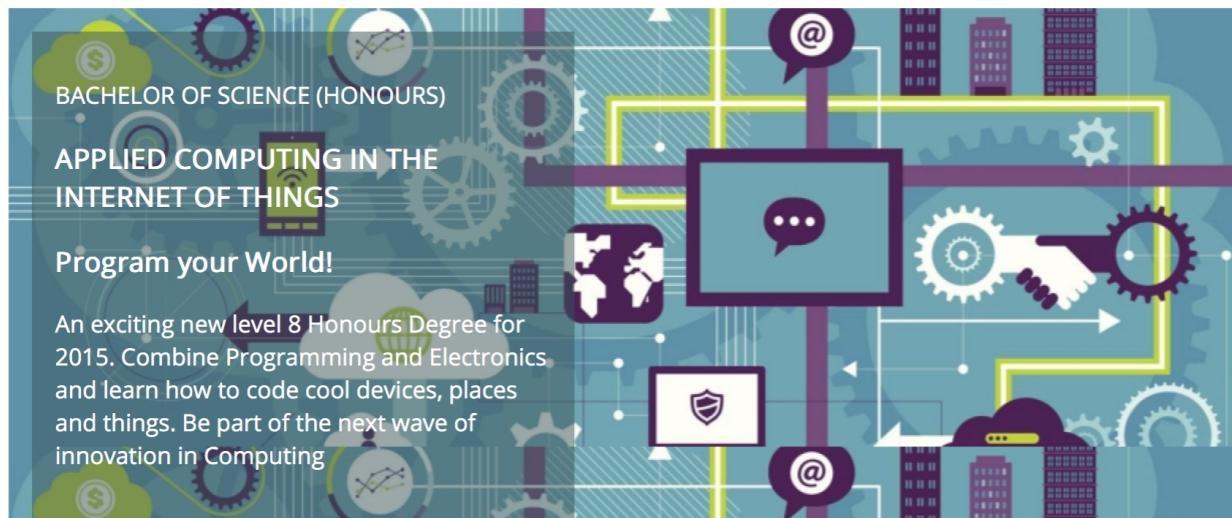


Web Development

Templates



Programming

Learn a broad range of programming and problem solving skills, including exciting new platforms, software tools and languages. Use these skills to build apps for mobile, cloud and device based IoT applications. Evolve a portfolio of fascinating applications.

Data Science

At the heart of many IoT applications is data: measurements, events alarms and other information that must be relayed, stored and ultimately turned into knowledge. Learn the fundamentals of modern approaches to data in this strand.

Devices

The 'Things' we connect to are often physical devices. These can range from simple temperature sensors to sophisticated control systems like traffic lights or cameras. Connecting to and interacting with the physical world is the subject of this strand.

Networks

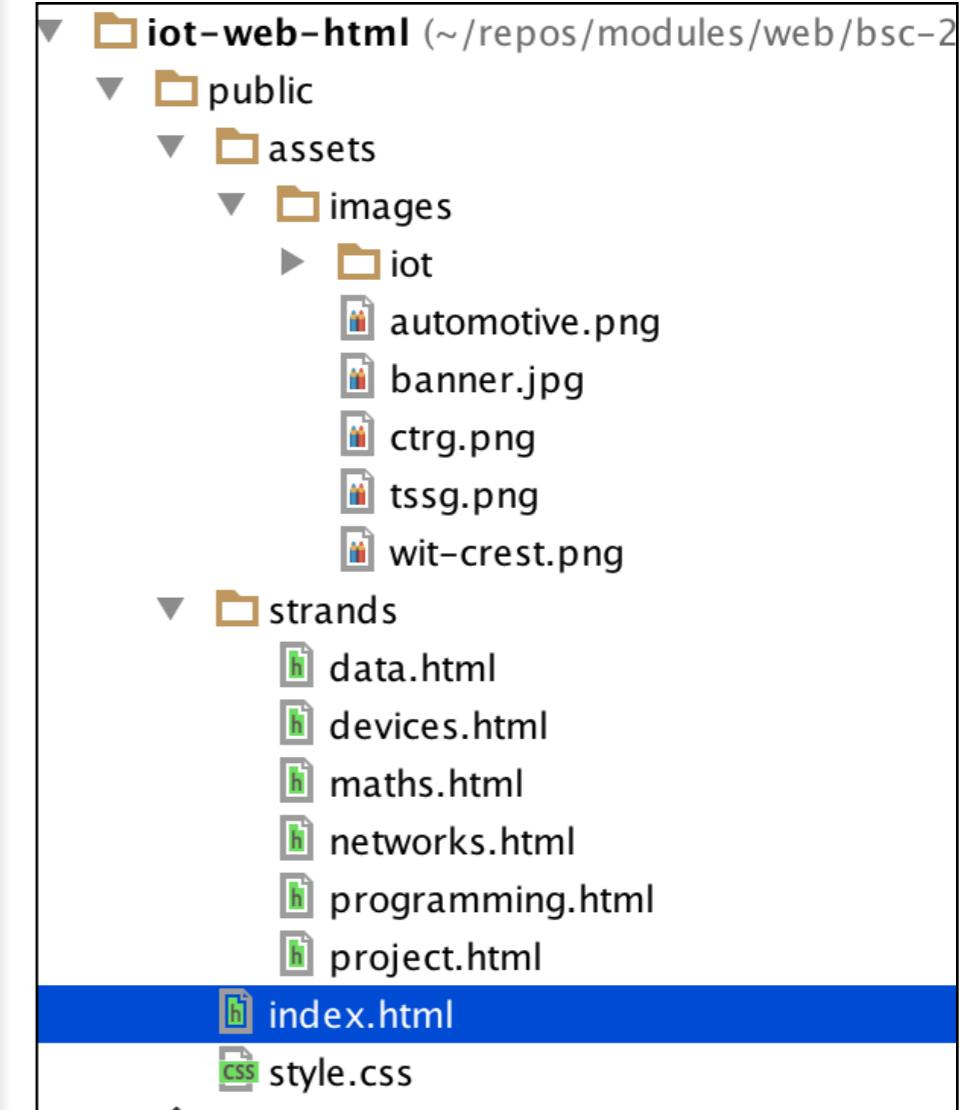
This strand will explore modern networks and cloud technology. Be able to configure, network and manage all categories of computer systems from simple controllers to single board computers, mobiles and full workstations.

Project

Building exciting IoT projects in every semester of the programme. Your projects will combine skills acquired from the other strands and enable you to build a comprehensive and compelling portfolio of IoT applications and services.

Mathematics

Introduce foundation concepts for many of the more applied concepts in the other Strands. Learn mathematical techniques in a modern context and apply core principles in new and interesting ways.



Supported by leading edge research at...





Programming

Learn a broad range of programming and problem solving skills, including exciting new platforms, software tools and languages. Use these skills to build apps for mobile, cloud and device based IoT applications. Evolve a portfolio of fascinating applications.

