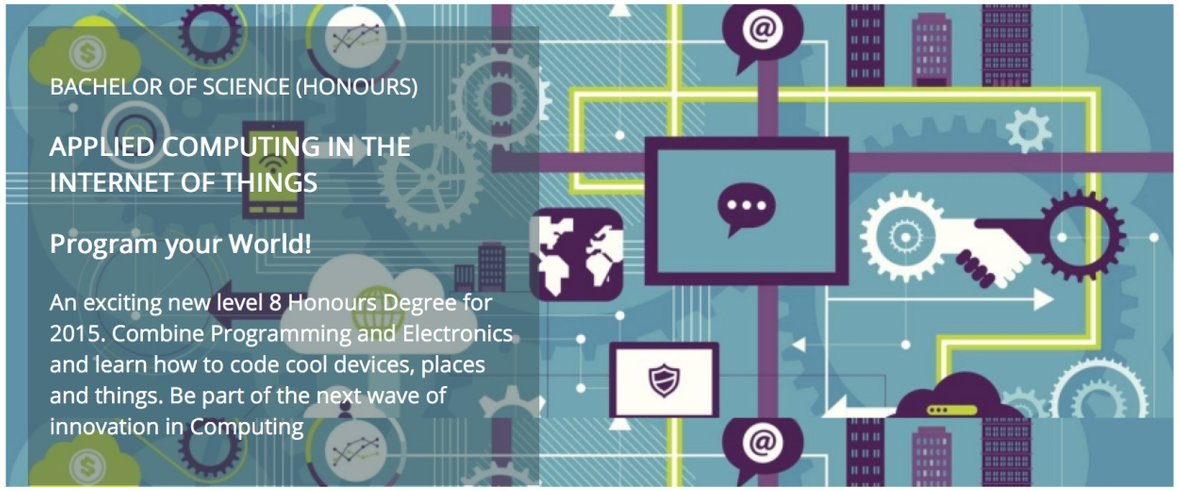
Web Development

Templates

Department of Computing & Mathematics

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA 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 porfolio of facinating aplications.

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 controlers to single board 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 an 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 an interesting ways.

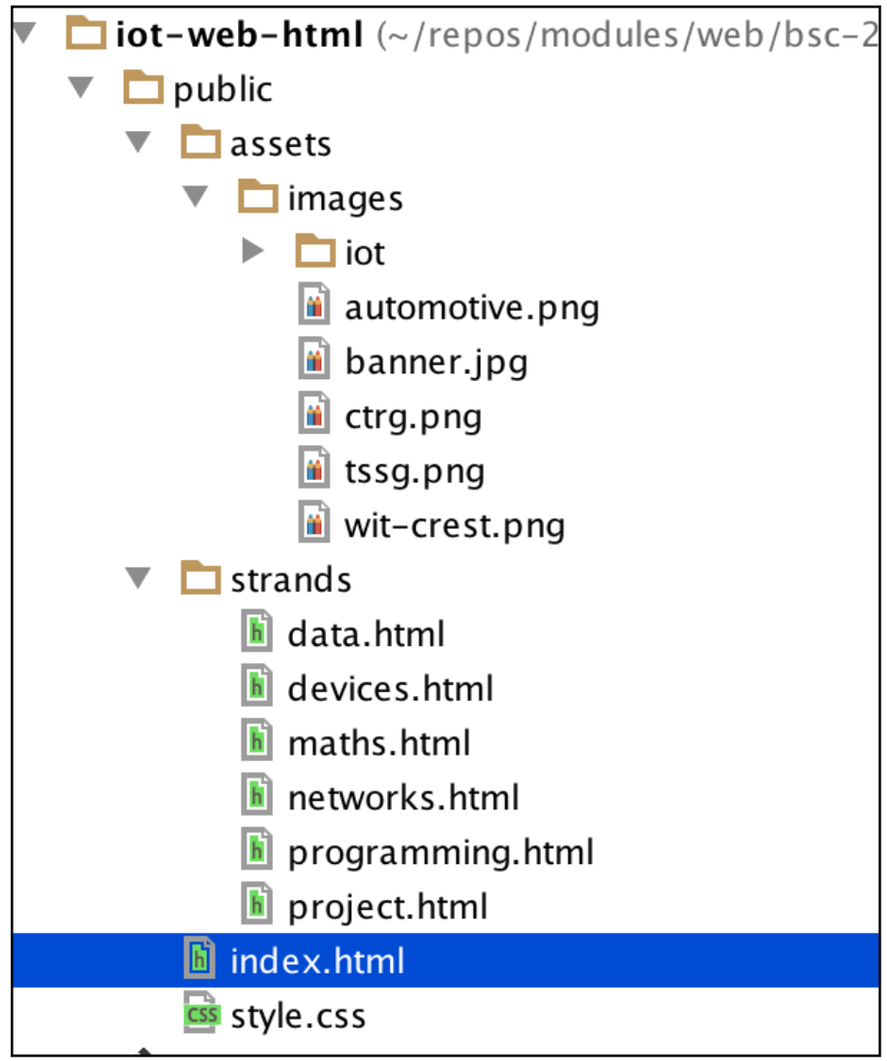
Supported by leading edge research at...

