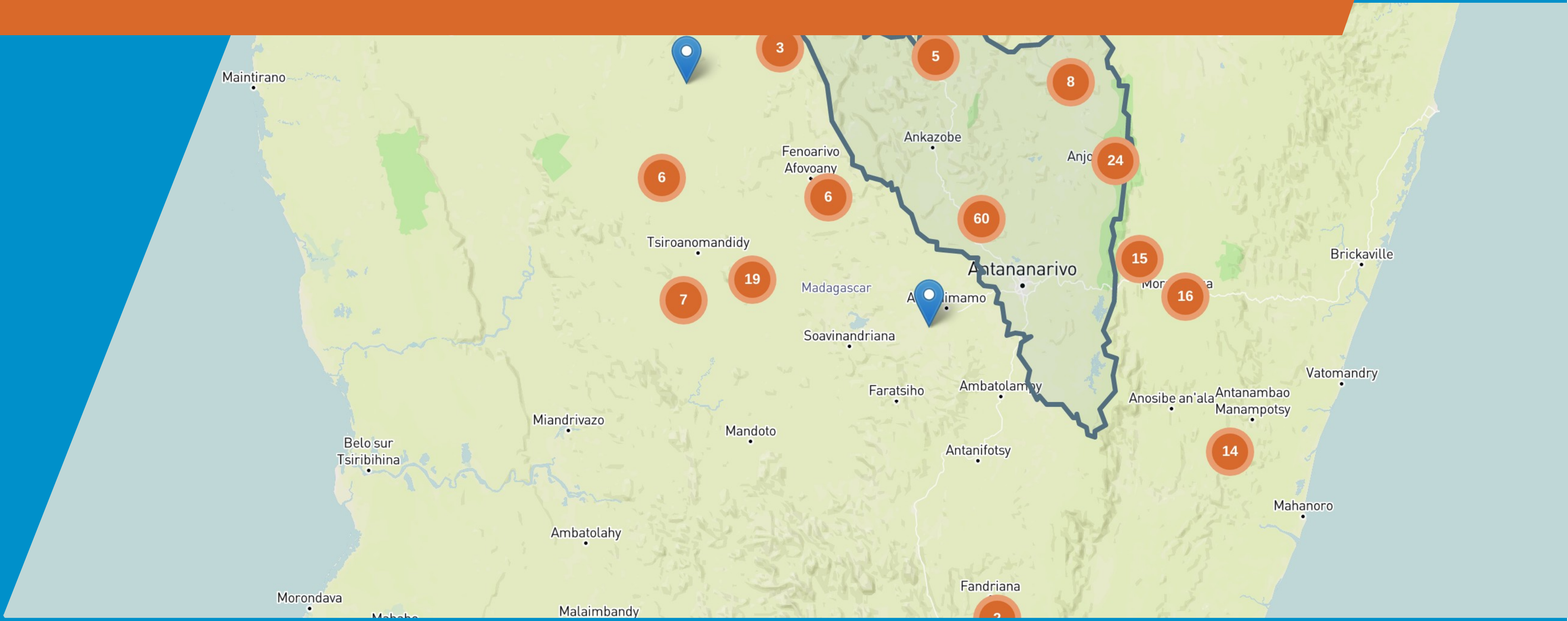
IEES · Public Health Surveillance

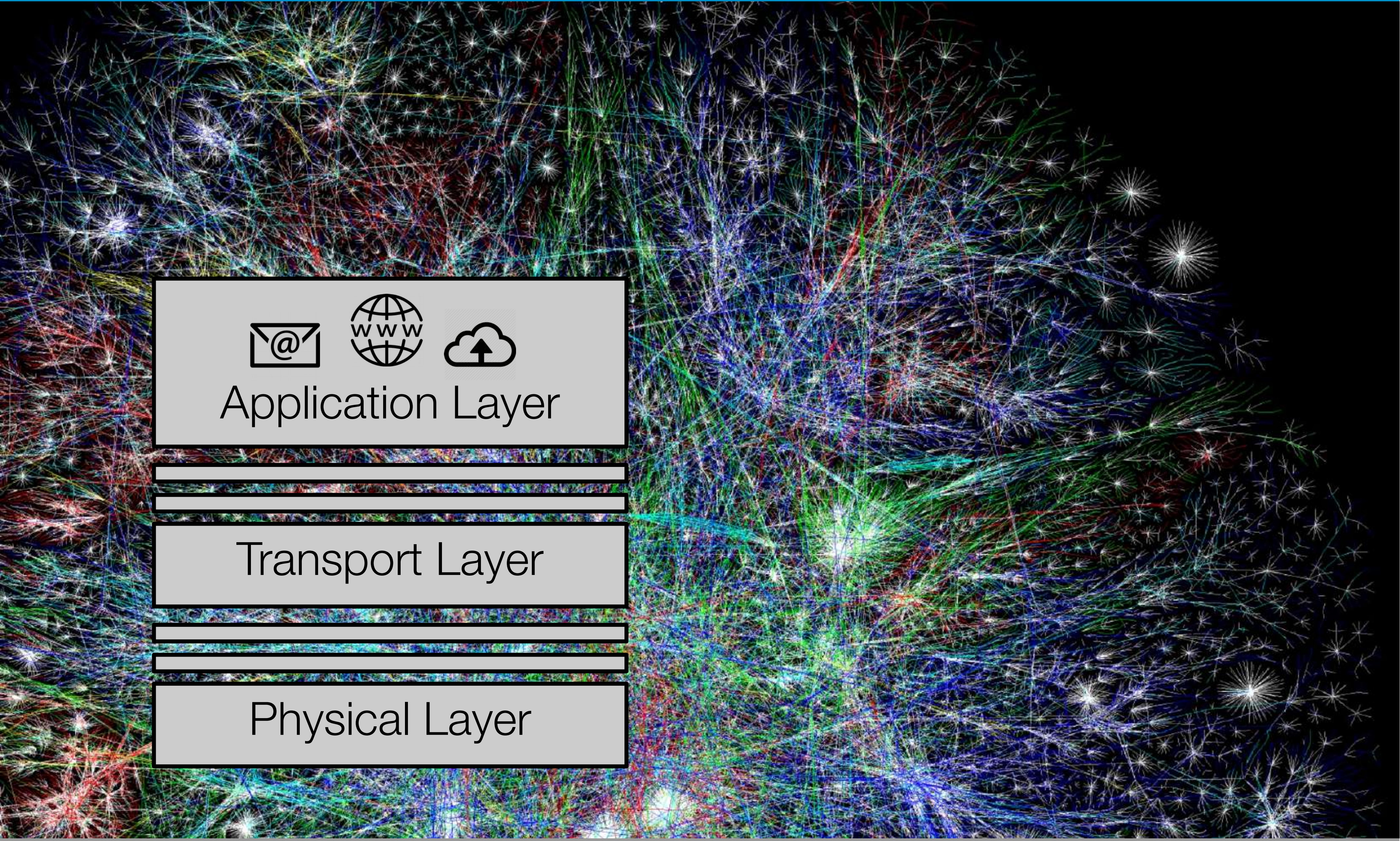


Meerkat Software Training
Web Programming

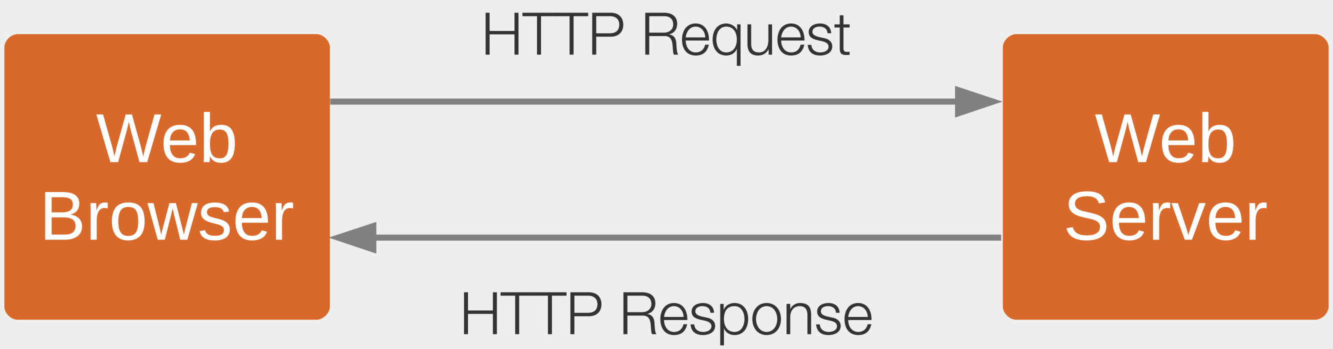
Overview

- **Talk:** Introduction to web programming
- **Exercises:** Recapping HTML, CSS and SASS