Data Science

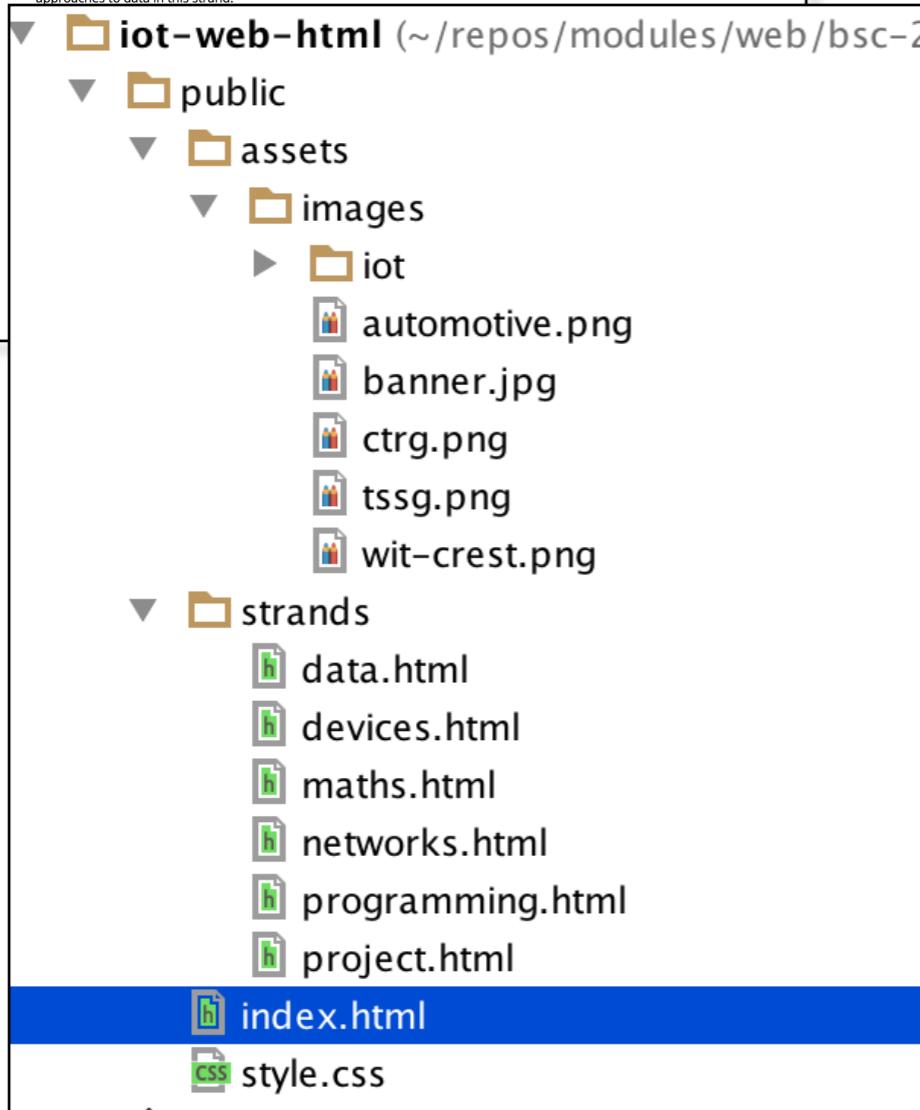
At the heart of many IoT applications is data: measurements, events alarms and other information that must be relayed, stored and ultimately turned into knowledge. Learn the fundamentals of modern approaches to data in this strand.

Networks

This strand will explore modern networks and cloud technology. Be able to configure, network and manage all categories of computer systems from simple controllers to single board computers, mobiles and full workstations.

Project

Building exciting IoT projects in every semester of the programme. Your projects will combine skills acquired from the other strands and enable you to build a comprehensive and compelling portfolio of IoT applications and services.



- This web site has 7 pages.
- Each page has:
 - Head Section
 - Body Section
- Each Body Section has
 - Header
 - Footer
- —>
 - 7 Identical Head Section
 - 7 Identical Header's
 - 7 Identical Footer's
- —>

Templates Why? —

- Its got its own Wikipedia Page!

Don't repeat yourself

From Wikipedia, the free encyclopedia

In software engineering, **don't repeat yourself (DRY)** is a principle of software development, aimed at reducing repetition of information of all kinds, especially useful in multi-tier architectures. The DRY principle is stated as "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system." The principle has been formulated by [Andy Hunt](#) and [Dave Thomas](#) in their book *The Pragmatic Programmer*, coauthored with [Dennis Ritchie](#) and [Francisco Granados](#). They apply it quite broadly to include "database schemas, test plans, the build system, even documentation."^[1] When the DRY principle is applied successfully, a modification of any single element of a system does not require a change in other logically unrelated elements. Additionally, elements that are logically related all change predictably and uniformly, and are thus kept in sync. Besides using methods and subroutines in their code, Thomas and Hunt rely on [code generators](#), automatic build systems, and scripting languages to observe the DRY principle across layers.

Contents [hide]

- [1 DRY vs WET solutions](#)
- [2 See also](#)
- [3 References](#)
- [4 External links](#)

DRY vs WET solutions [edit]

Violations of DRY are typically referred to as WET solutions, which is commonly taken to stand for either "write everything twice" or "we enjoy typing".^{[2][3]}

https://en.wikipedia.org/wiki/Don%27t_repeat_yourself

DRY vs WET

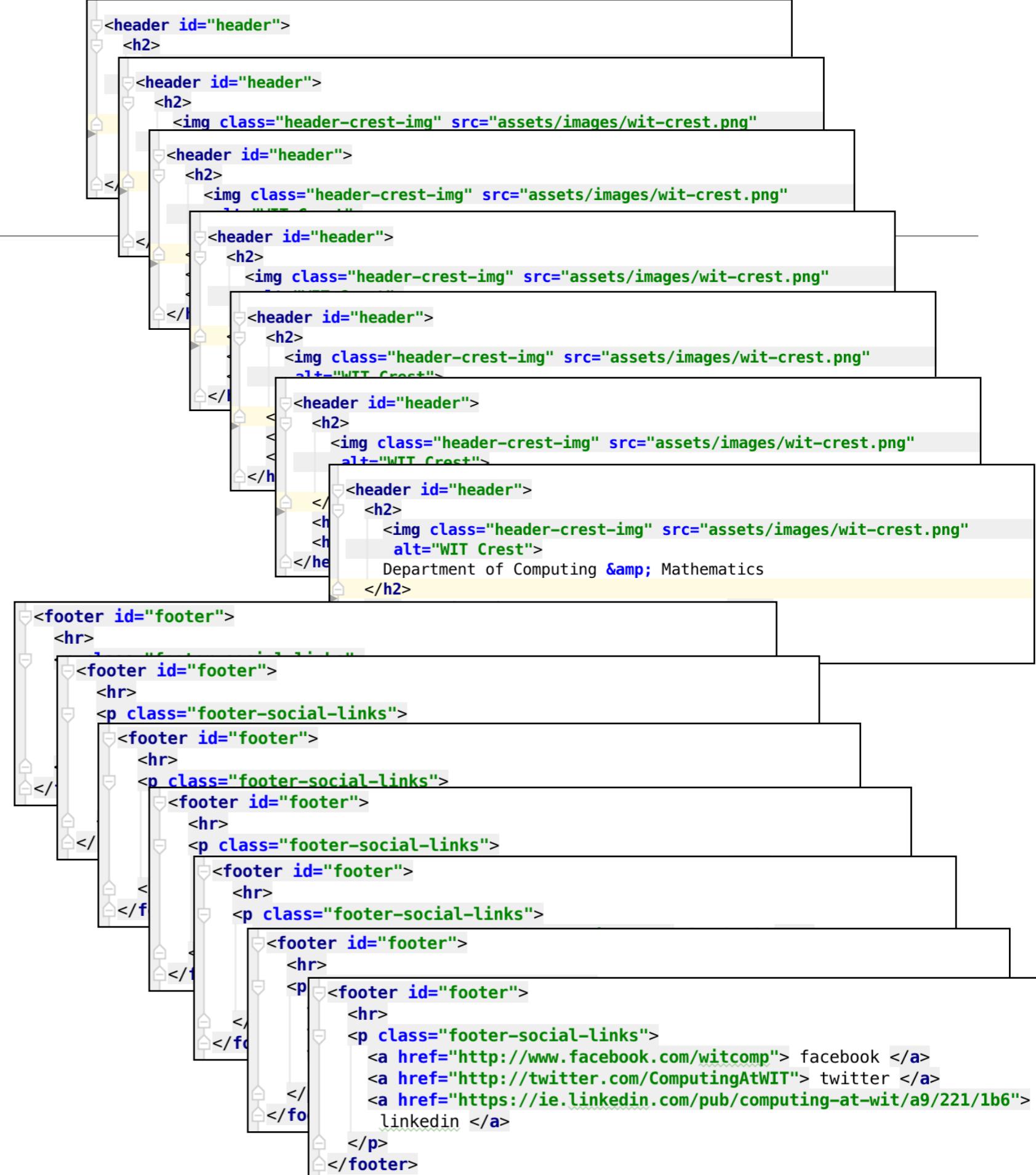
Don't Repeat Yourself

VS

Write Everything Twice

OR

We Enjoy Typing



Single Header + Footer Template

DRY

- Incorporate the SAME single header/footer into ALL pages

```
<header id="header">
  <h2>
    
    Department of Computing & Mathematics
  </h2>
  <h3> BSc (Hons) the Internet of Things </h3>
  <hr>
</header>
```