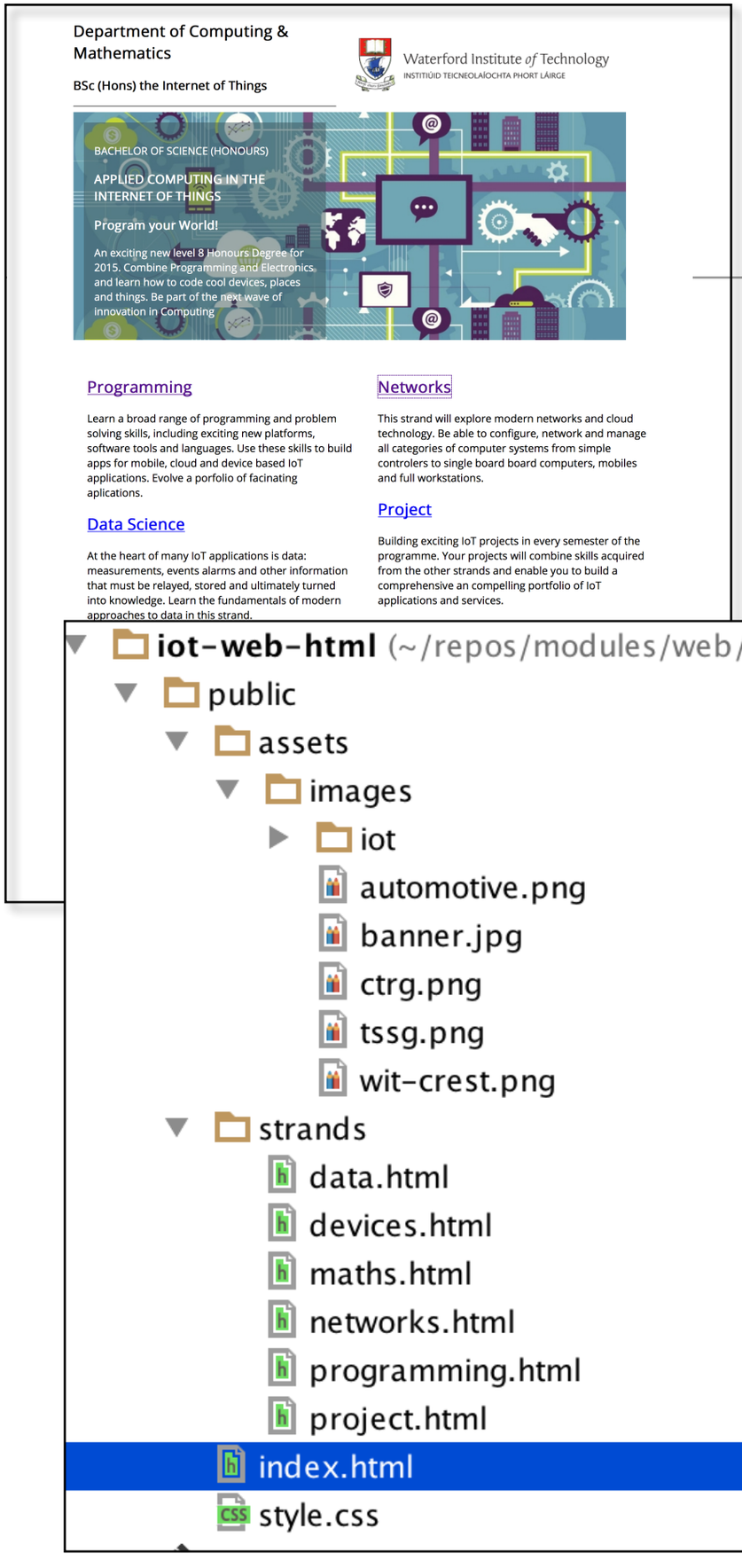
TSSG

ctrg
convergent technologies research group

AUTOMOTIVE CONTROL GROUP
Software Engineering for the Connected Car

facebook twitter linkedin





Templates Why?

- 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
- —>

21 Repeated Sections

- 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]