- **Talk:** Meerkat Frontend – A Python Flask Application
- **Exercise:** Editing Meerkat Frontend
- Summary

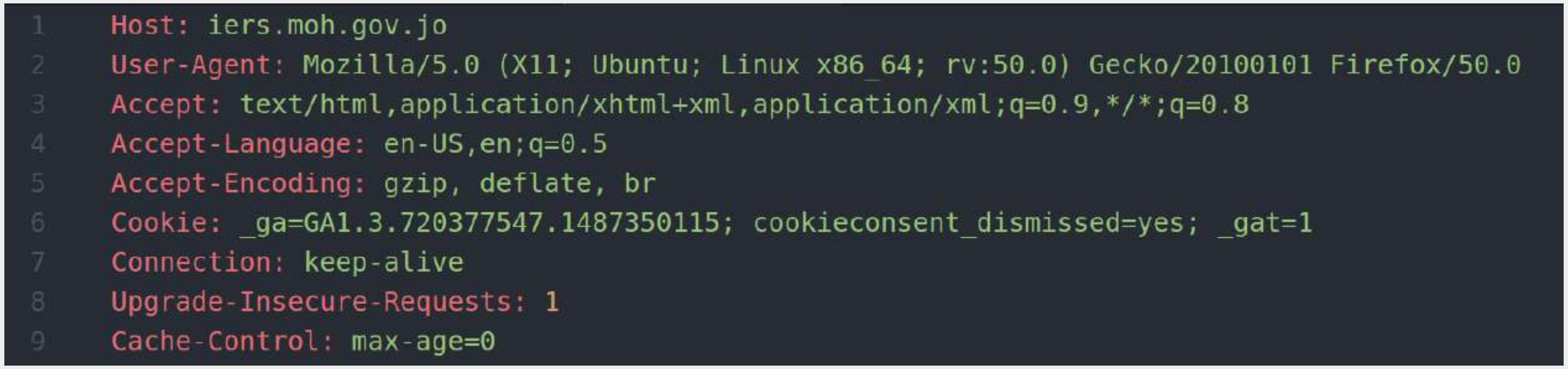
Introducing the internet



HTTP: Hypertext Transfer Protocol



Headers for a 'GET' request to iers.moh.gov.jo/en:

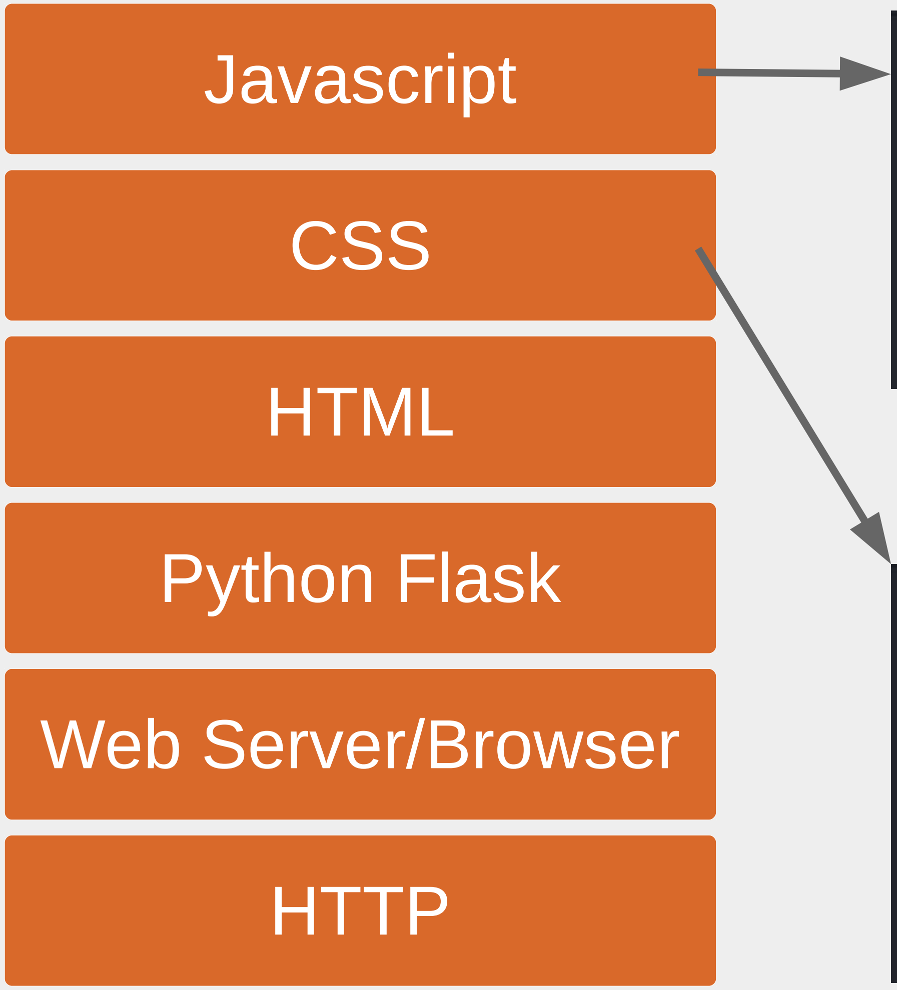


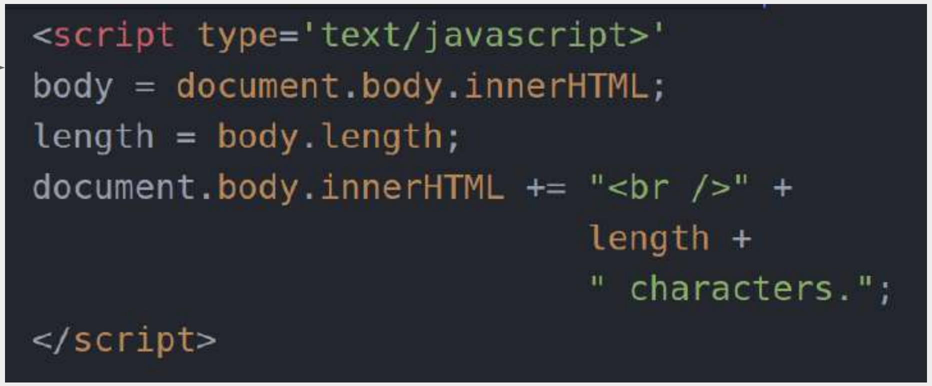
```
Host: iers.moh.gov.jo
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Cookie: _ga=GA1.3.720377547.1487350115; cookieconsent_dismissed=yes; _gat=1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

A typical web page response

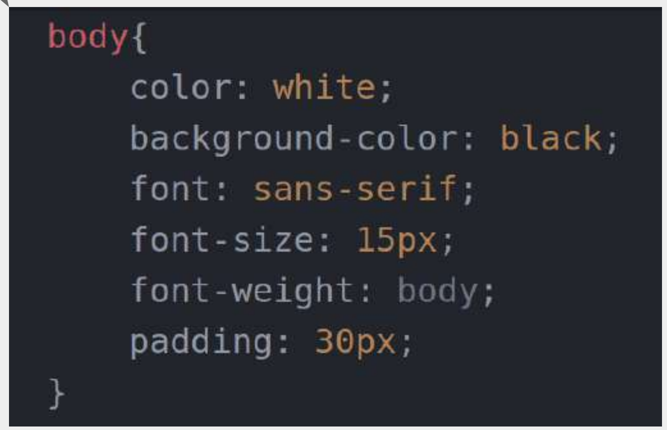
```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Wed, 22 Feb 2017 11:33:15 GMT
Server: nginx/1.10.3
Transfer-Encoding: chunked

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Jordan Public Health Surveillance</title>
    <link rel="stylesheet" type="text/css" href="/static/css/main.css" />
    <script type="text/javascript" src="/static/css/main.css" />
  </head>
  <body>
    Jordan Public Health Serveillance
  </body>
</html>
```

The web technology stack





```
<script type='text/javascript>'
body = document.body.innerHTML;
length = body.length;
document.body.innerHTML += "<br />" +
                            length +
                            " characters.";
</script>
```

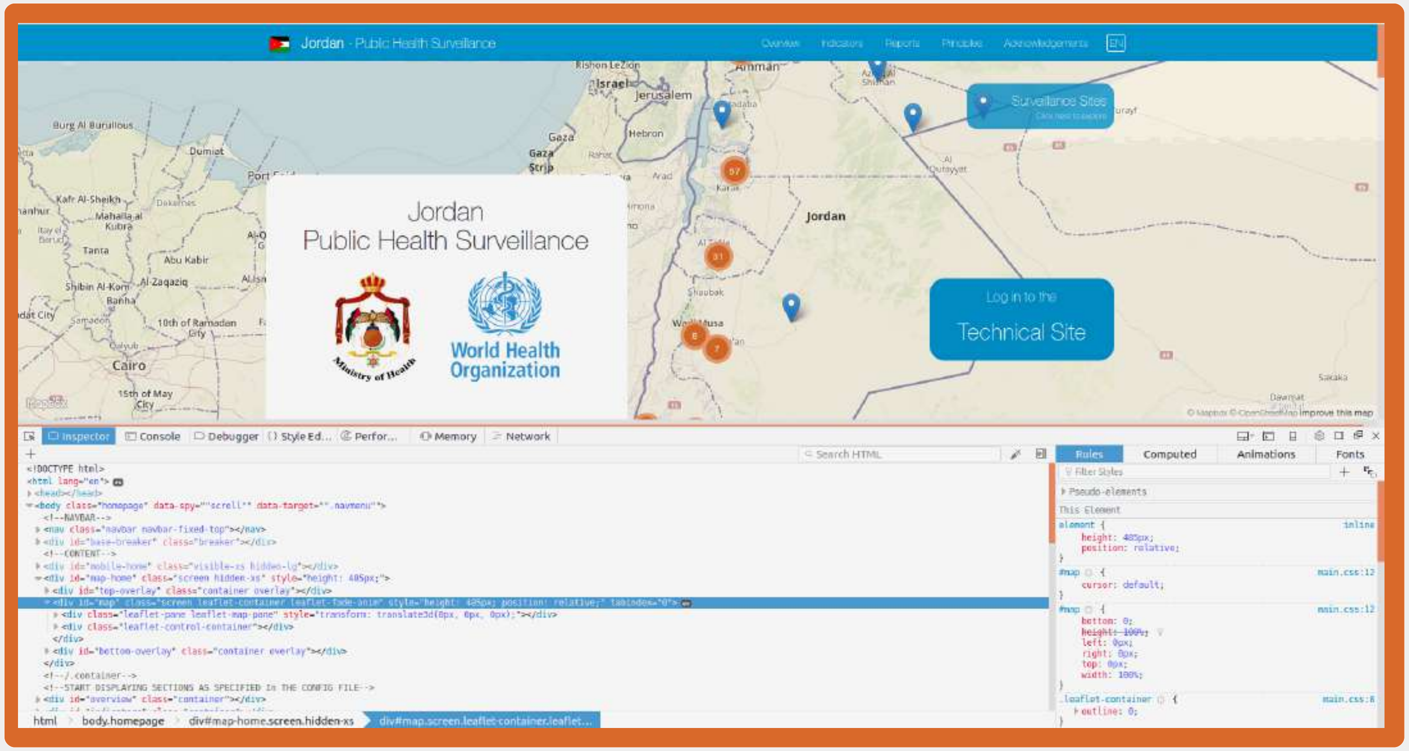