- Any changes - made just once in the single header/footer

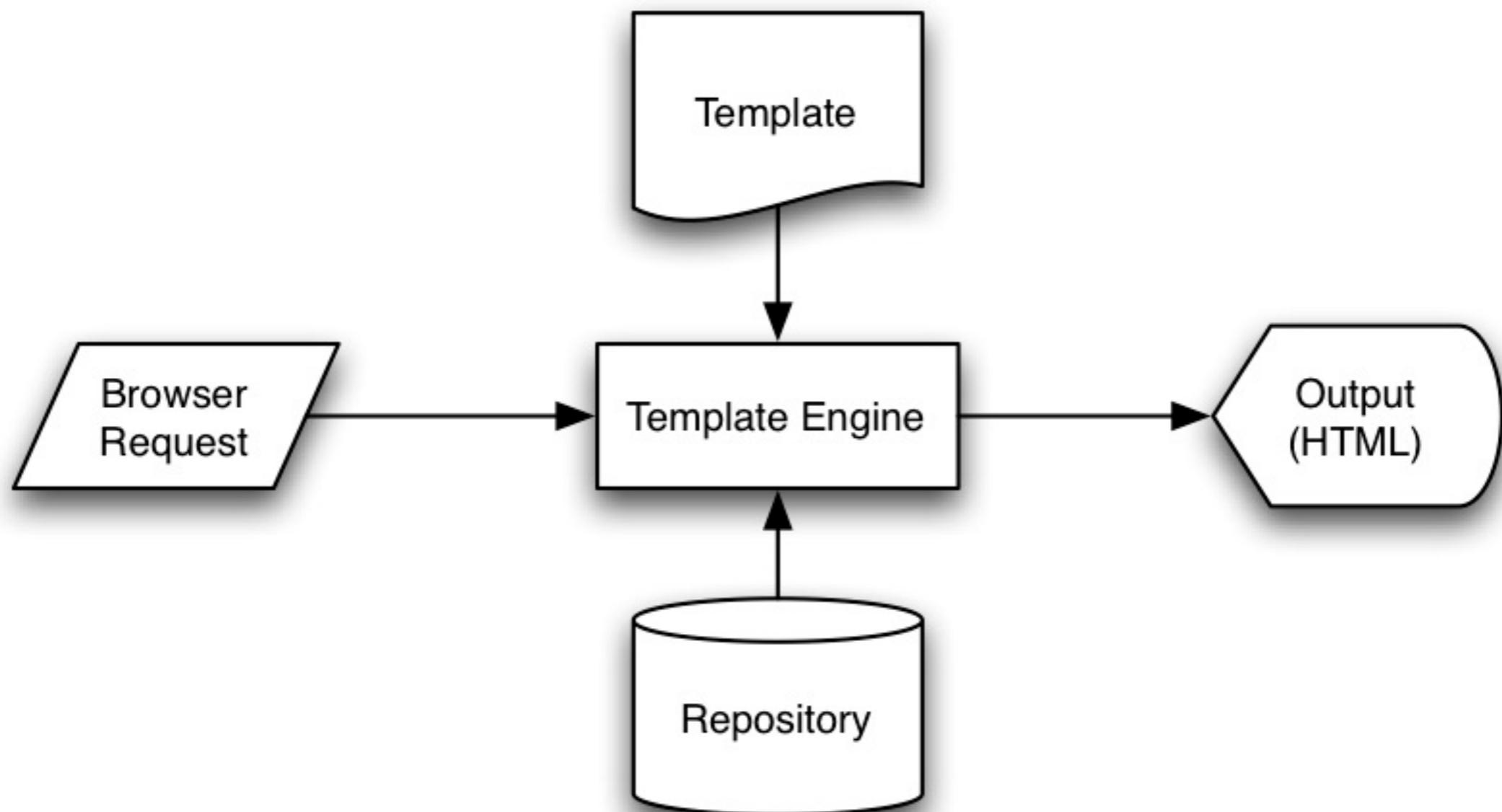
```
<footer id="footer">
  <hr>
  <p class="footer-social-links">
    <a href="http://www.facebook.com/witcomp"> facebook </a>
    <a href="http://twitter.com/ComputingAtWIT"> twitter </a>
    <a href="https://ie.linkedin.com/pub/computing-at-wit/a9/221/1b6">
      linkedin </a>
  </p>
</footer>
```

A large, solid dark teal arrow pointing to the right.

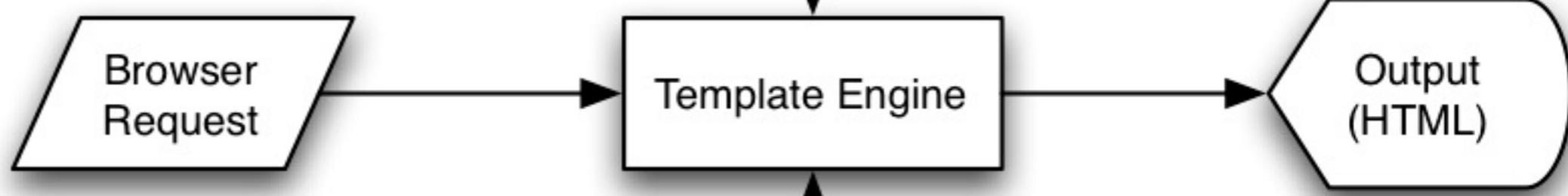
Web Template System

A web template system uses a template processor to combine web templates to form finished web pages, possibly using some data source to customize the pages or present a large amount of content on similar-looking pages. It is a web publishing tool present in content management systems, web application frameworks, and HTML editors.

https://en.wikipedia.org/wiki/Web_template_system



Harp.js



- Harp.js is our Template Engine
- It ‘serves’ the site
- If *Request* is for ordinary page the page is ‘rendered’ without modification
- If *Request* is for a page that is composed of templates, harp assembles the page and renders the complete page to the browser

harp Documentation

The static web server with built-in preprocessing.

Harp serves Jade, Markdown, EJS, CoffeeScript, Sass, LESS and Stylus as HTML, CSS & JavaScript—no configuration necessary.

[Follow @HarpWebServer](#) [Star Harp on GitHub](#)

Lab05b

HTML Templates

Lab-5b Templating

01

02

03

04

05

06

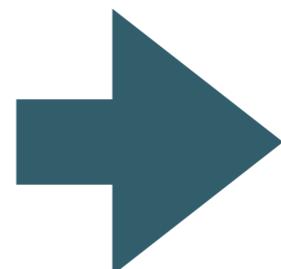
Exercises



Objectives

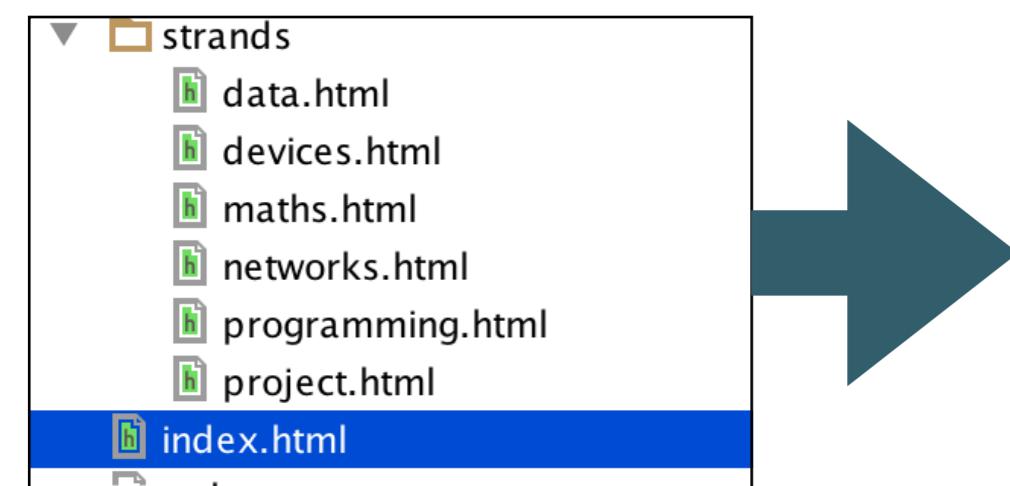
Rebuild the IoT web site from thee last lab using templating. This version of the site will aim to significantly reduce the content the author has to manage by reusing 'templates' containing common sections.

```
iot-web-html (~/repos/modules/web/bsc-2)
  public
    assets
      images
        iot
          automotive.png
          banner.jpg
          ctrg.png
          tssg.png
          wit-crest.png
    strands
      data.html
      devices.html
      maths.html
      networks.html
      programming.html
      project.html
    index.html
    style.css
```

