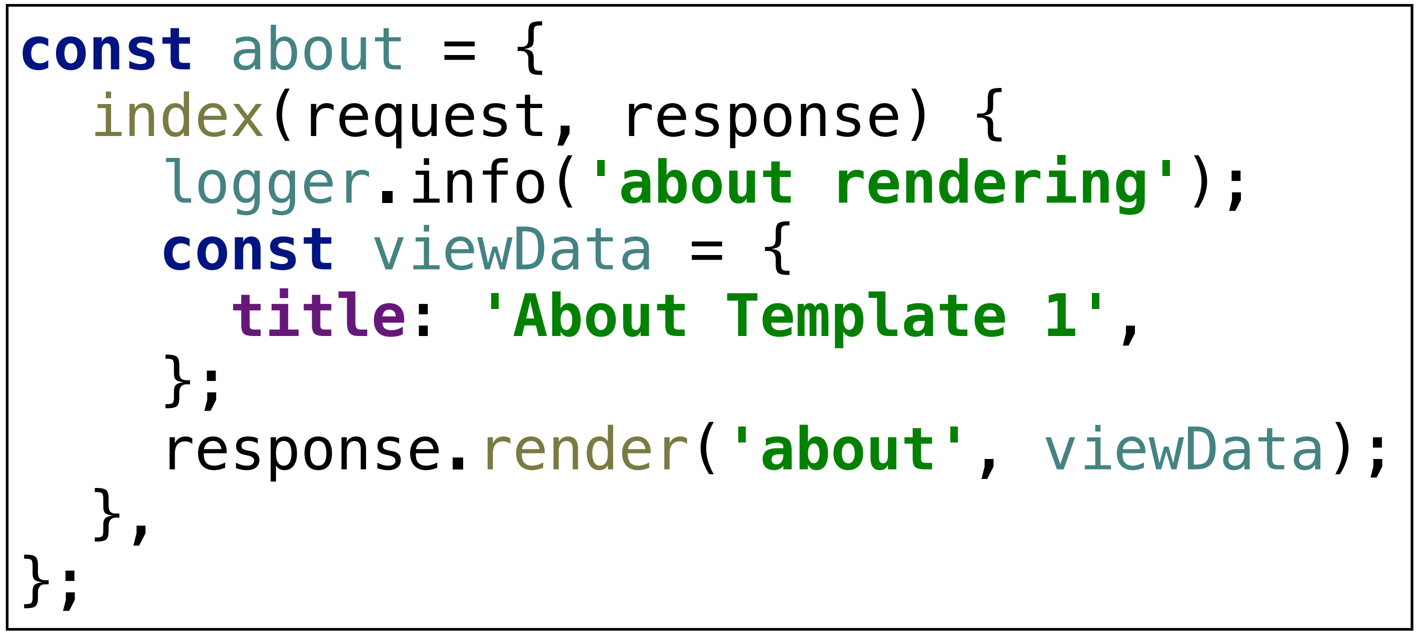
/dashboard' → dashboard.index

/about' → about.index



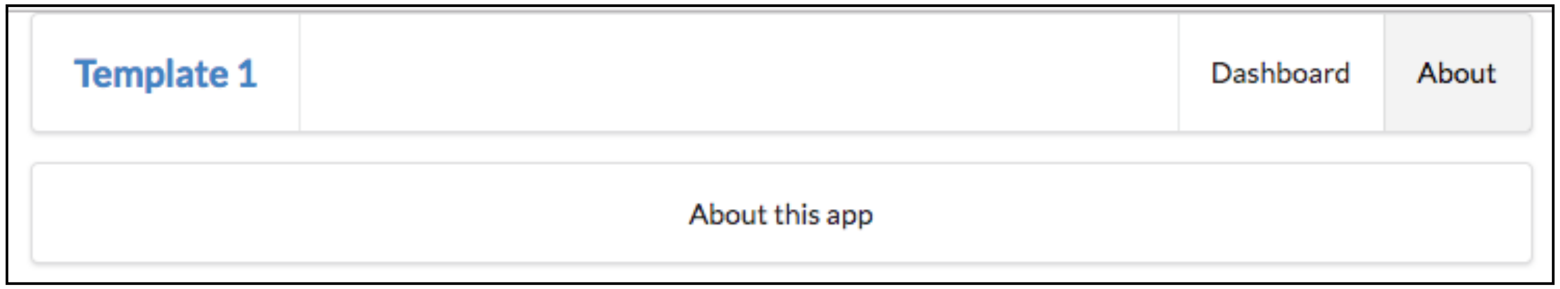
Controller method invoked to handle request