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

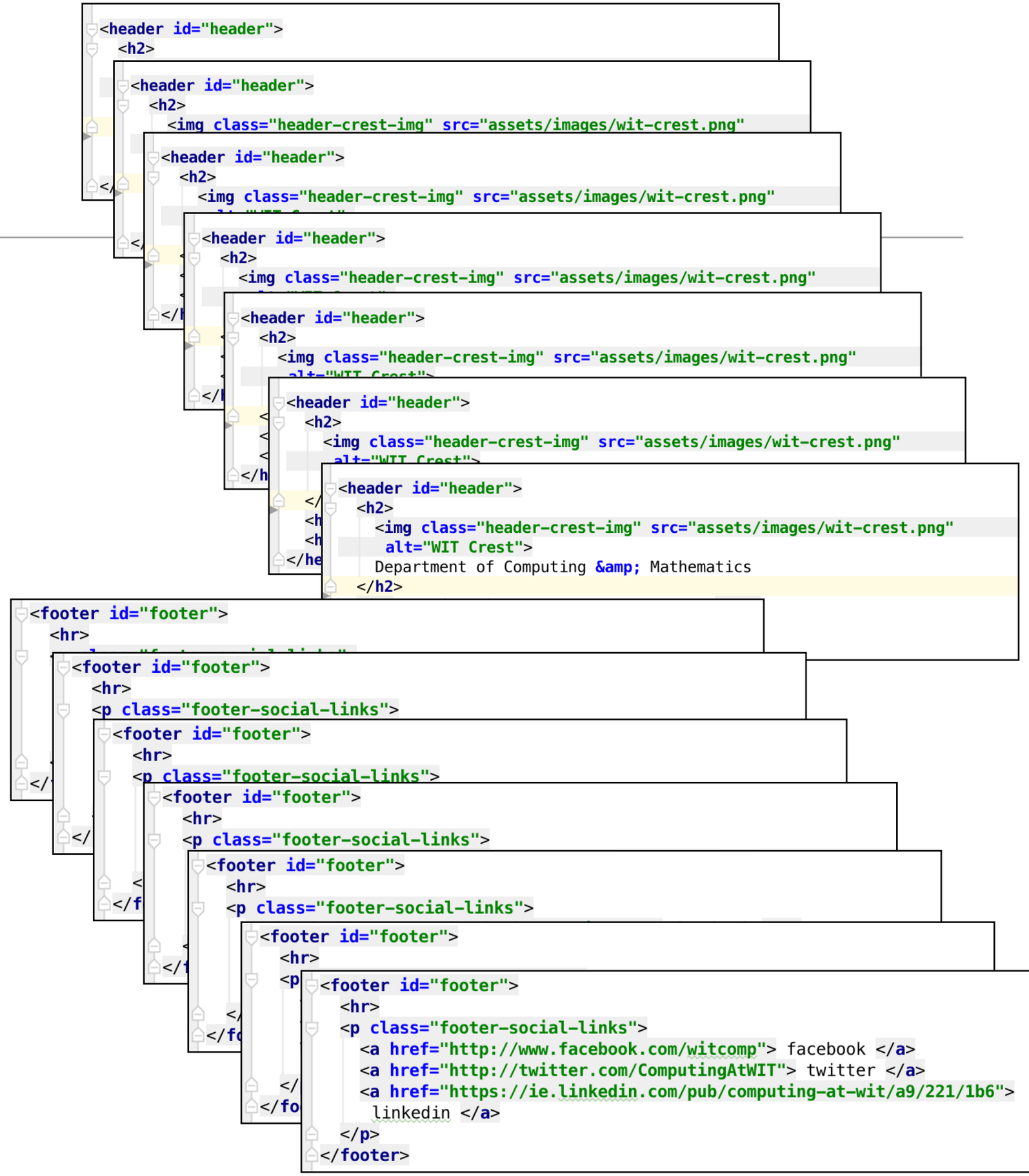
Don’t Repeat Yourself

vs

Write Everything Twice

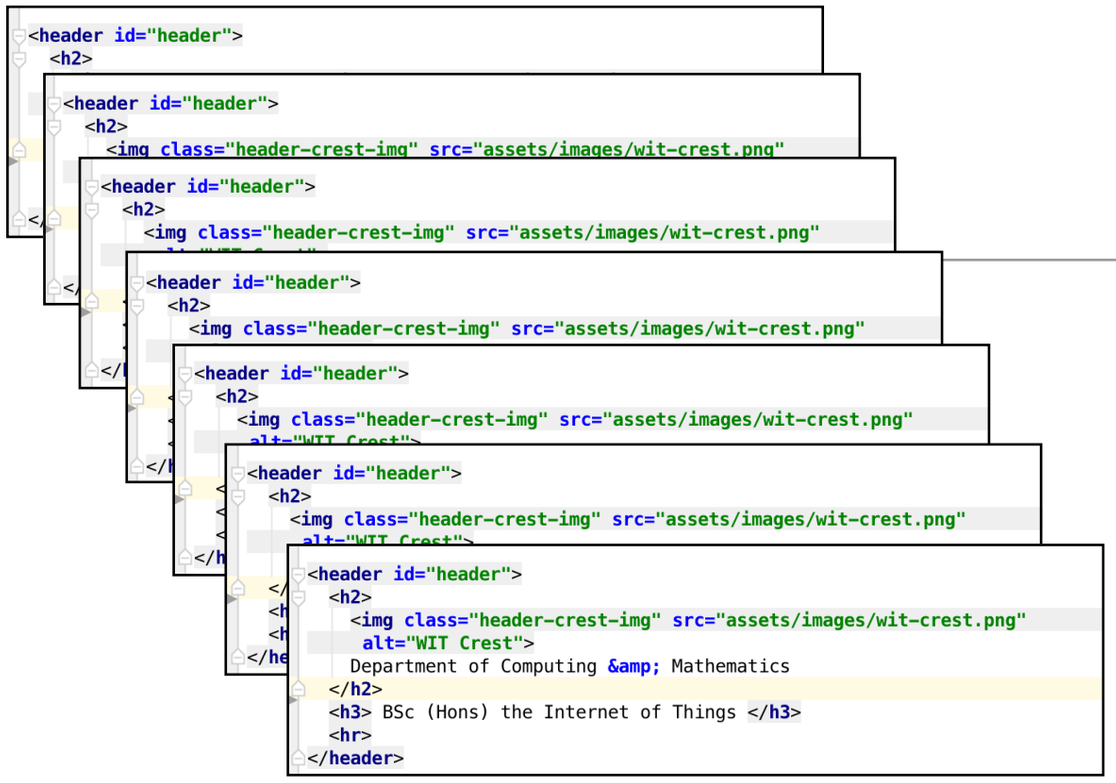
OR

We Enjoy Typing



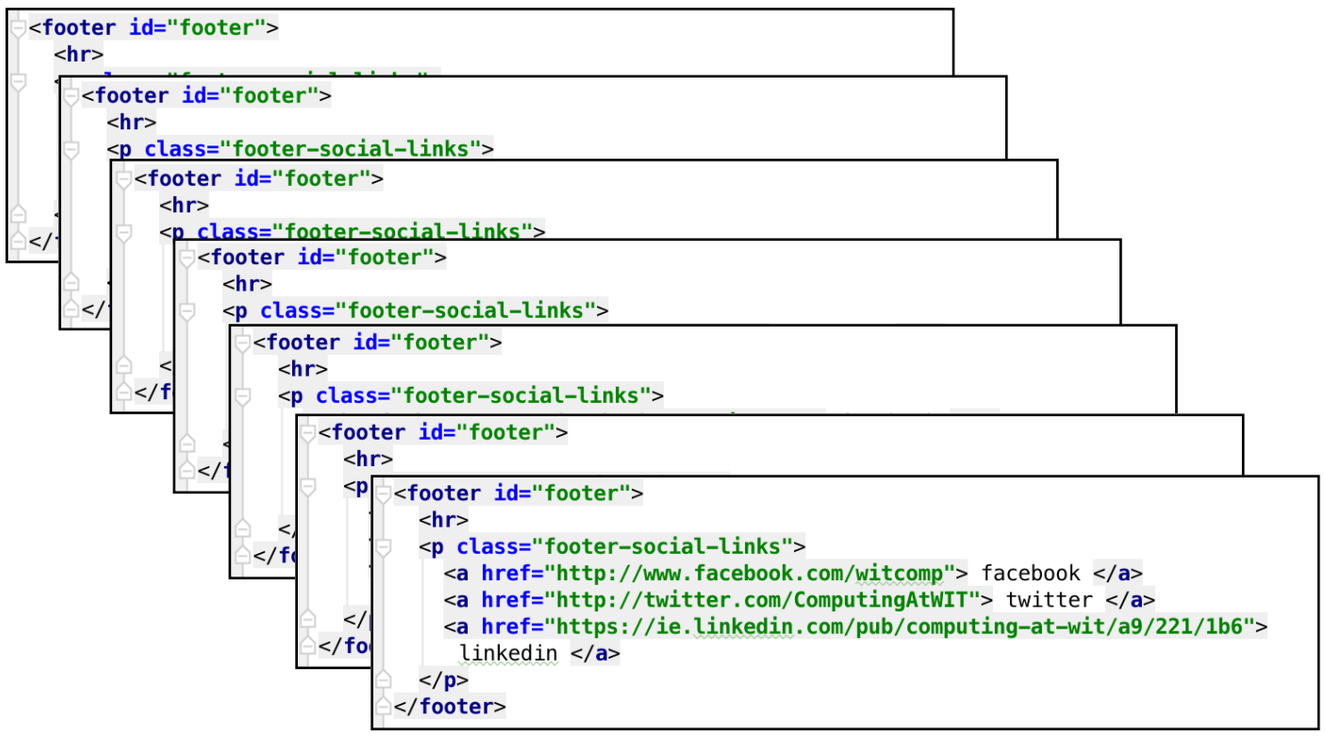
DRY

Single Header + Footer Template



- Incorporate the SAME single header/footer into ALL pages

```
<header id="header">
  <h2>
    <img class="header-crest-img" src="assets/images/wit-crest.png"
     alt="WIT Crest">
    Department of Computing &amp;amp; Mathematics
  </h2>
  <h3> BSc (Hons) the Internet of Things </h3>
  <hr>
</header>
```



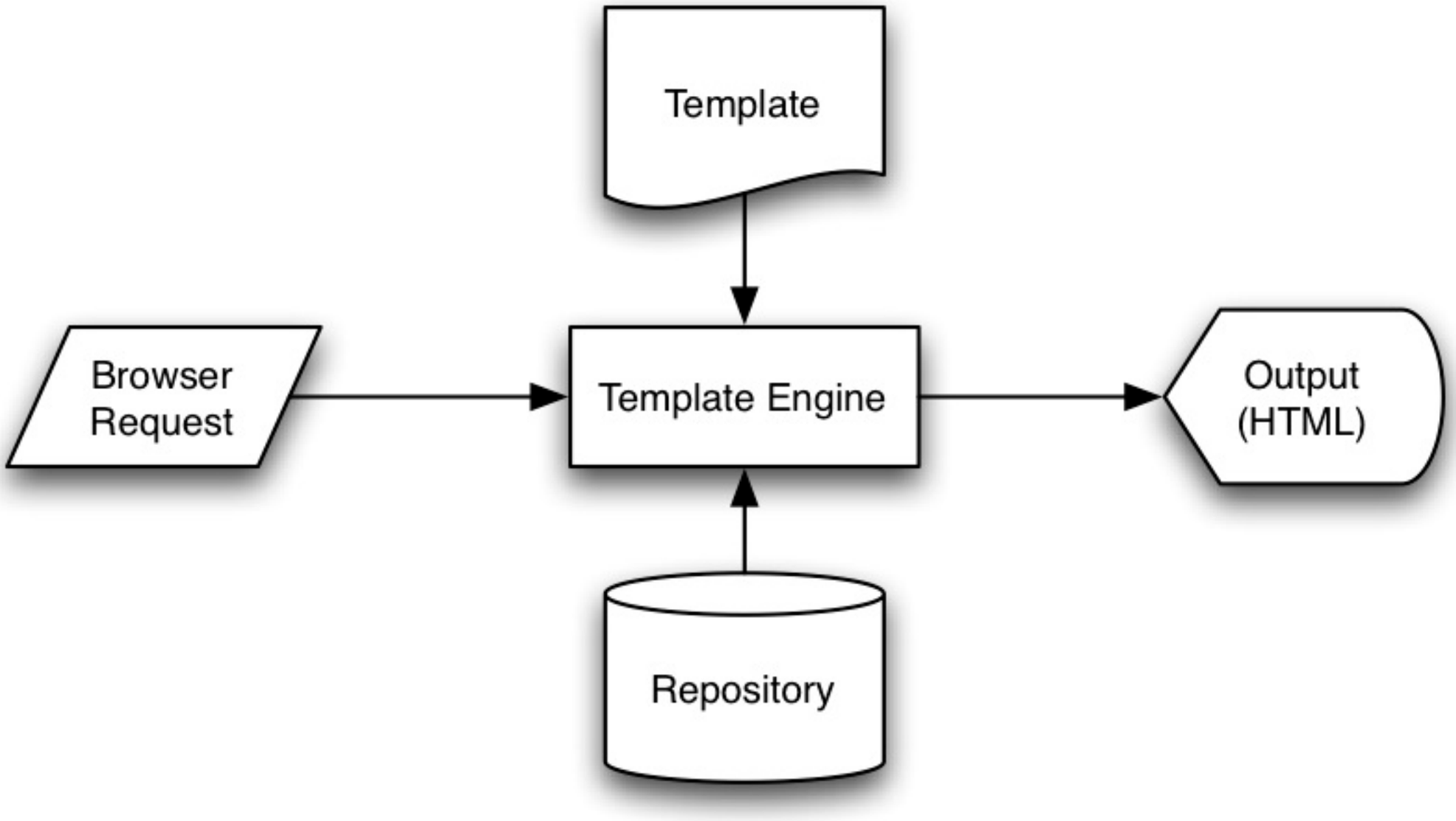
- Any changes - made just once in the single header/footer

```
<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>
```