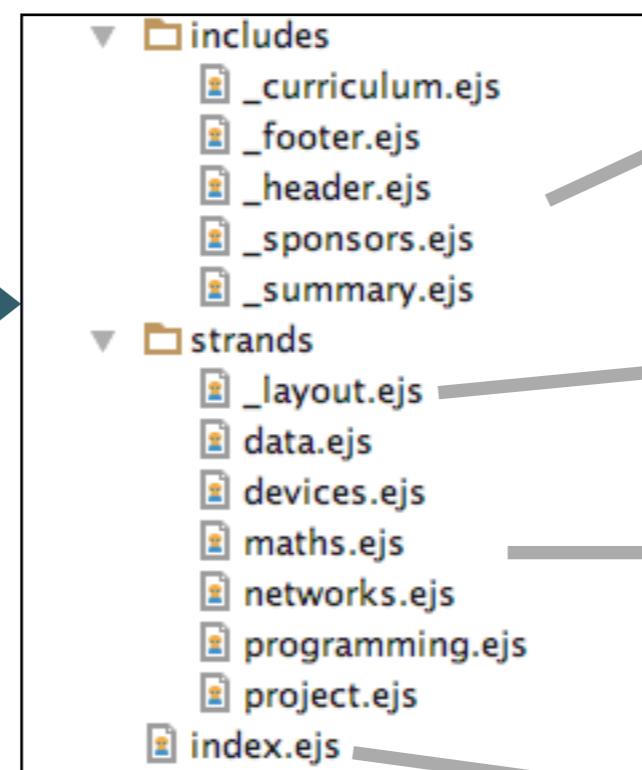


```
iot-web-ejs (~/repos/modules/web/bsc-2)
  public
    assets
      images
        iot
          automotive.png
          banner.jpg
          ctrg.png
          tssg.png
          wit-crest.png
  includes
    _header.ejs
    _footer.ejs
    _curriculum.ejs
    _sponsors.ejs
    _summary.ejs
  strands
    _layout.ejs
    data.ejs
    devices.ejs
    maths.ejs
    networks.ejs
    programming.ejs
    project.ejs
  index.ejs
  style.css
```

Lab05b



WET Version



DRY Version

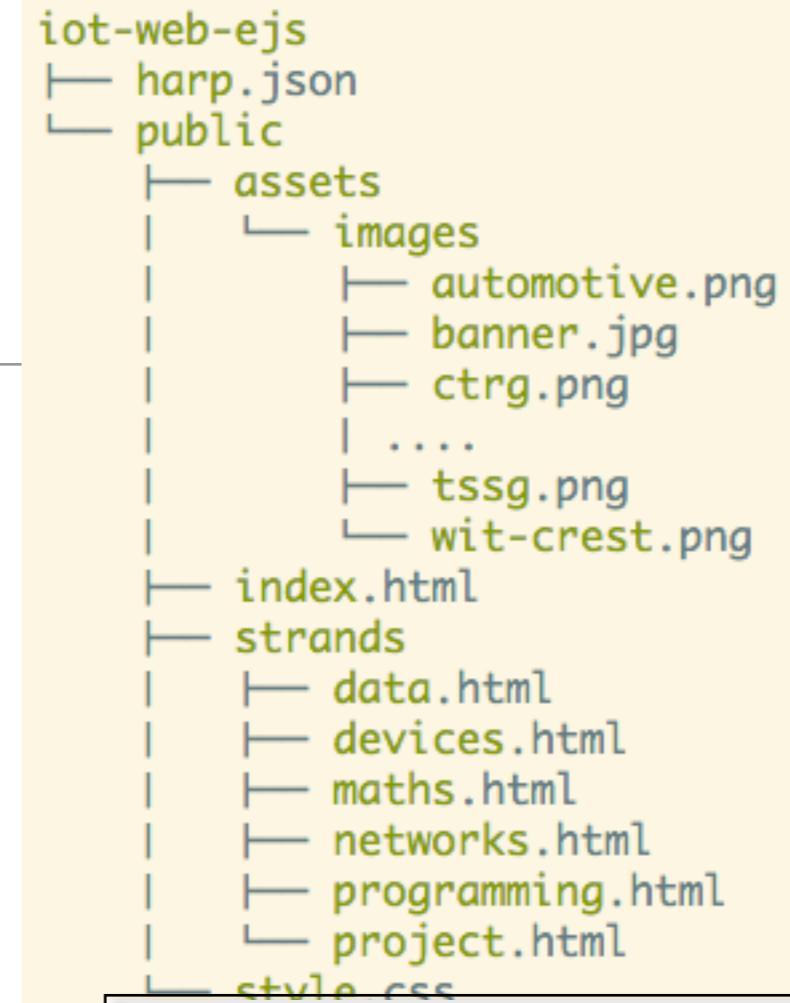
- reusable templates included in various pages
- reusable layout
- reworked pages based on layout
- simplified home page

- Overall - more files
- But less content!

Step 1

```
G:\> cd iot-web-ejs  
G:\iot-web-ejs> harp server  
Your server is listening at http://localhost:9000/  
Press Ctrl+C to stop the server
```

- Visit:
 - <http://localhost:9000/>
- WET (non templated) version of site



The screenshot shows the homepage of the Waterford Institute of Technology Bachelor of Science (Honours) in Applied Computing in the Internet of Things program. The page features a banner with the title 'APPLIED COMPUTING IN THE INTERNET OF THINGS' and 'Program your World!'. Below the banner, there is descriptive text about the degree, mentioning it's a Level 8 Honours Degree for 2015, combining Programming and Electronics, and encouraging students to learn how to code cool devices, places, and things. The page is divided into several sections: 'Programming', 'Networks', 'Data Science', 'Devices', and 'Mathematics'. Each section has a brief description and a link to more information. At the bottom, there is a footer with logos for TSSG, ctrg, and AUTOMOTIVE CONTROL GROUP, along with links to social media platforms.

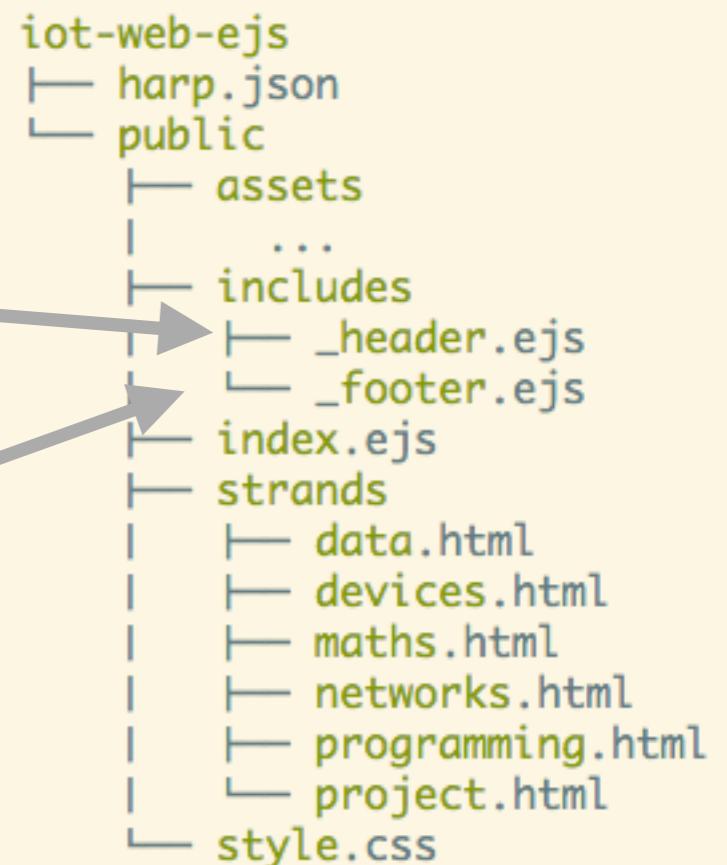
Step 02 - Header & Footer templates

_header.ejs

```
<header id="header">
  <h2>
    
    Department of Computing & Mathematics
  </h2>
  <h3> BSc (Hons) the Internet of Things </h3>
  <hr>
</header>
```