```
body{
    color: white;
    background-color: black;
    font: sans-serif;
    font-size: 15px;
    font-weight: body;
    padding: 30px;
}
```

Developer's tools

Use the developer's tools!

[right click] > [inspect]



```
<!DOCTYPE html>
<html lang="en">
▸ <head></head>
▾ <body class="homepage" data-spy=""scroll"" data-target="".navmenu"">
    <!--NAVBAR-->
  ▸ <nav class="navbar navbar-fixed-top"></nav>
  ▸ <div id="base-breaker" class="breaker"></div>
    <!--CONTENT-->
  ▸ <div id="mobile-home" class="visible-xs hidden-lg"></div>
  ▾ <div id="map-home" class="screen hidden-xs" style="height: 485px;">
    ▸ <div id="top-overlay" class="container overlay"></div>
    ▾ <div id="map" class="screen leaflet-container leaflet-fade-anim" style="height: 485px; position: relative;" tabindex="0">
      ▸ <div class="leaflet-pane leaflet-map-pane" style="transform: translate3d(0px, 0px, 0px);"></div>
      ▸ <div class="leaflet-control-container"></div>
      </div>
    ▸ <div id="bottom-overlay" class="container overlay"></div>
    </div>
    <!--/.container-->
    <!--START DISPLAYING SECTIONS AS SPECIFIED IN THE CONFIG FILE-->
  ▸ <div id="overview" class="container"></div>
```

html > body.homepage > div#map-home.screen.hidden-xs > div#map.screen.leaflet-container.leaflet...

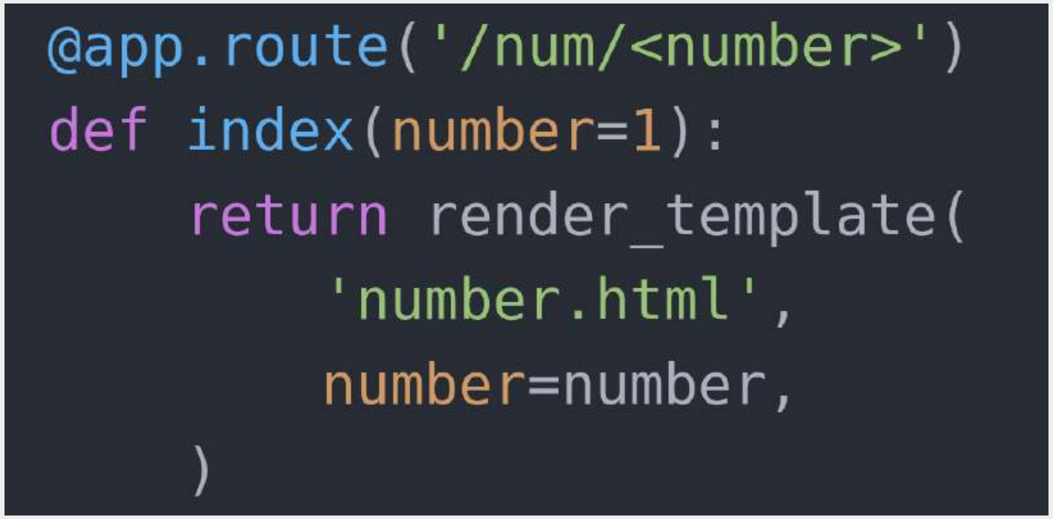
```
▸ Pseudo-elements
This Element
element {                                  inline
    height: 485px;
    position: relative;
}
#map {                                     main.css:12
    cursor: default;
}
#map {                                     main.css:12
    bottom: 0;
    ~~height: 100%;~~
    left: 0px;
    right: 0px;
    top: 0px;
    width: 100%;
}
.leaflet-container {                       main.css:8
  ▸ outline: 0;
}
```

Exercise: Content and Formatting

HTML and CSS

Meerkat Frontend
A Python Flask Application

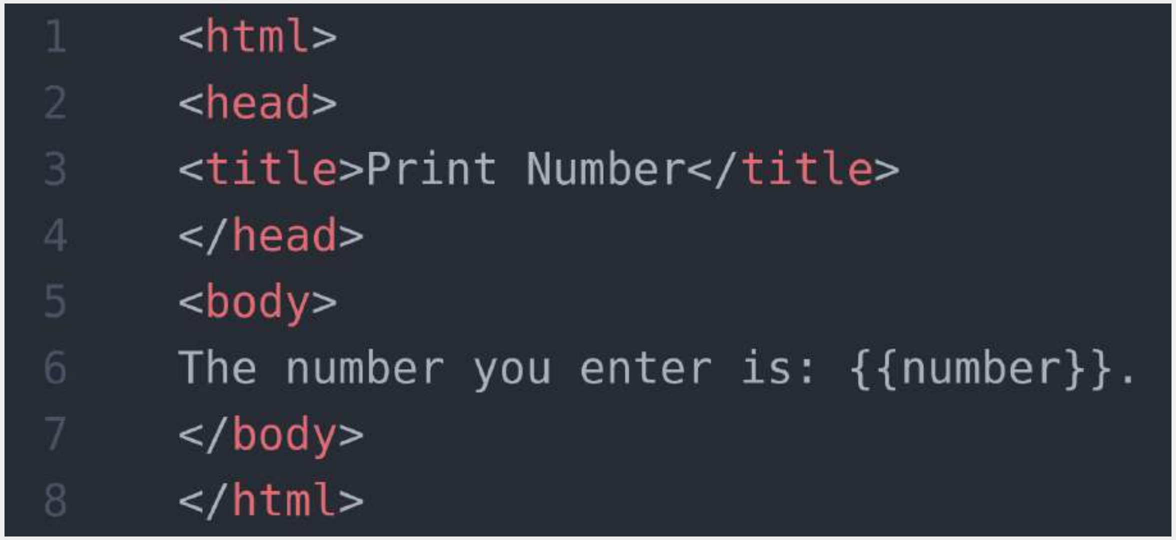
Python Flask



```
@app.route('/num/<number>')
def index(number=1):
    return render_template(
        'number.html',
        number=number,
    )
```