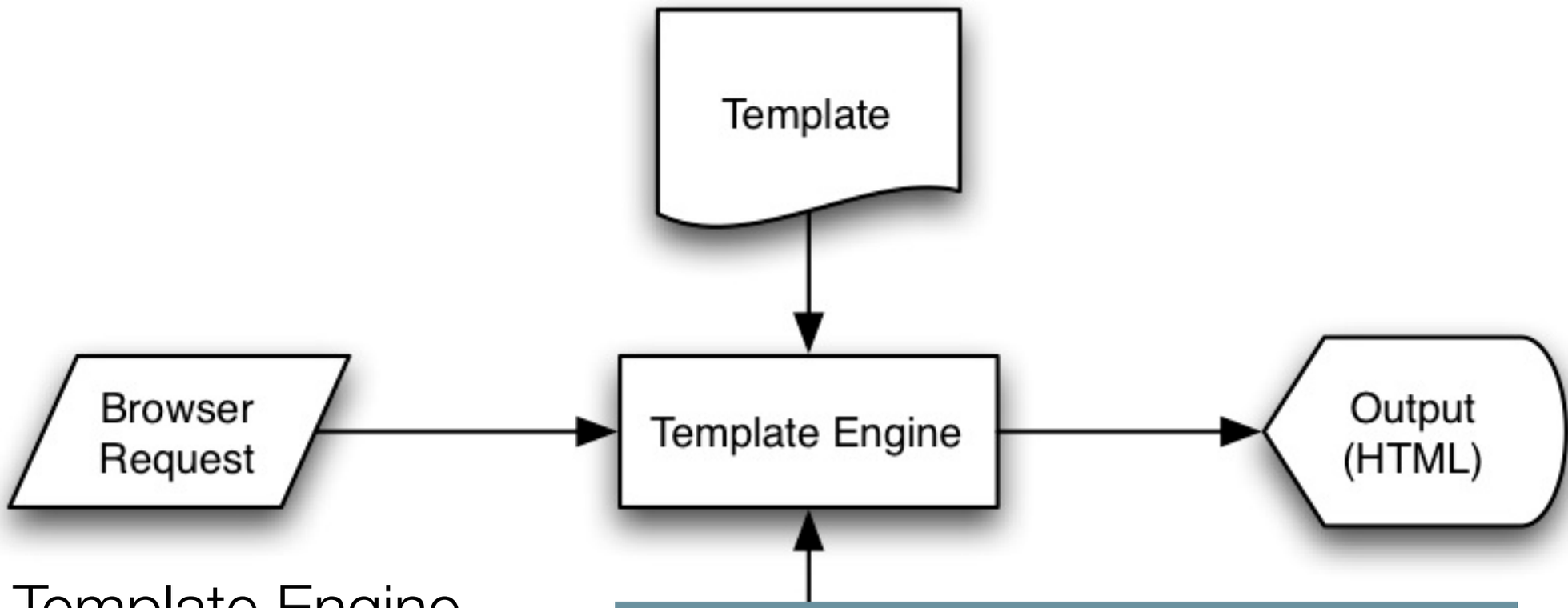
Web Template System

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


Template
Browser Request
Template Engine
Output (HTML)
Repository

Harp.js



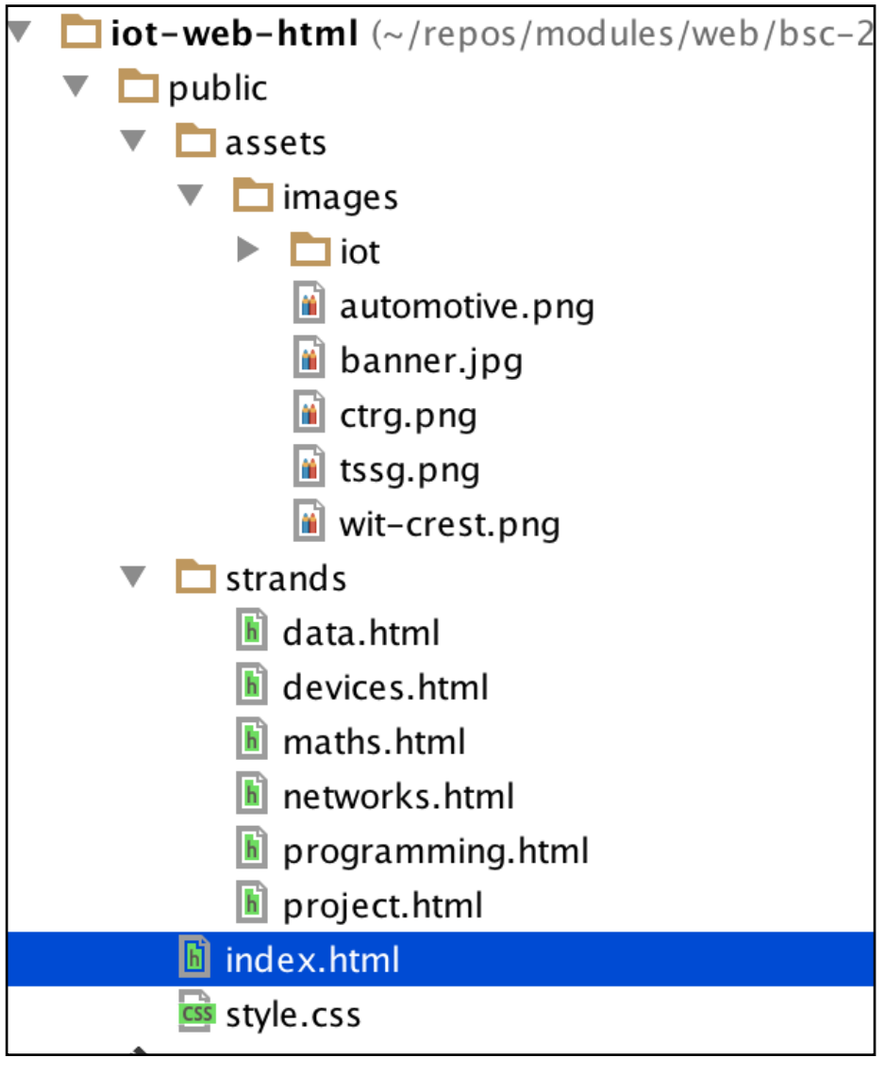
- Harp.js is our Template Engine
- It ‘serves’ the site
- If *Request* is for ordinary page the page is ‘rendered’ without modification
- If *Request* is for a page that is composed of templates, harp assembles the page and renders the complete page to the browser