_footer.ejs

```
<footer id="footer">
  <hr>
  <p class="footer-social-links">
    <a href="http://www.facebook.com/witcomp"> facebook </a>
    <a href="http://twitter.com/ComputingAtWIT"> twitter </a>
    <a href="https://ie.linkedin.com/pub/computing-at-wit/a9/221/1b6"> linkedin </a>
  </p>
</footer>
```



- New folder in project called ‘includes’
- ... containing reusable templates ‘_header.ejs’ & ‘_footer.ejs’
- These are exactly the same content as in all our other pages

Step 02: index.html

- Replace the <header> and <footer> elements with :

```
...
<%- partial("includes/_header.ejs") %>
...
<%- partial("includes/_footer.ejs") %>
...
```

- These will be ‘included’ in the page when it is rendered via harp.
- However, if the page loaded directly from disk page will not be rendered correctly:

```
<%- partial("includes/_header.ejs") %>
```



Step 03: Resource Paths

header.ejs

```
<header id="header">
  <h2>
    
    Department of Computing & Mathematics
  </h2>
  <h3> BSc (Hons) the Internet of Things </h3>
  <hr>
</header>
```

- The ‘src’ link in the image is relative - it assumes the ‘assets’ path is in the current folder
- This may not always be the case
- Change this to an ‘absolute’ path:

```

```

- This will enable the template to be included in any file, regardless of where the file is in the site structure

Step 03: Relative vs Absolute

header.ejs

```
<header id="header">
  <h2>
    
    Department of Computing & Mathematics
  </h2>
  <h3> BSc (Hons) the Internet of Things </h3>
  <hr>
</header>
```

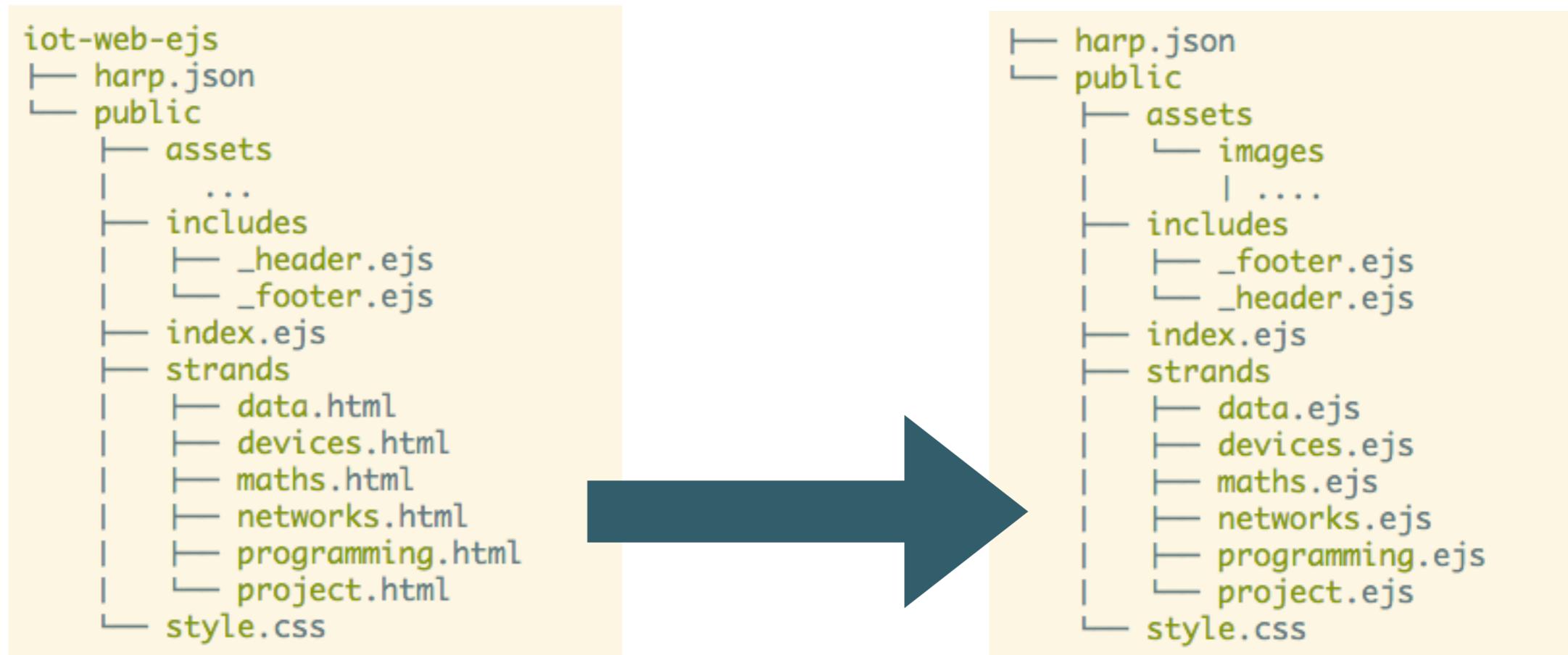
- Harp server will make sure correct image server on:
 - http://localhost:9000/

```
C:\My Documents> G:
G:\> node\init
G:\> cd iot-web-ejs
G:\iot-web-ejs> harp server
Your server is listening at http://localhost:9000/
Press Ctrl+C to stop the server
```

```

```

Step 04: Rename Files



- Rename all “.html” files to “.ejs”
- This instructs harp to process these files, incorporating template features as necessary

Step 04:

- Delete <header> & <footer> form all pages

- Replace with

```
<%- partial("../includes/_header.ejs") %>
```

```
<%- partial("../includes/_footer.ejs") %>
```

- DRY first steps...

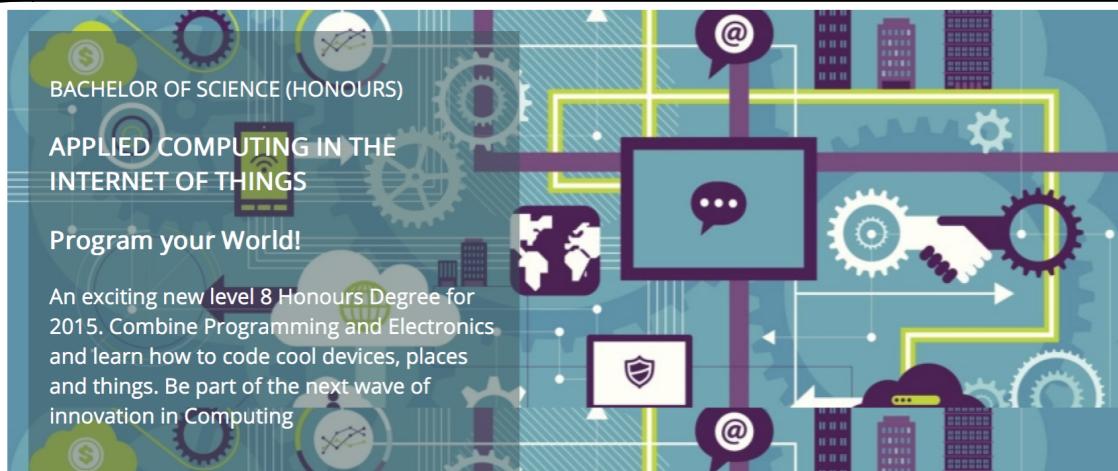
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/
  <link type="text/css" rel="stylesheet" href="../style.css" media="screen
  <title> Devices </title>
</head>
<body>