about.js

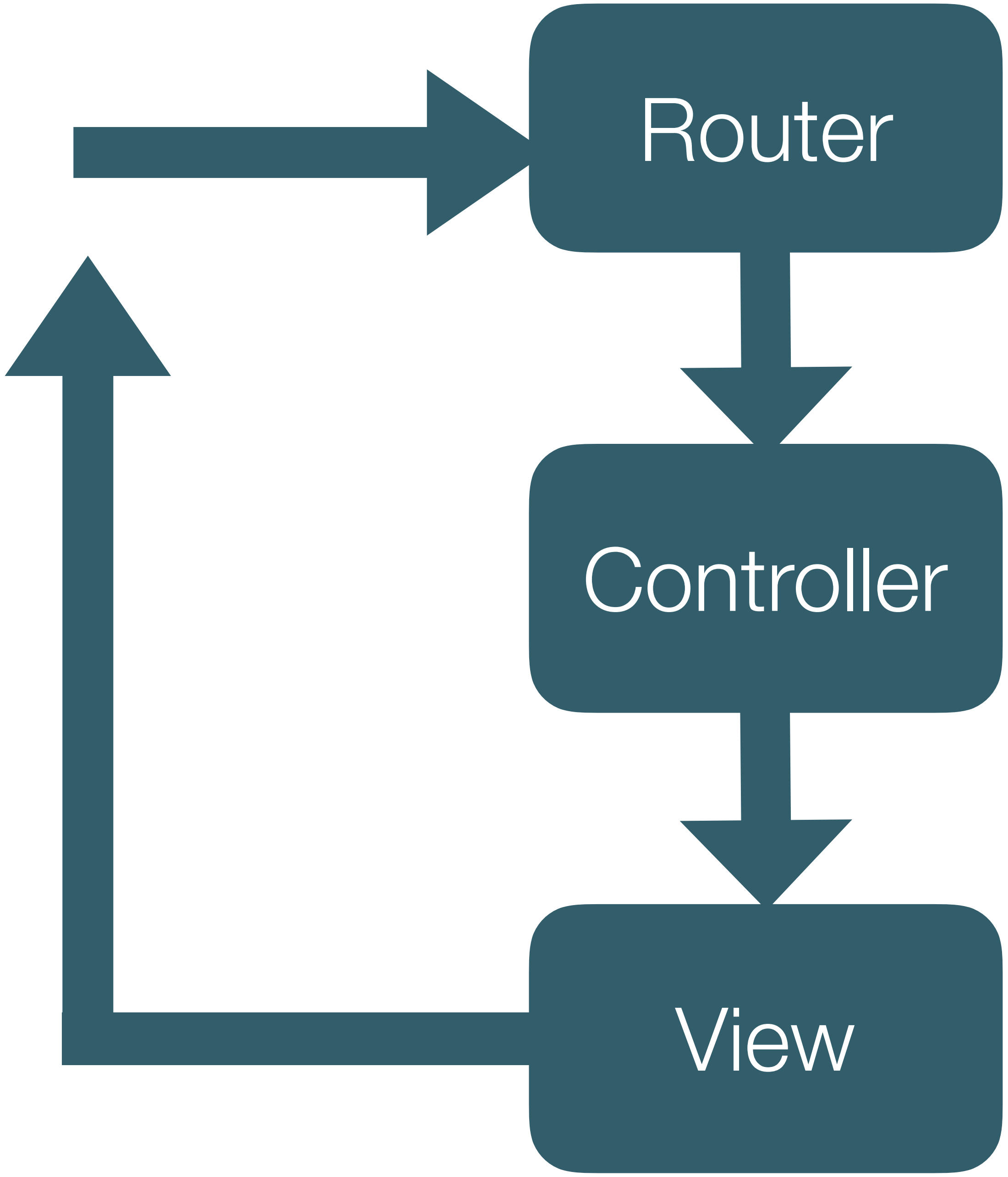


```
const about = {
  index(request, response) {
    logger.info('about rendering');
    const viewData = {
      title: 'About Template 1',
    };
    response.render('about', viewData);
  },
};
```