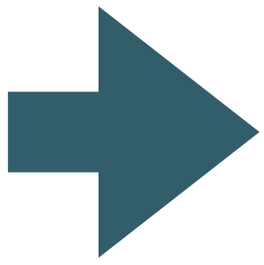
Lab05b

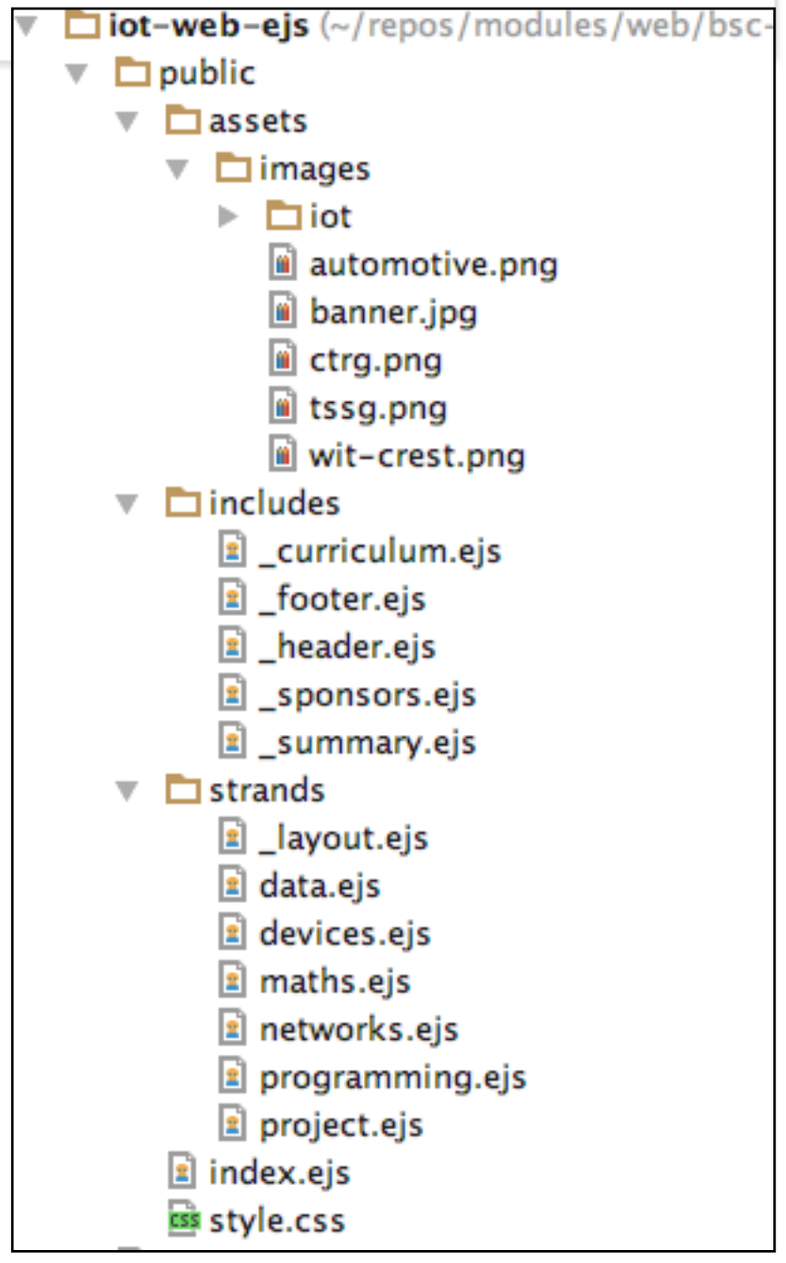
HTML Templates | Lab-5b Templating | 01 | 02 | 03 | 04 | 05 | 06 | Exercises | 🏠

Objectives

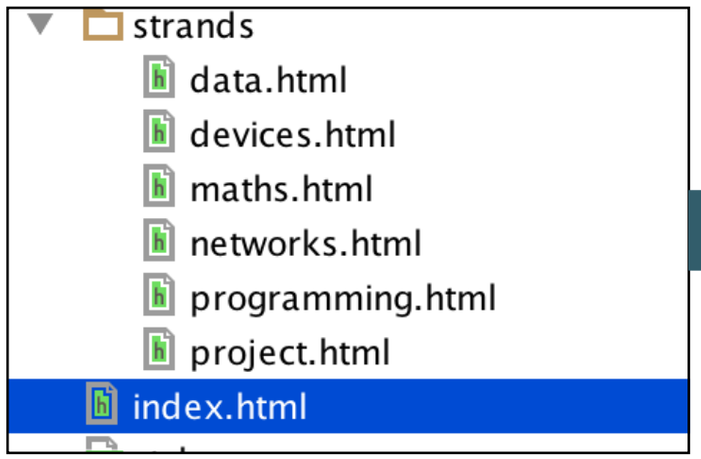
Rebuild the IoT web site from thee last lab using templating. This version of the site will aim to significantly reduce the content the author has to manage by reusing 'templates' containing common sections.



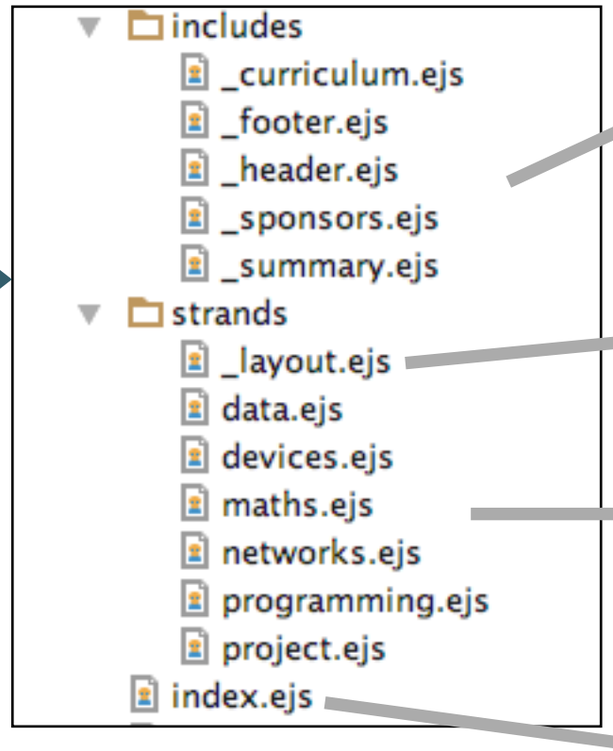




Lab05b



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

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

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

```
iot-web-ejs
├── harp.json
└── public
    ├── assets
    |   └── images
    |       ├── automotive.png
    |       ├── banner.jpg
    |       ├── ctrg.png
    |       | ....
    |       ├── tssg.png
    |       └── wit-crest.png
    ├── index.html
    ├── strands
    |   ├── data.html
    |   ├── devices.html
    |   ├── maths.html
    |   ├── networks.html
    |   ├── programming.html
    |   └── project.html
    └── style.css
```