<%- partial("../includes/_header.ejs") %>

<article>
  <h1> Devices </h1>
  <p>
    
  
</article>

<%- partial("../includes/_footer.ejs") %>

</body>
</html>
```



Programming

Learn a broad range of programming and problem solving skills, including exciting new platforms, software tools and languages. Use these skills to build apps for mobile, cloud and device based IoT applications. Evolve a portfolio of fascinating applications.

Data Science

At the heart of many IoT applications is data: measurements, events alarms and other information that must be relayed, stored and ultimately turned into knowledge. Learn the fundamentals of modern approaches to data in this strand.

Devices

The 'Things' we connect to are often physical devices. These can range from simple temperature sensors to sophisticated control systems like traffic lights or cameras. Connecting to and interacting with the physical world is the subject of this strand.

Networks

This strand will explore modern networks and cloud technology. Be able to configure, network and manage all categories of computer systems from simple controllers to single board computers, mobiles and full workstations.

Project

Building exciting IoT projects in every semester of the programme. Your projects will combine skills acquired from the other strands and enable you to build a comprehensive and compelling portfolio of IoT applications and services.

Mathematics

Introduce foundation concepts for many of the more applied concepts in the other Strands. Learn mathematical techniques in a modern context and apply core principles in new and interesting ways.

Supported by leading edge research at...



[facebook](#) [twitter](#) [linkedin](#)

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head...>
```

```
</body>
```

```
<header id="header">
```

```
<h2>
```

```
    
```

```
    Department of Computing & Mathematics
```

```
</h2>
```

```
<h3> BSc (Hons) the Internet of Things </h3>
```

```
<hr>
```

```
</header>
```

```
<article class="banner">
```

```
<div id="summary">
```

```
<p>
```

```
    BACHELOR OF SCIENCE (HONOURS)
```

```
</p>
```

```
<h3>
```

```
    APPLIED COMPUTING IN THE INTERNET OF THINGS
```

```
</h3>
```

```
<h3>
```

```
    Program your World!
```

```
</h3>
```

```
<p>
```

```
    An exciting new level 8 Honours Degree for 2015. Combine Programming and Electronics and learn how to code cool devices, places and things. Be part of the next wave of innovation in Computing
```

```
</p>
```

```
</div>
```

```
</article>
```

```
<article id="curriculum"...>
```

```
<section id="sponsors">
```

```
<hr>
```

```
<h4> Supported by leading edge research at... </h4>
```

```
<p>
```

```
    
```

```
    
```

```
    
```

```
</p>
```

```
</section>
```

```
<footer id="footer">
```

```
<hr>
```

```
<p class="footer-social-links">
```

```
    <a href="http://www.facebook.com/witcomp"> facebook </a>
```

```
    <a href="http://twitter.com/ComputingAtWIT"> twitter </a>
```

```
    <a href="https://ie.linkedin.com/pub/computing-at-wit/a9/221/1b6">
```

```
        linkedin </a>
```

```
</p>
```

```
</footer>
```

```
</body>
```

```
</html>
```

Step 05:

```
└── harp.json  
└── public  
    └── assets  
        └── images  
            | ....  
    └── includes  
        └── _curriculum.ejs  
        └── _footer.ejs  
        └── _header.ejs  
        └── _sponsors.ejs  
        └── _summary.ejs  
    └── index.ejs  
    └── strands  
        └── data.ejs  
        └── devices.ejs  
        └── maths.ejs  
        └── networks.ejs  
        └── programming.ejs  
        └── project.ejs  
    └── style.css
```

```
<!DOCTYPE html>  
<html lang="en">  
  <head...>  
  <body>  
  
    <header id="header">  
      <h2>  
          
        Department of Computing & Mathematics  
      </h2>  
      <h3> BSc (Hons) the Internet of Things </h3>  
      <hr>  
    </header>  
  
    <article class="banner">  
      <div id="summary">  
        <p> BACHELOR OF SCIENCE (HONOURS)  
        </p>  
  
        <h3> APPLIED COMPUTING IN THE INTERNET OF THINGS  
        </h3>  
  
        <h3> Program your World!  
        </h3>  
        <p> An exciting new level 8 Honours Degree for 2015. Combine Programming and Electronics and learn how to code cool devices, places and things. Be part of the next wave of innovation in Computing  
        </p>  
      </div>  
    </article>  
  
    <article id="curriculum" ...>  
  
    <section id="sponsors">  
      <hr>  
      <h4> Supported by leading edge research at... </h4>  
      <p>  
          
          
          
      </p>  
    </section>  
  
    <footer id="footer">  
      <hr>  
      <p class="footer-social-links">  
        <a href="http://www.facebook.com/witcomp"> facebook </a>  
        <a href="http://twitter.com/ComputingAtWIT"> twitter </a>  
        <a href="https://ie.linkedin.com/pub/computing-at-wit/a9/221/1b6">  
          linkedin </a>  
        </p>  
    </footer>  
  
  </body>  
</html>
```

- ‘Factor out’ sections of the index.html pages into includes...

Step 05: index.html

index.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Open+Sans" />
  <link type="text/css" rel="stylesheet" href="style.css" media="screen"/>
  <title>BSc in the Internet of Things</title>
</head>
<body>

<%- partial("includes/_header.ejs") %>
<%- partial("includes/_summary.ejs") %>
<%- partial("includes/_curriculum.ejs") %>
<%- partial("includes/_sponsors.ejs") %>
<%- partial("includes/_footer.ejs") %>

</body>
</html>
```

- Simplified significantly
- All of the design implemented in the includes

Step 05: summary & sponsors

summary.ejs

```
<article class="banner">
  <div id="summary">
    <p>
      BACHELOR OF SCIENCE (HONOURS)
    </p>

    <h3>
      APPLIED COMPUTING IN THE INTERNET OF THINGS
    </h3>

    <h3>
      Program your World!
    </h3>
    <p>
      An exciting new level 8 Honours Degree for 2015. Combine Programming and Electronics and learn how to code cool
    </p>
  </div>
</article>
```

sponsors.ejs

```
<section id="sponsors">
  <hr>
  <h4> Supported by leading edge research at... </h4>
  <p>
    
    
    
  </p>
</section>
```

Step 05: curriculum

curriculum.ejs

```
<article id="curriculum">
  <hr>
  <section id="col1">
    <h2><a href="strands/programming.html"> Programming </a></h2>
    <p>
      Learn a broad range of programming and problem solving skills, including exciting new platforms, software tools
    </p>

    <h2><a href="strands/data.html"> Data Science </a></h2>
    <p>
      At the heart of many IoT applications is data: measurements, events alarms and other information that must be re
    </p>
    <h2><a href="strands/devices.html"> Devices </a></h2>
    <p>
      The 'Things' we connect to are often physical devices. These can range from simple temperature sensors to sophis
    </p>
  </section>
  <section id="col2">
    <h2><a href="strands/networks.html"> Networks </a></h2>
    <p>
      This strand will explore modern networks and cloud technology. Be able to configure, network and manage all cate
    </p>
    <h2><a href="strands/project.html"> Project </a></h2>
    <p>
      Building exciting IoT projects in every semester of the programme. Your projects will combine skills acquired fr
    </p>

    <h2><a href="strands/mathematics.html"> Mathematics </a></h2>
    <p>
      Introduce foundation concepts for many of the more applied concepts in the other Strands. Learn mathematical tec
    </p>
  </section>
</article>
```

Step

```
└── harp.json
└── public
    ├── assets
    │   └── images
    │       └── ....
    ├── includes
    │   ├── _curriculum.ejs
    │   ├── _footer.ejs
    │   ├── _header.ejs
    │   ├── _sponsors.ejs
    │   └── _summary.ejs
    ├── index.ejs
    └── strands
        ├── data.ejs
        ├── devices.ejs
        ├── maths.ejs
        ├── networks.ejs
        ├── programming
        │   └── project.ejs
        └── style.css
```

- harp will now compose the page from 5 templates

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="h
  <link type="text/css" rel="stylesheet" href="s
  <title>BSc in the Internet of Things</title>
</head>
<body>

<%- partial("includes/_header.ejs") %>
<%- partial("includes/_summary.ejs") %>
<%- partial("includes/_curriculum.ejs") %>
<%- partial("includes/_sponsors.ejs") %>
<%- partial("includes/_footer.ejs") %>

</body>
</html>
```

Department of Computing &
Mathematics



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHA PHORT LÁIRGE

BSc (Hons) the Internet of Things



Programming

Learn a broad range of programming and problem solving skills, including exciting new platforms, software tools and languages. Use these skills to build apps for mobile, cloud and device based IoT applications. Evolve a portfolio of fascinating applications.