The About controller



About this app



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

```
const about = {

  index(request, response) {

    logger.info('about rendering’);

    const viewData = {
      title: 'About Template 1',
    };

    response.render('about', viewData);

  },

};
```

The ‘About’ controller index body

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

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

```
const about = {

  index(request, response) {

    logger.info('about rendering’);

    const viewData = {
      title: 'About Template 1',
    };

    response.render('about', 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**)

```
const about = {

  index(request, response) {

    logger.info('about rendering’);

    const viewData = {
      title: 'About Template 1',
    };

    response.render('about', viewData);

  },

};
```

The About Controller - Complete

strict mode javascript for safety

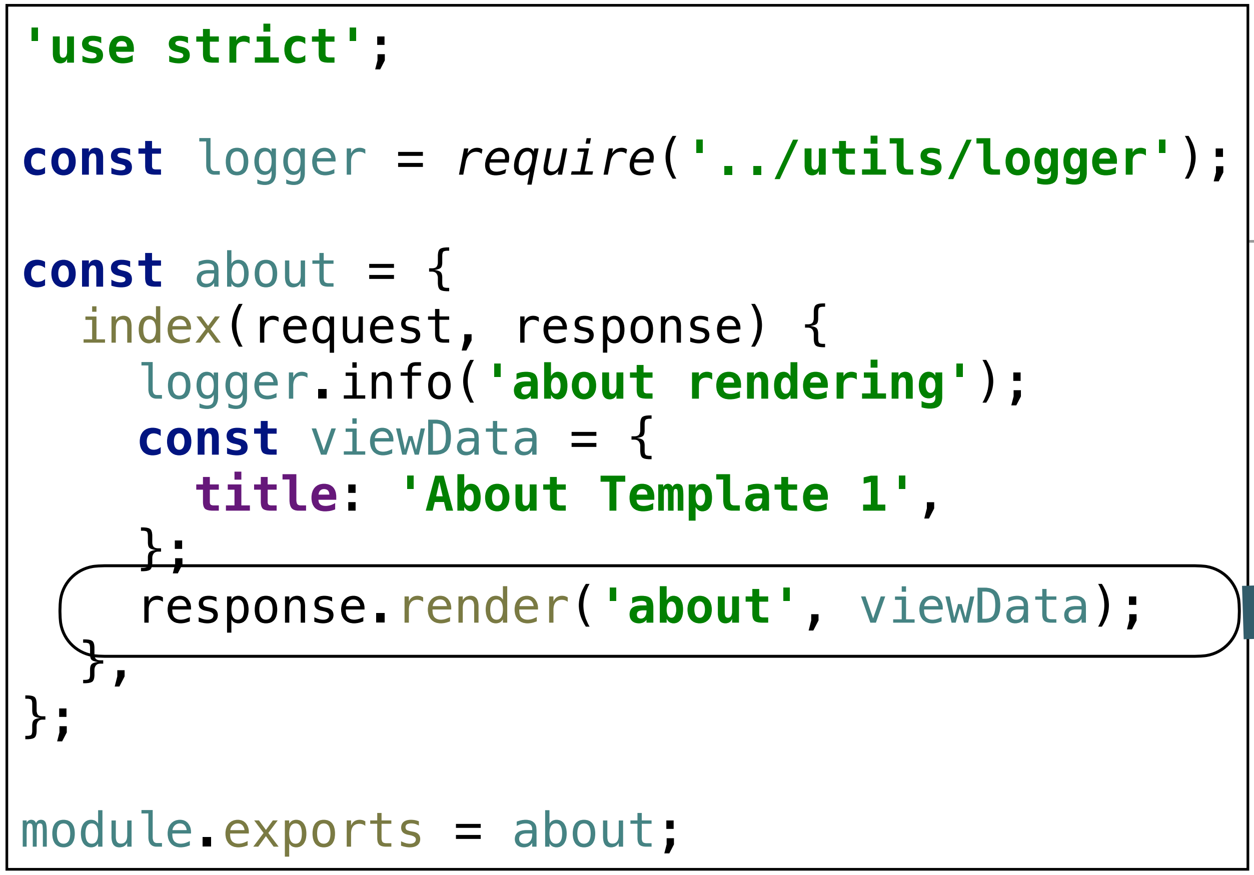
import the **logger** so we can use it

Export the **about** object to it can be used by the router