```
meerkat_frontend/
    setup.py
    node_modules/
    bower_components/
    ...
    meerkat_frontend/
        __init__.py
        views/
        templates/
        static/
        src/
        ...
```

Jinja2 and template rendering

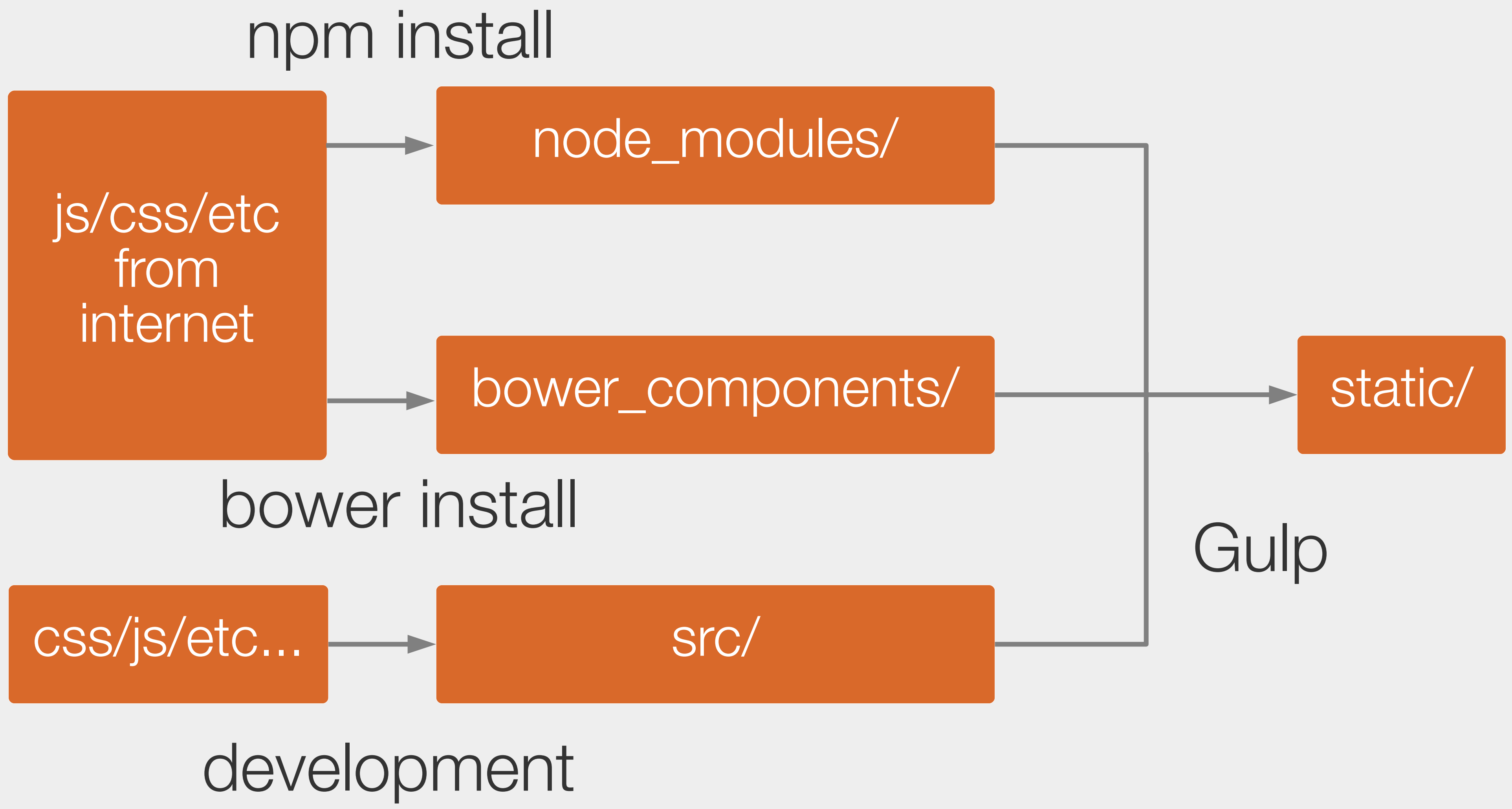