Networks

This strand will explore modern networks and cloud technology. Be able to configure, network and manage all categories of computer systems from simple controllers to single board computers, mobiles and full workstations.

Project

Building exciting IoT projects in every semester of the programme. Your projects will combine skills acquired from the other strands and enable you to build a comprehensive and compelling portfolio of IoT applications and services.

Mathematics

Introduce foundation concepts for many of the more applied concepts in the other Strands. Learn mathematical techniques in a modern context and apply core principles in new and interesting ways.



[facebook](#) [twitter](#) [linkedin](#)

Step 06: Partials

- Many Pages can share the same general structure.
- Using partial can help in making the site DRY
- We can include different sections to the same general structure
- Each section is called a *Partial*

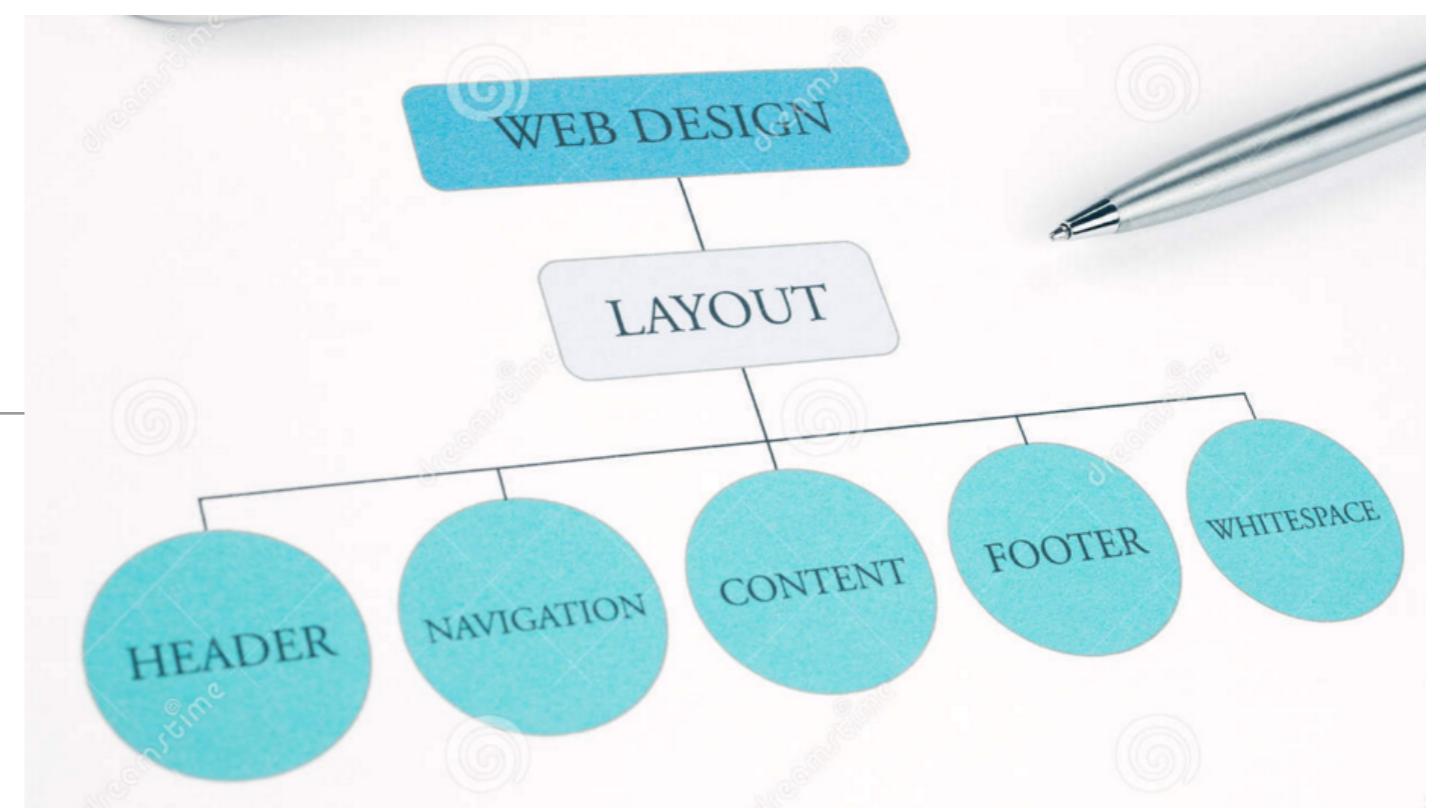
index.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="h...
  <link type="text/css" rel="stylesheet" href="s...
  <title>BSc in the Internet of Things</title>
</head>
<body>

<% partial("includes/_header.ejs") %>
<% partial("includes/_summary.ejs") %>
<% partial("includes/_curriculum.ejs") %>
<% partial("includes/_sponsors.ejs") %>
<% partial("includes/_footer.ejs") %>

</body>
</html>
```

Step 06: Layouts



- Layouts are another powerful mechanisms for adopting a DRY approach
- With Layouts, we can define the structure of the overall page...
- ... and each page that uses the layout substituting into a specific part of the page

Step 06: Layouts

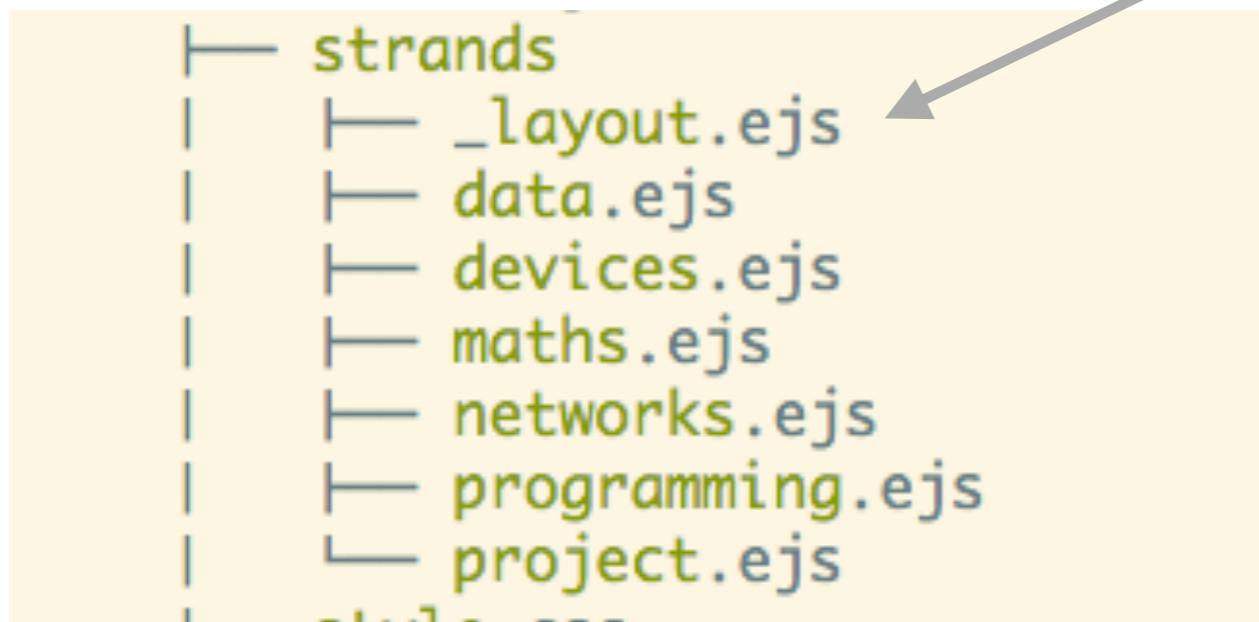
_layout.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Open+Sans"
  <link type="text/css" rel="stylesheet" href="../style.css" media="screen"/>
  <title> IoT Strands </title>
</head>
<body>
<%- partial("../includes/_header.ejs") %>
<%- yield %>
<%- partial("../includes/_footer.ejs") %>
</body>
</html>
```

- A layout is always called ‘ _layout.ejs’
- It **can** contain standard html + partial includes if necessary
- It **must** contain a **<% yield %>** statement
- This yield is replaced by the contents of another template...