```
'use strict';

const logger = require('../utils/logger');

const about = {
  index(request, response) {
    logger.info('about rendering');
    const viewData = {
      title: 'About Template 1',
    };
    response.render('about', viewData);
  },
};

module.exports = about;
```

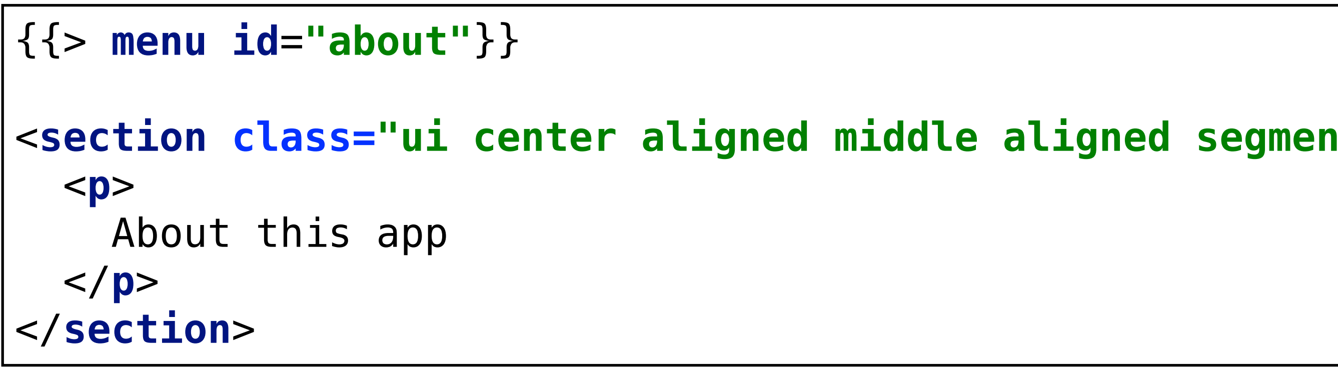
Back-end + Front-End

about.js



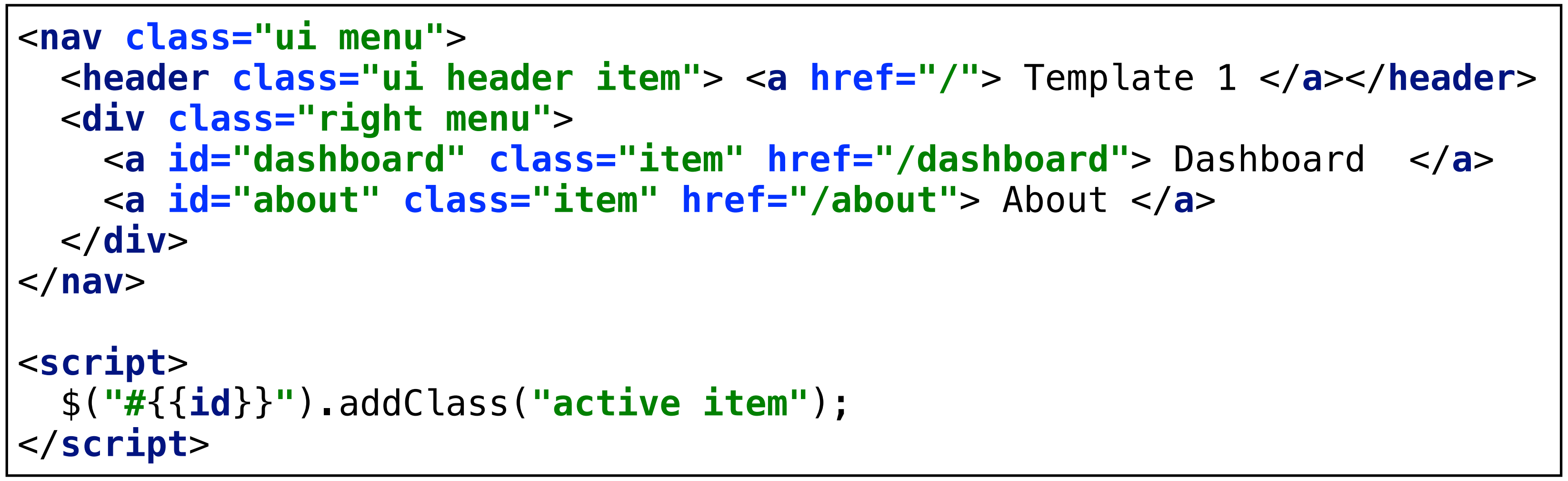
```
'use strict';