```
<html>
<head>
<title>Print Number</title>
</head>
<body>
The number you enter is: {{number}}.
</body>
</html>
```

- Print variables
- Loop
- Import html
- Nest html

and more...

The build process: Gulp, Bower & NPM



Exercise: Editing Meerkat Frontend

Getting to grips with the code

Summary

A web application uses many different technologies.

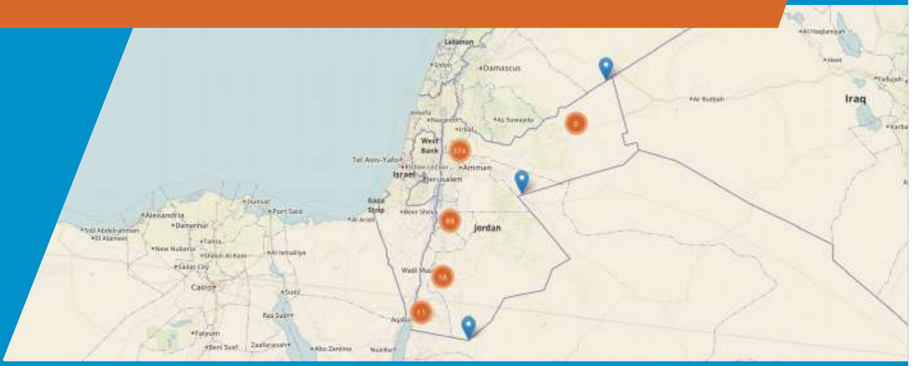
Meerkat Frontend requires understanding:
HTTP, Python, Jinja2, HTML, CSS (SASS), Javascript

Meerkat Frontend is built using Gulp, Bower and NPM

Countless tutorials/docs online to help you e.g...

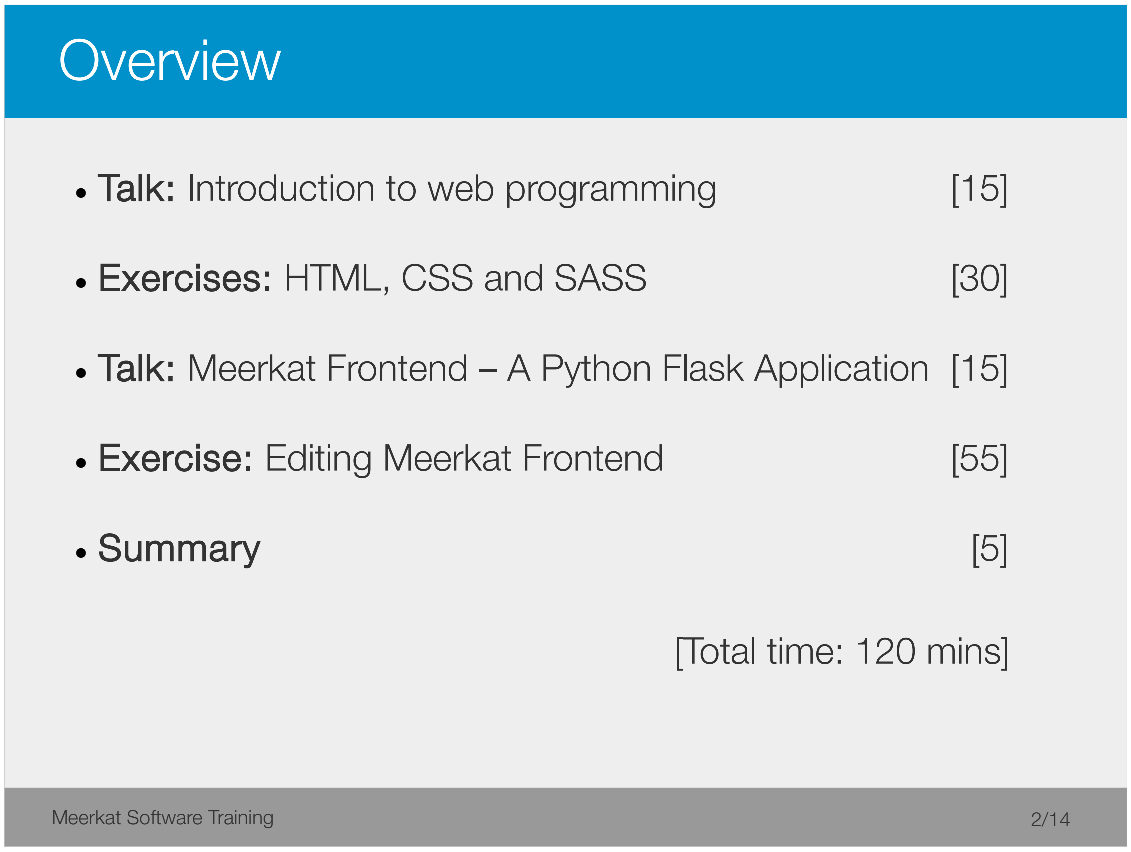
www.w3schools.com
flask.pocoo.org
jinja.pocoo.org
and of course: google.com

Jordan · Public Health Surveillance



Meerkat Software Training

Web Programming with Python



Overview

- **Talk:** Introduction to web programming [15]
- **Exercises:** HTML, CSS and SASS [30]
- **Talk:** Meerkat Frontend – A Python Flask Application [15]
- **Exercise:** Editing Meerkat Frontend [55]
- Summary [5]

[Total time: 120 mins]

KEY AIMS

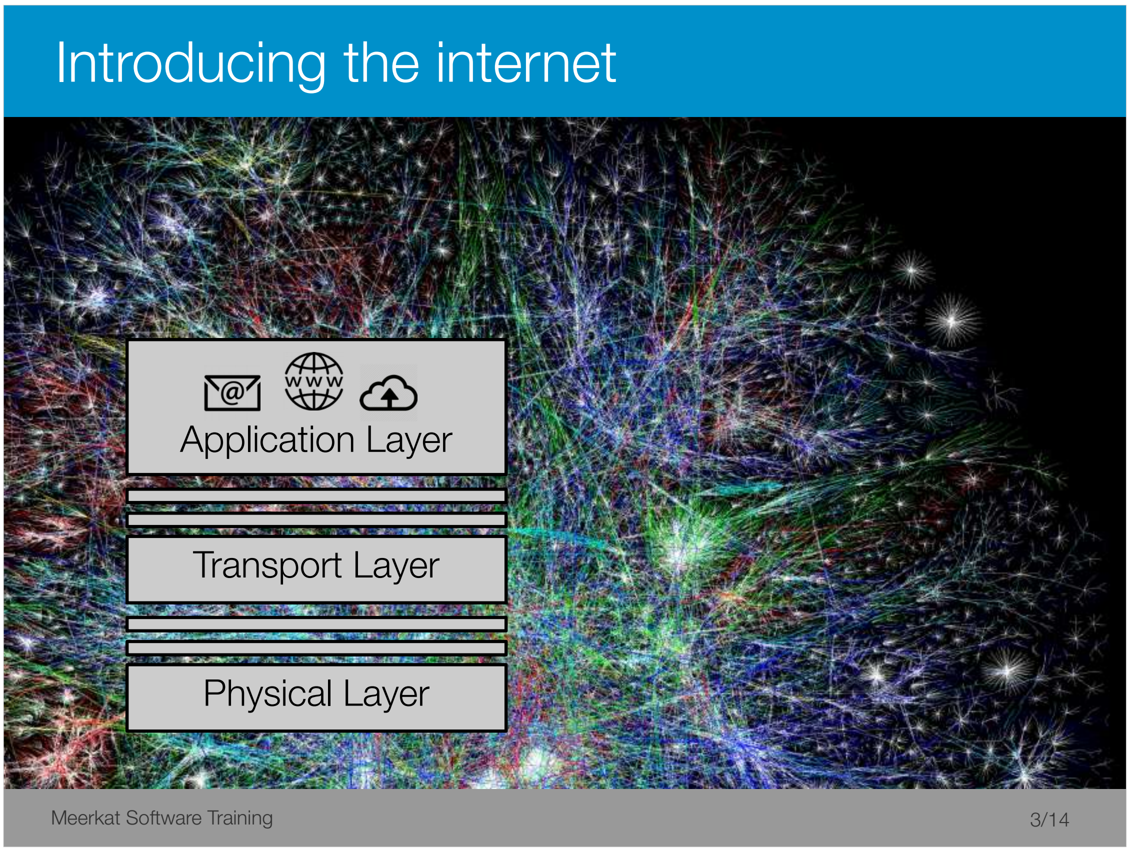
- Undersand that a web application is made up of many different technologies.

- Gain basic knowledge of the technologies used in meerkat_frontend and how they talk to each other: Web servers, HTML, CSS and SASS, Javascript, Jinja2 & Python Flask.