Step 06: Layouts

- If a folder contains a file called ‘_layout.ejs’:
 - Each page is assumed to be based on this layout
 - The template engine will build each page from the _layout + the individual page concerned



_layout.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Open+Sans"
  <link type="text/css" rel="stylesheet" href="../style.css" media="screen"/>
  <title> IoT Strands </title>
</head>
<body>

<%- partial("../includes/_header.ejs") %>
<%- yield %>
<%- partial("../includes/_footer.ejs") %>

</body>
</html>
```

- For the ‘programming’ page - its contents are inserted into the layout, replacing the ‘yield’ statement.
- ‘Programming.ejs’ is a page template without head, body or other elements..
- It just contains just content to complete the layout.

Step 06: Layouts

strands/programming.ejs

```
<article>
  <h1> Programming </h1>
  <p>
    
    The IoT requires a new breed of software skills, wi
  </p>
</article>

<figure>
  
  
</figure>

<article>
  <h2> Programming Learning Path </h2>
  <p>
    The Data Science strand will begin with the fundame
  </p>
</article>
```

Template Languages - EJS

EJS

<% Embedded JavaScript %>

About.
EJS cleans the HTML out of your JavaScript with client side templates. After EJS gets its rubber gloves on dirty code, you'll feel organized and uncluttered.

Download EJS
Developer 1.0 Production (9.8KB) 

Featured in [JavaScriptMVC - Develop with Direction!](#)

Try it!

The Data.

EJS combines data and a template to produce HTML. Here, our example data has a title and a list of supplies.

```
{ title: 'Cleaning Supplies'  
supplies: ['mop', 'broom', 'duster'] }
```

The Template.

Like ERB, JavaScript between `<% %>` is executed. JavaScript between `<%= %>` adds HTML to the result.

- ★ Type HTML or JavaScript in the **template**
- Watch as your changes update the **result**.

Try typing these suggestions:

[Add the title](#) [Add links](#) [Add image](#)

Insert "`<h1> <%= title %> </h1>`"

[Show me »](#)

```
<ul>  
<% for(var i=0; i<supplies.length; i++) { %>  
  <li><%= supplies[i] %></li>  
<% } %>  
</ul>
```

Type here

The Result.

The **result** on the right is the output of the **template** processed with the **data**.

If you make a mistake, EJS provides the line number and a message for easier debugging.

- mop
- broom
- duster

The result is shown here

Getting Started with EJS

Are you ever faced with a mess of HTML string concatenations like this?

```
var html = "<h1>" + data.title + "</h1>"  
html += "<ul>"  
for(var i=0; i<data.supplies.length; i++) {  
    html += "<li><a href='supplies/" + data.supplies[i] + "'>"  
    html += data.supplies[i] + "</a></li>"  
}  
html += "</ul>"
```

If you're a web developer, the answer is probably yes. Beyond being ugly, the structure of your HTML is lost in the JavaScript. Adding to this layout would be difficult. How can we clean it up?

In this tutorial, we'll show you how to improve the above code that produces a list of supply links. We'll use Embedded JavaScript (EJS) to return this code to a straightforward, maintainable HTML structure.

EJS is a JavaScript templating library. It is commonly used for building html strings from JSON data. Typically, EJS works like this:



This tutorial walks you through:

1. Including EJS
2. Creating a template
3. Using view helpers
4. Using error handling
5. When to use EJS

Include EJS

Before we put on the rubber gloves and get to the heavy scrubbing, lets get set up a bit.

Your page needs to include EJS so your JavaScript can use it. Start by downloading [ejs_production.js](#) from [Google Code](#) or the [subversion repository](#).

Next add EJS to your HTML like this:



- Another template language... different in approach from EJS...

[Language Reference](#)[API](#)[Command Line](#)[Change Log](#)[Code Coverage](#)[Test Results](#)[GitHub Repository](#)

```
doctype html
html(lang="en")
  head
    title= pageTitle
    script(type='text/javascript').
      if (foo) {
        bar(1 + 5)
      }
  body
    h1 Jade - node template engine
    #container.col
      if youAreUsingJade
        p You are amazing
      else
        p Get on it!
    p.
      Jade is a terse and simple
      templating language with a
      strong focus on performance
      and powerful features.
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Jade</title>
    <script type="text/javascript">
      if (foo) {
        bar(1 + 5)
      }
    </script>
  </head>
  <body>
    <h1>Jade - node template engine</h1>
    <div id="container" class="col">
      <p>You are amazing</p>
      <p>
        Jade is a terse and simple
        templating language with a
        strong focus on performance
        and powerful features.
      </p>
    </div>
  </body>
</html>
```

Search the docs, try “EJS” or “Stylus”

- Harp includes EJS and Jade template engines
- As long as your page is being ‘served’ by harp, Ejs & Jade directives will be implemented

[Overview](#)

[Quick Start](#)

[Environment](#)

[Install](#)

[Init](#)

[Server](#)

[Multihost](#)

[Compile](#)

[Update](#)

[Lib](#)

[Development](#)

[The Rules](#)

[Public](#)

[Layout](#)

[Yield](#)

[Partial](#)

[Globals](#)

[Metadata](#)

Layouts

A Layout is a common template that includes all content except for one main content area. You can think of a Layout as the inverse of a [partial](#).

- [Creating Layouts with EJS](#)
- [Creating Layouts with Jade](#)
- [Multiple Layouts](#)
- [Explicit Layouts](#)
- [No Layout](#)

Why?

Often sites and apps will have common headers and footers and the only area that needs to change is the body. This is an ideal use case for a layout.

Usage

A Layout requires a layout file, written in EJS or Jade, and a [yield](#) property to tell Harp where to insert the content.

Example using EJS Templating

Given a really simple app / project with this structure:

```
myapp.harp.io/
  |- _layout.ejs
  +- index.ejs
```

_layout.ejs

Compiling Pages

```
iot-web-ejs
├── harp.json
└── public
    ├── assets
    │   └── images
    │   ...
    ├── includes
    │   ├── _curriculum.ejs
    │   ├── _footer.ejs
    │   ├── _header.ejs
    │   ├── _sponsors.ejs
    │   └── _summary.ejs
    ├── index.ejs
    ├── strands
    │   ├── _layout.ejs
    │   ├── data.ejs
    │   ├── devices.ejs
    │   ├── maths.ejs
    │   ├── networks.ejs
    │   ├── programming.ejs
    │   └── project.ejs
    └── style.css
```

harp compile

```
.
├── harp.json
└── public
    ├── assets
    │   ...
    ├── includes
    │   ├── _curriculum.ejs
    │   ├── _footer.ejs
    │   ├── _header.ejs
    │   ├── _sponsors.ejs
    │   └── _summary.ejs
    ├── index.ejs
    ├── strands
    │   ├── _layout.ejs
    │   ├── data.ejs
    │   ├── devices.ejs
    │   ├── maths.ejs
    │   ├── networks.ejs
    │   ├── programming.ejs
    │   └── project.ejs
    └── style.css
www
├── assets
│   └── images
│   ...
├── index.html
└── strands
    ├── data.html
    ├── devices.html
    ├── maths.html
    ├── networks.html
    ├── programming.html
    └── project.html
    └── style.css
```

- the harp ‘compile’ command will generate a complete copy of your site - with all directives removed and full pages replacing all fragments.