const logger = require('../utils/logger');

const about = {
  index(request, response) {
    logger.info('about rendering');
    const viewData = {
      title: 'About Template 1',
    };
    response.render('about', viewData);
  },
};

module.exports = about;
```

about.hbs



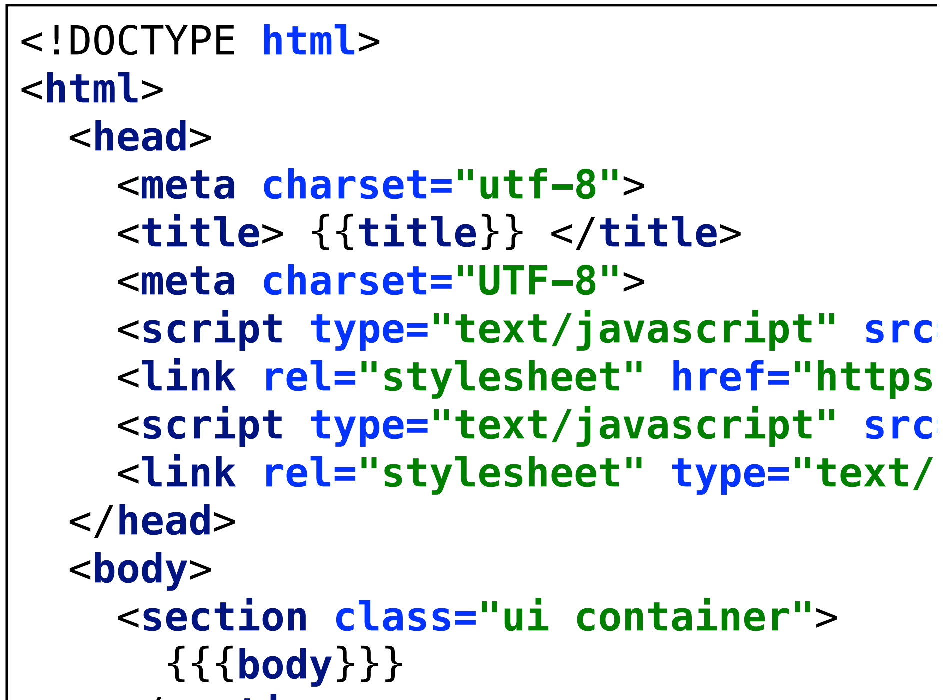
```
{{> menu id="about"}}