Step 02 - Header & Footer templates

_header.ejs

```
<header id="header">
  <h2>
    <img class="header-crest-img" src="assets/images/wit-crest.png" alt="WI
    Department of Computing &amp; 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"> linke
  </p>
</footer>
```

```
iot-web-ejs
├── harp.json
└── public
    ├── assets
    |     ...
    ├── includes
    |   ├── _header.ejs
    |   └── _footer.ejs
    ├── index.ejs
    ├── strands
    |   ├── data.html
    |   ├── devices.html
    |   ├── maths.html
    |   ├── networks.html
    |   ├── programming.html
    |   └── project.html
    └── style.css
```

- 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>
    <img class="header-crest-img" src="assets/images/wit-crest.png" alt="WIT Crest">
    Department of Computing &amp;amp; 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:

```
<img class="header-crest-img" src="/assets/images/wit-crest.png" alt="WIT Crest">
```

- 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

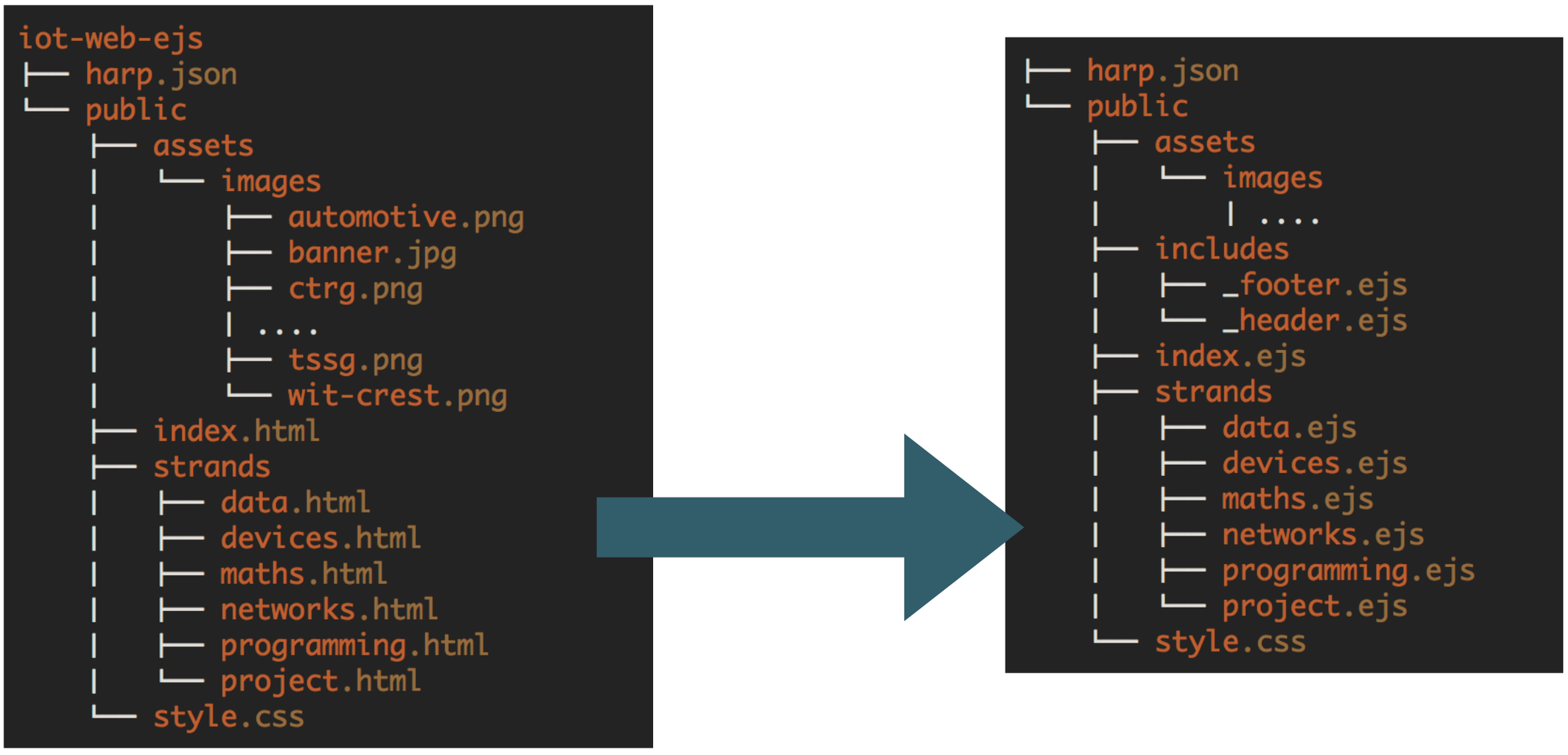
```
<header id="header">
  <h2>
    <img class="header-crest-img" src="assets/images/wit-crest.png" alt="WIT Crest">
    Department of Computing &amp;amp; Mathematics
  </h2>
  <h3> BSc (Hons) the Internet of Things </h3>
  <hr>
</header>
```

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

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

```
<img class="header-crest-img" src="/assets/images/wit-crest.png" alt="WIT Crest">
```

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") %>
```

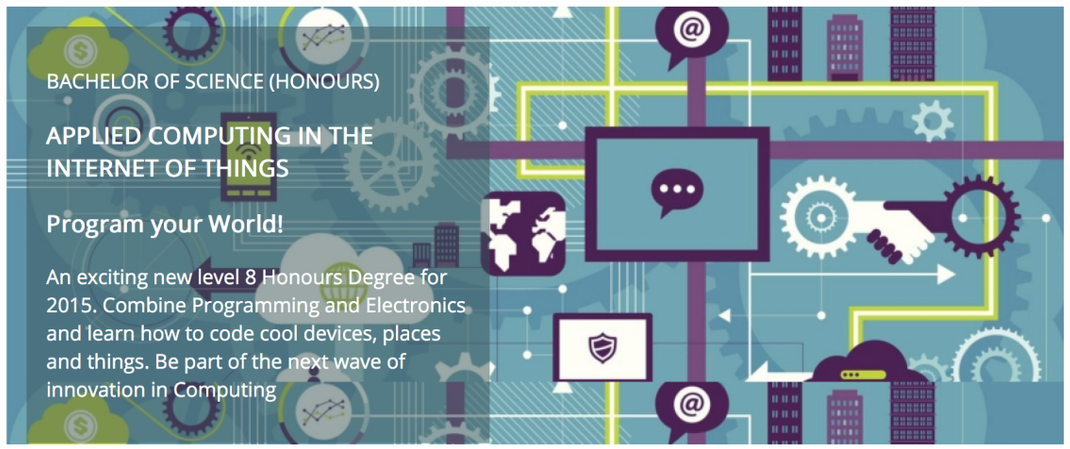
- DRY first steps…

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

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

<article>
  <h1> Devices </h1>
  <p>
    <img class="strand-right-img" src="../assets/images/ic
    The IoT professional must be comfortable when dealing
  </p>
</article>

<figure>
  <img class="strand-timeline-img" src="../assets/images/i
  <img class="strand-modules-treble-img" src="../assets/im
</figure>

<article>
  <h2> Devices Learning Path </h2>
  <p>
    <img class="strand-left-img" src="../assets/images/iot
    As a student on this programme, you will start to buil
  </p>
</article>

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

</body>
</html>
```

Department of Computing & Mathematics

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA 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 porfolio of facinating aplications.

