AGENDA

→ Node

→ Express

→ EJS

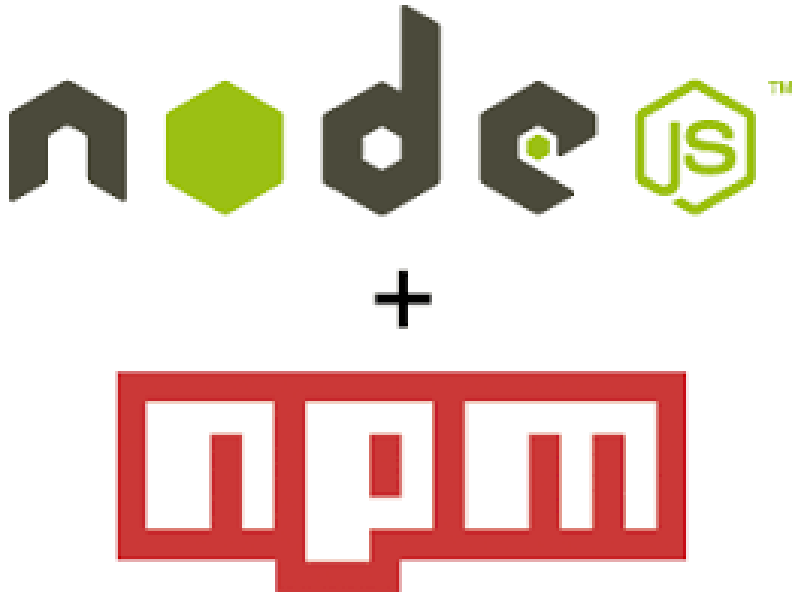
NODE.JS

- → is a runtime enviroment to execute Javascript code outside of a browser
- → quite often used to build all sorts of back-end services, e.g. APIs (even though in CIS 3368 we just use it as the web server)
- → great for prototyping and production ("real-time websites with push capability")
- → comes with a large ecosystem of open-source libraries and tools

Install from https://nodejs.org/en/download/

NPM

- → online repository for the publishing of open-source Node.js project
- → is the default package manager for Node.js (like pip or conda in Python)
- → *package.json* file - is a Node.js project manifest that includes the packages and applications it depends on, and specific metadata like the project's name, description, and author, etc.



Usage:

```
npm install <module> --save
# Where <module> is the name of the module you want to install
npm install
# to install via a package.json file
```

EXPRESS

is a Node.js web application framework that allows devevelpers to:

- → route HTTP requests
- → serve up HTML statically or dynamically

Install via:

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

EXPRESS

is a Node.js web application framework that allows devevelpers to:

- → route HTTP requests
- → serve up HTML statically or dynamically

Install via:

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

EJS TEMPLATING ENGINE

Static HTML are too: static

Most webapges are dynamic - they change based on data

Templates allow developers to create structure, layout and style for pages without specific data - the data is dynamic

- → EJS - Embedded JavaScript lets devlopers write Javascript directly in HTML templates
- → the server renders these templates by *passing in data* in JSON from
- → The templates have *placeholders* and *logic* to change the page based on data

Install via:

command line

```
$ npm install ejs --save
```

CLIENT SIDE RENDERING

```
response.render('page', {
    data: jsonObject
});
```

- **page.ejs** is an HTML template file
- **jsonObject** contains any JSON data
- **data** will be usable within the page.ejs template

CLIENT SIDE RENDERING

```
response.render('page', {
    data: jsonObject
});
```

- **page.ejs** is an HTML template file
- **jsonObject** contains any JSON data
- **data** will be usable within the page.ejs template

CLIENT SIDE RENDERED VALUES

- In an **ejs** file, EJS segments can use the data passed on the server
- JavaScript is rendered within <%= … %>

page.ejs

```
<p>Name: <%= data.name %></p>
<p>Age: <%= data.age %></p>
```

data

```
{ name: 'Al', age: 2 }
```

Rendered output

```
<p>Name: Al</p>
<p>Age: 2</p>
```

CLIENT SIDE SCRIPTLETS

For control flow

- EJS segments can also contain **logic** to change the rendered output
- These special EJS *scriptlets* use **<%** … **%>** (no equals sign)
- They use JavaScript to *conditionally* or *repeatedly* display HTML
- They do not directly output anything to the rendered output

EJS IF ELSE EXAMPLE

page.ejs

```
<% if (data.age >= 16) { %>
    <p>You can drive</p>
<% } %>
```

data (1)

```
{ name: 'Al', age: 2 }
```

Rendered output (1)

Nothing!

data (2)

```
{ name: 'Sam', age: 22 }
```

Rendered output (2)

```
<p>You can drive</p>
```

EJS FOR LOOP EXAMPLE

page.ejs

```
<% for (let i = 0; i < 3; i++) { %>
    <p>Number <%= i %></p>
<% } %>
```

Rendered output

```
<p>Number 0</p>
<p>Number 1</p>
<p>Number 2</p>
```

Let's build a Express Node.js app that uses EJS.

We will use the following structure in our client-side app:

```
- views
----- partials
---------- footer.ejs
---------- head.ejs
---------- header.ejs
----- pages
---------- index.ejs
---------- about.ejs
- package.json
- server.js
```