<section class="ui center aligned middle aligned segmen
  <p>
    About this app
  </p>
</section>
```

menu.hbs



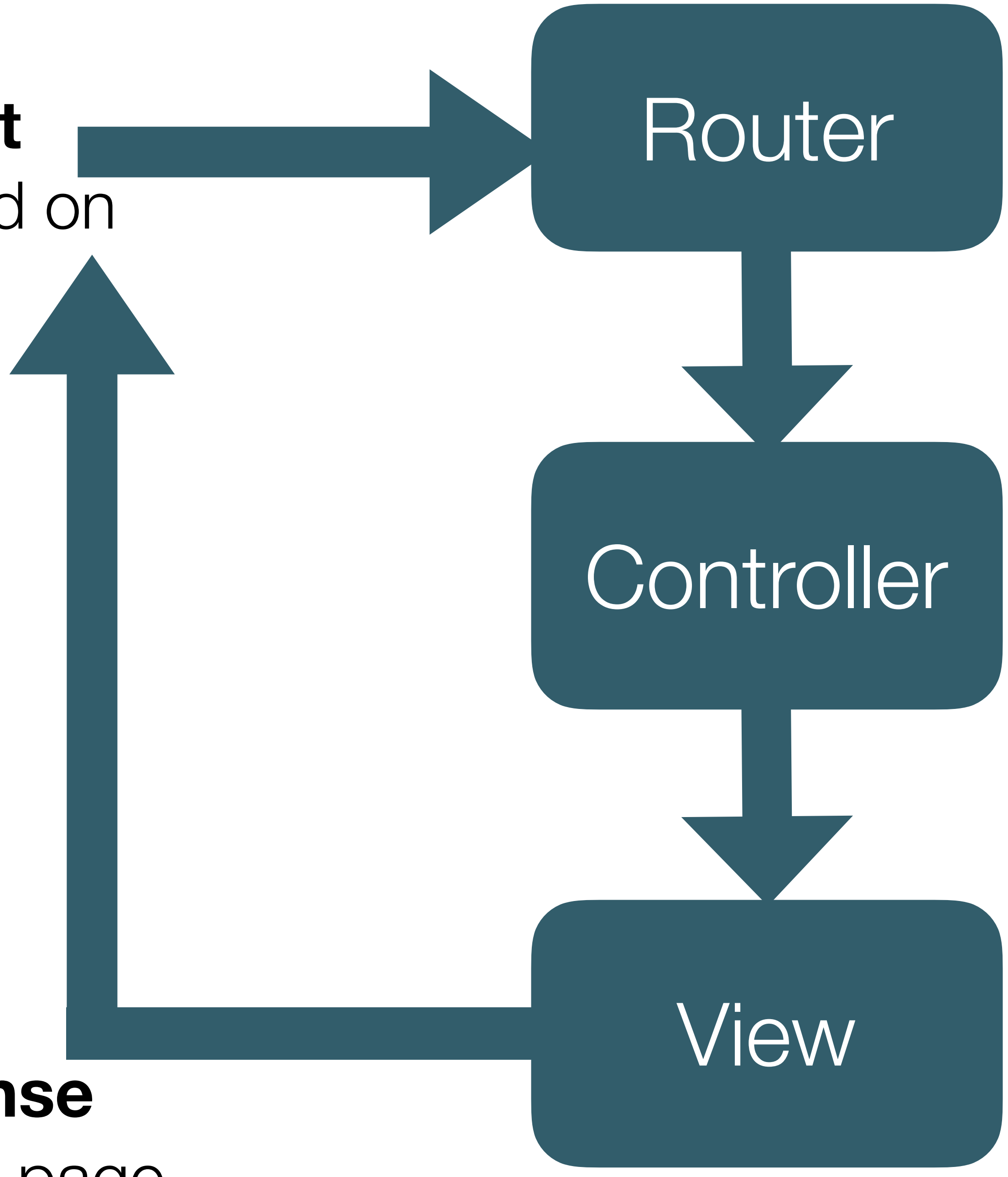
```
<nav class="ui menu">
  <header class="ui header item"> <a href="/"> Template 1 </a></header>
  <div class="right menu">
    <a id="dashboard" class="item" href="/dashboard"> Dashboard  </a>
    <a id="about" class="item" href="/about"> About </a>
  </div>
</nav>

<script>
  $("#{{id}}").addClass("active item");
</script>
```

layout.hbs



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title> {{title}} </title>
    <meta charset="UTF-8">
    <script type="text/javascript" src
    <link rel="stylesheet" href="https
    <script type="text/javascript" src
    <link rel="stylesheet" type="text/
  </head>
  <body>
    <section class="ui container">
      {{{body}}}
```