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 controlers to single board 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 an 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 an interesting ways.

Supported by leading edge research at...

TSSG

ctrg convergent technologies research group

AUTOMOTIVE CONTROL GROUP
Software Engineering for the Connected Car

facebook twitter linkedin

```
<!DOCTYPE html>
<html lang="en">
<head...>
<body>

<header id="header">
  <h2>
    <img class="header-crest-img" src="assets/images/wit-crest.png"
     alt="WIT Crest">
    Department of Computing &amp;amp; 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>
    <img class="footer-img" src="assets/images/tssg.png" alt="TSSG">
    <img class="footer-img" src="assets/images/ctrg.png" alt="CTRG">
    <img class="footer-img" src="assets/images/automotive.png" alt="ATG">
  </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
```

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

```
<!DOCTYPE html>
<html lang="en">
<head...>
<body>

<header id="header">
  <h2>
    <img class="header-crest-img" src="assets/images/wit-crest.png"
     alt="WIT Crest">
    Department of Computing &amp;amp; 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>
    <img class="footer-img" src="assets/images/tssg.png" alt="TSSG">
    <img class="footer-img" src="assets/images/ctrg.png" alt="CTRG">
    <img class="footer-img" src="assets/images/automotive.png" alt="ATG">
  </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: 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 cod
    </p>
  </div>
</article>
```

_sponsors.ejs

```
<section id="sponsors">
  <hr>
  <h4> Supported by leading edge research at... </h4>
  <p>
    <img class="footer-img" src="assets/images/tssg.png" alt="TSSG">
    <img class="footer-img" src="assets/images/ctrg.png" alt="CTRG">
    <img class="footer-img" src="assets/images/automotive.png" alt="ATG">
  </p>
</section>
```

Step 05: curriculum

_curriculum.ejs

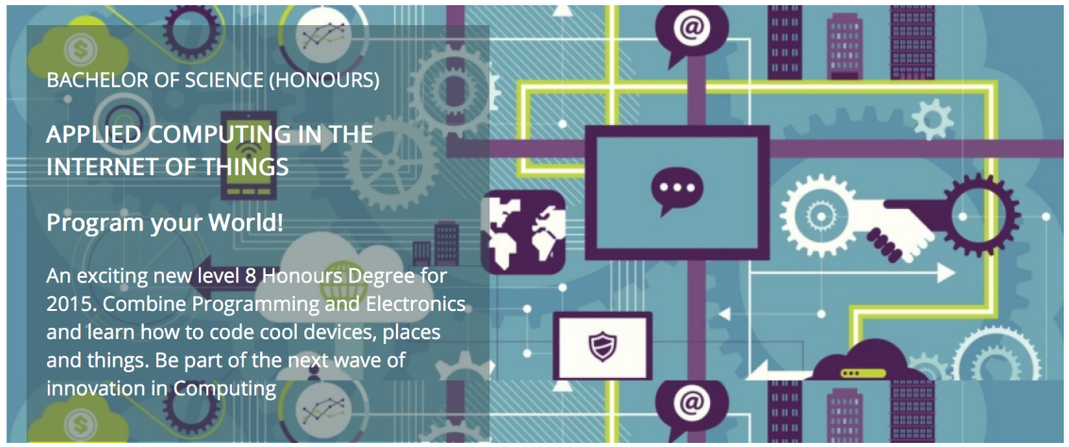
```
<article id="curriculum">
  <hr>
  <section id="col1">
    <h2><a href="strands/programming.html"> Programming </a></h2>
    <p>
      Learn a broad range of programming and problem solving skills, including exciting new platforms, software too
    </p>

    <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
    </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 soph
    </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 co
    </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
    </p>

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

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

index.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="htt
  <link type="text/css" rel="stylesheet" href="sty
  <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>
```

- harp will now compose the page from 5 templates



Department of Computing & Mathematics

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

BSc (Hons) the Internet of Things

Programming

Learn a broad range of programming and problem
rms,
kills to build
T
;

ta:
nformation
y turned
f modern

cal devices.
sensors to
ghts or
th the

Networks

This strand will explore modern networks and cloud technology. Be able to configure, network and manage all categories of computer systems from simple controlers to single board 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 an 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 an interesting ways.

ctrg
convergent technologies research group

AUTOMOTIVE CONTROL GROUP
Software Engineering for the Connected Car

facebook twitter linkedin

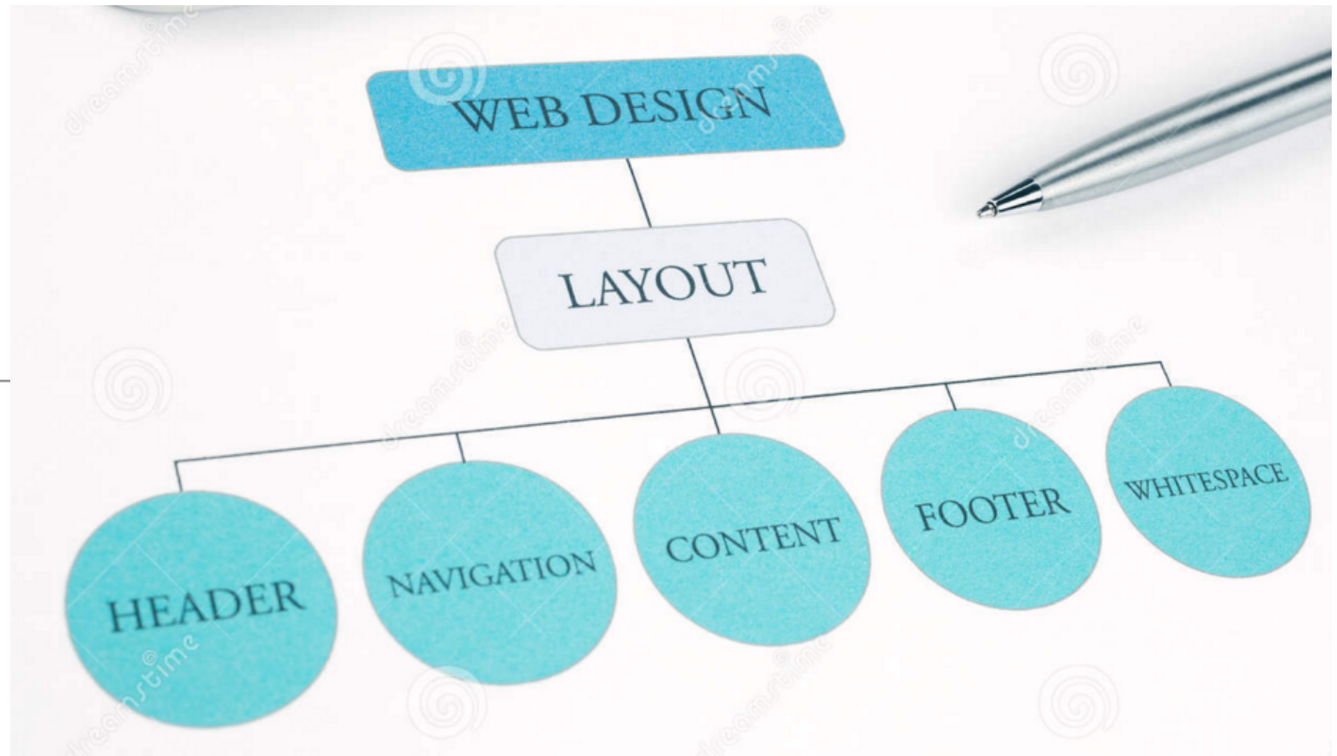
Step 06: Partials

- 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="htt
  <link type="text/css" rel="stylesheet" href="sty
  <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

```
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
```