Router/Controller/View



request
link pressed on
page

find matching
controller
object

method
invoked to
handle request

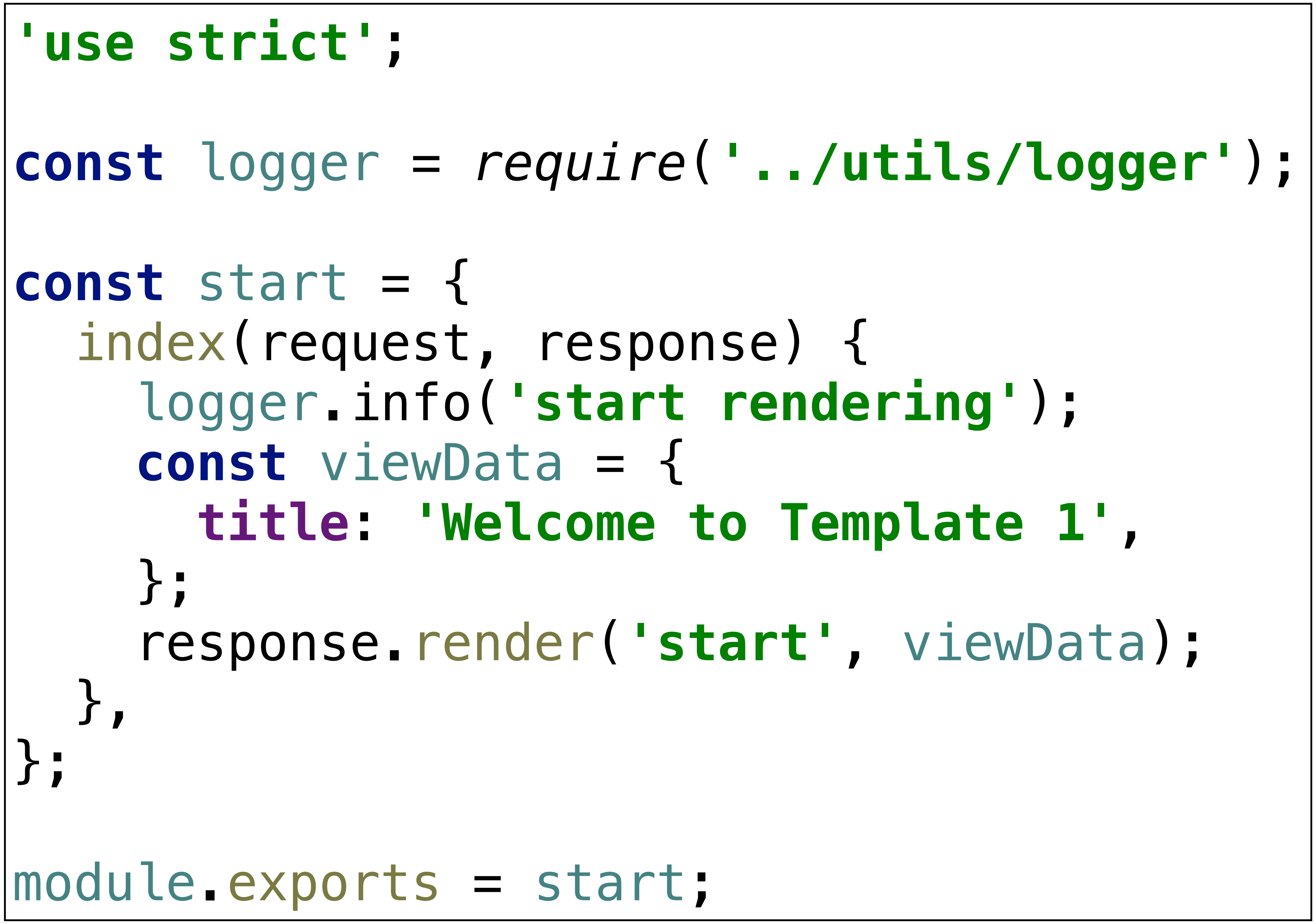
data sent from
controller to view
to construct
response

response
complete page
rendered into
browser

Dashboard Controller

```
'use strict';

const logger = require('../utils/logger');

const dashboard = {
  index(request, response) {
    logger.info('dashboard rendering');
    const viewData = {
      title: 'Template 1 Dashboard',
    };
    response.render('dashboard', viewData);
  },
};

module.exports = dashboard;
```

Start Controller



```
'use strict';

const logger = require('../utils/logger');

const start = {
  index(request, response) {
    logger.info('start rendering');
    const viewData = {
      title: 'Welcome to Template 1',
    };
    response.render('start', viewData);
  },
};

module.exports = start;
```