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

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>
```

strands/programming.ejs

```
<article>
  <h1> Programming </h1>
  <p>
    <img class="strand-right-img" src="../assets/images/iot/p
    The IoT requires a new breed of software skills, with an
  </p>
</article>

<figure>
  <img class="strand-timeline-img" src="../assets/images/iot/
  <img class="strand-modules-double-img" src="../assets/image
</figure>

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

- 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.

<%= EJS %>

Effective JavaScript templating.

GET STARTED

What is EJS?

"E" is for "effective." EJS is a simple templating language that lets you generate HTML markup with plain JavaScript. No religiousness about how to organize things. No reinvention of iteration and control-flow. It's just plain JavaScript.

Use plain JavaScript

We love JavaScript. It's a totally friendly language. All templating languages grow to be Turing-complete. Just cut out the middle-man, and use JS!

Fast development time

Don't waste time and attention figuring out arcane new syntax because 'elegance' — or how to preprocess your data so it will actually render right.

Simple syntax

JavaScript code in simple, straightforward scriptlet tags. Just write JavaScript that emits the HTML you want, and get your shit done.

Speedy execution

We all know how fast V8 and the other JavaScript runtimes have gotten. EJS caches the intermediate JS functions for fast execution.

Easy debugging

It's easy to debug EJS errors: your errors are plain JavaScript exceptions, with template line-numbers included.

Active development

EJS has a large community of active users, and the library is under active development. We're happy to answer your questions or give you help.

JADE LANGUAGE

Node Template Engine

Home API Command Line Language Reference

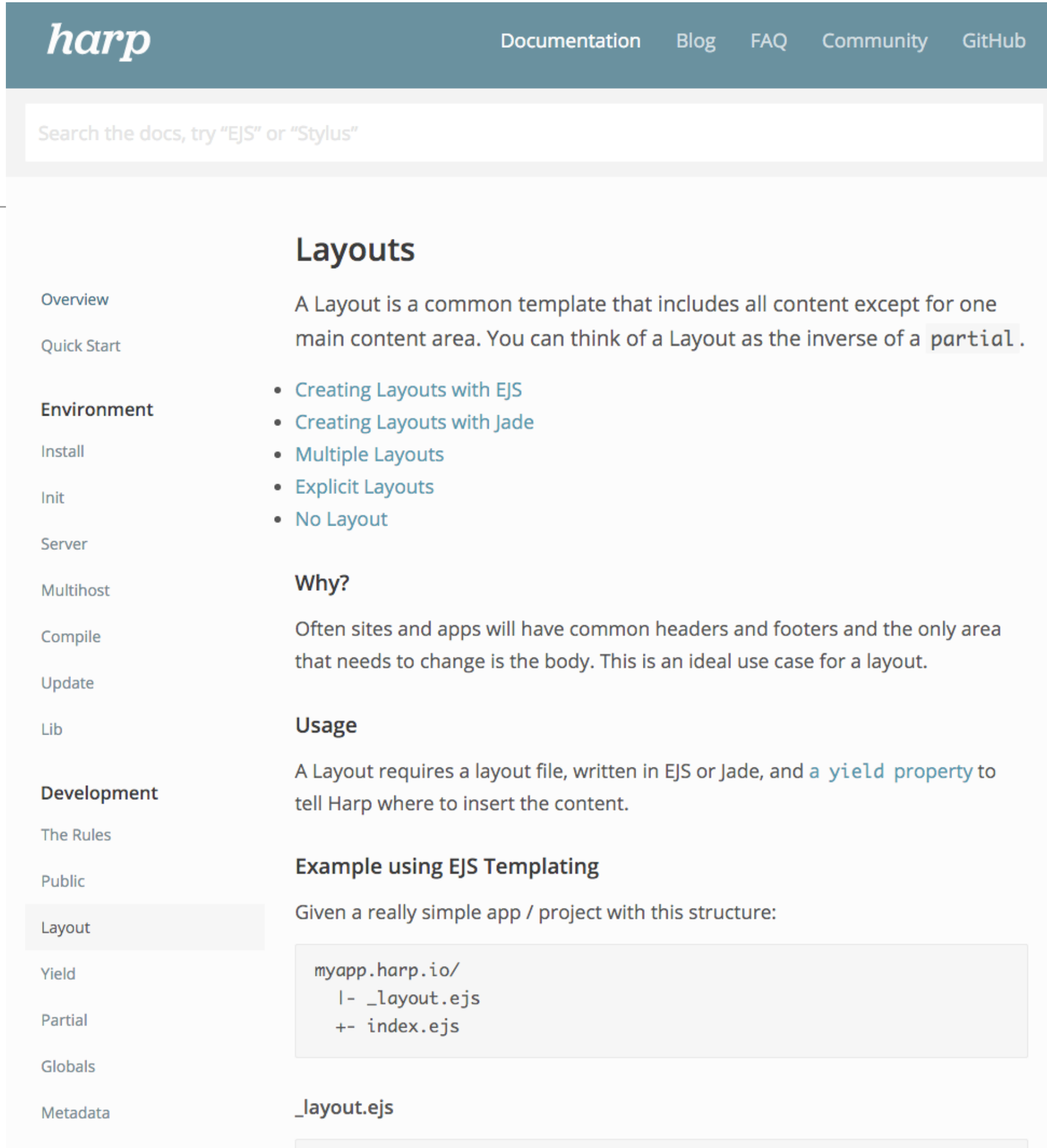
- Another template language… different in approach from EJS…

```
!!! 5
html(lang="en")
  head
    title= pageTitle
    :javascript
      | if (foo) {
      |    bar()
      | }
  body
    h1 Jade - node template engine
    #container
      - if (youAreUsingJade)
         You are amazing
      - else
         Get on it!
         Get on it!
         Get on it!
         Get on it!
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Jade</title>
    <script type="text/javascript">
      //<![CDATA[
      if (foo) {
        bar()
      }
      //]]>
    </script>
  </head>
  <body>
    <h1>Jade - node template engine</h1>
    <div id="container">
      <p>You are amazing</p>
    </div>
  </body>
</html>
```

- Harp includes EJS and Jade template engines

- As long as your page is being ‘served’ by harp, Ejs & Jade directives will be implemented

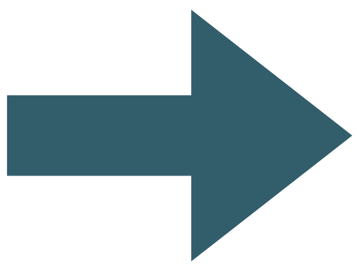


Compiling Pages

```
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
```

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



```
.
├── 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
```