- Introduction to meerkat_frontend structure and build process.

- Experience making simple edits of CSS, HTML, JINJA2.

- Understand how to learn more about this.



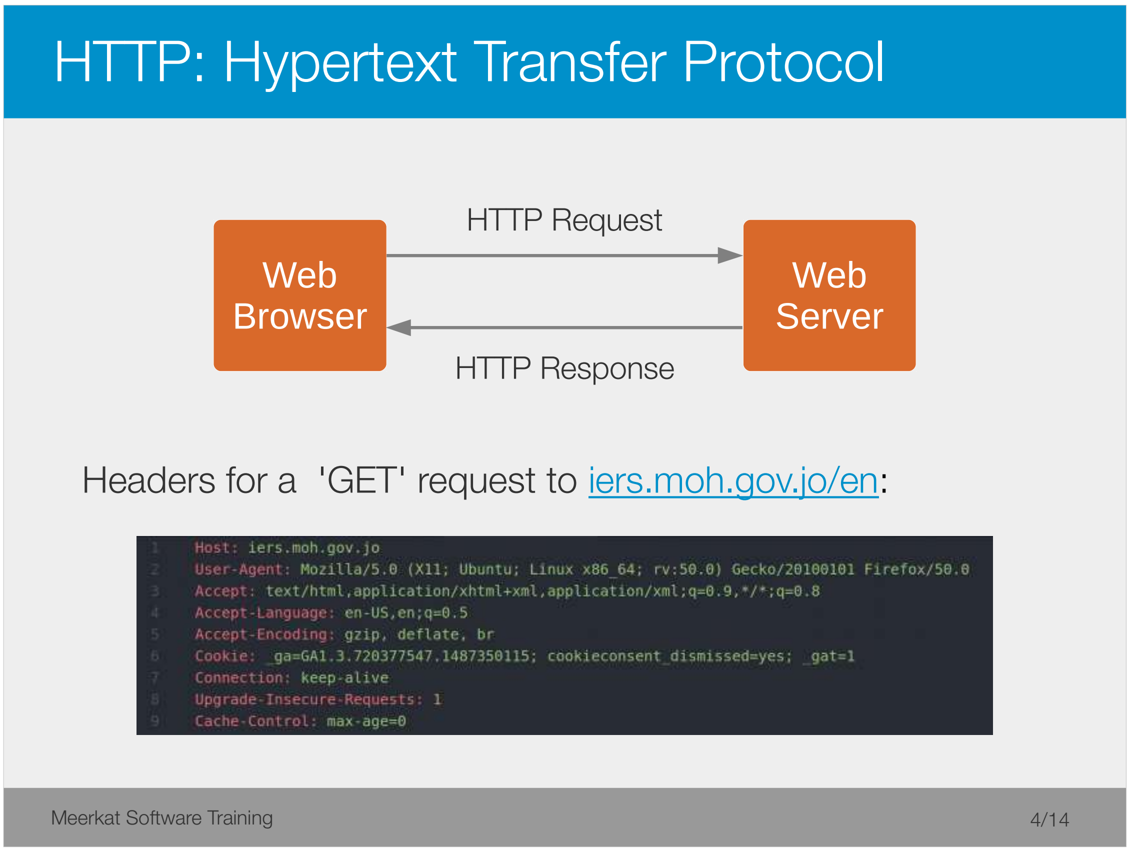
Introducing the internet

Application Layer
Transport Layer
Physical Layer

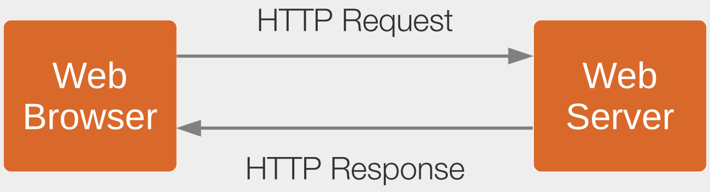
Meerkat Software Training

3/14

- Probably best thought of as a global network of computer networks.
- And all these networks agree on a defined protocol, a set of rules, or a recipe for communicating with each other – we call this the “Internet Protocol Suite” TCP/IP (Transmission Control P/ Internet P)
- The internet is made of up many layers of engineering and infrastructure, from the physical layer – the cabling and hardware devices at the bottom, to the protocols in the transport layer determining how information should be transported between devices, to the application layer at the very top, from which spills out all of the creative uses we have made for the internet:
- Email, Domain names, File Transfer, Internet Radio, Instant Messaging and of course the World Wide Web... which is what we're focusing on today.



HTTP: Hypertext Transfer Protocol

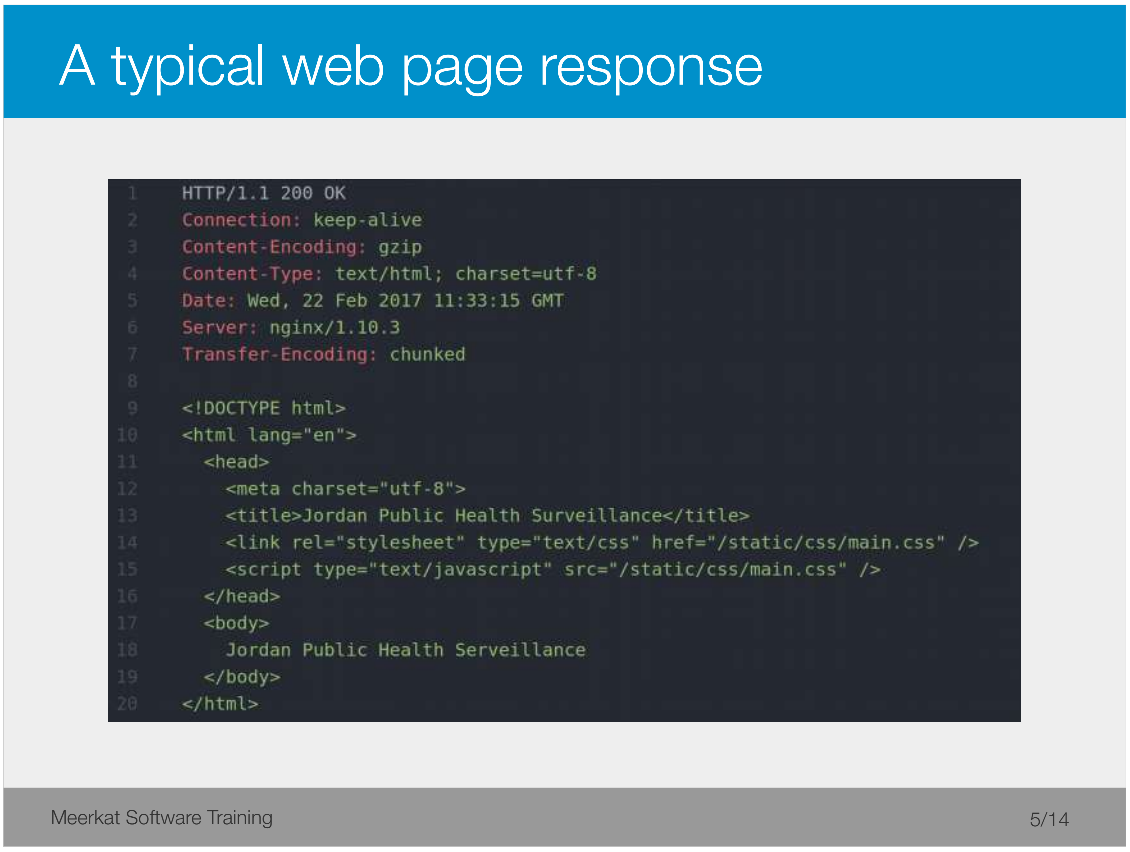


Headers for a 'GET' request to iers.moh.gov.jo/en:

```
Host: iers.moh.gov.jo
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Cookie: _ga=GA1.3.720377547.1487350115; cookieconsent_dismissed=yes; _gat=1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Meerkat Software Training 4/14

- The world wide web is based upon a protocol called Hypertext Transfer Protocol or HTTP.
- Hypertext is structured content that is linked together using logical links, or hyperlinks.
- HTTP specifies how we should transfer this structured content between a “Web Server” application and a “web browser” application.
- When you type iers.moh.gov.jo into your web browser, the bowser assembles this little bit of text, using the rules of http, and pushes it into the internet.
- This request reaches the web server we have set up with Amazon, and the web server responds with hypertext that your browser knows how to display.
- Our job as web developers is to define the content of that response...

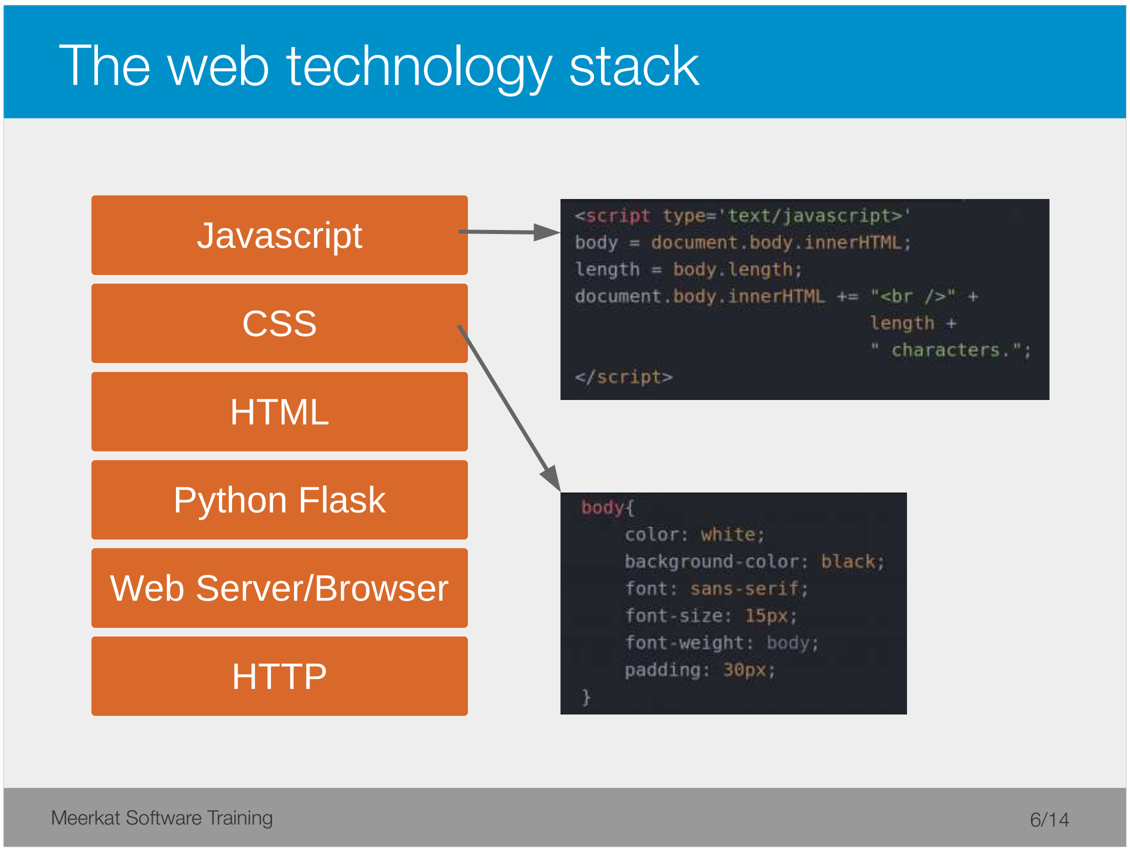


A typical web page response

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Wed, 22 Feb 2017 11:33:15 GMT
Server: nginx/1.10.3
Transfer-Encoding: chunked

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Jordan Public Health Surveillance</title>
    <link rel="stylesheet" type="text/css" href="/static/css/main.css" />
    <script type="text/javascript" src="/static/css/main.css" />
  </head>
  <body>
    Jordan Public Health Serveillance
  </body>
</html>
```

A typical response would be a HTML file with HTTP headers. HTML stands for Hyper Text Markup Langauge, is the way we define our hyper text content.

This file has a number of ways of then referencing other files which are separately requested from the server. Images, Javascript Files, CSS files,etc...



The web technology stack

```
<script type='text/javascript>'
body = document.body.innerHTML;
length = body.length;
document.body.innerHTML += "<br />" +
                          length +
                          " characters.";
</script>
```

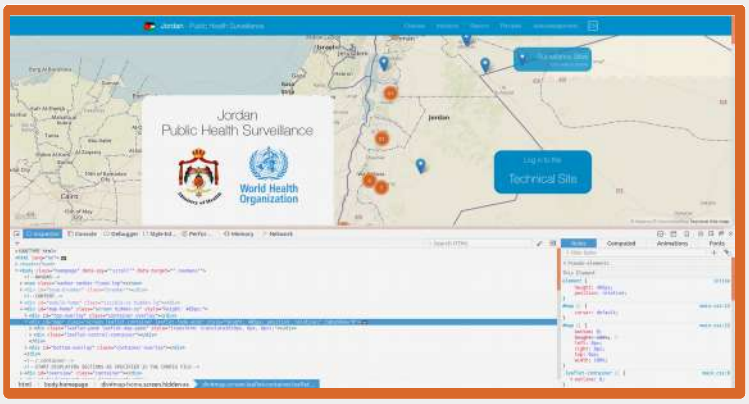
```
body{
      color: white;
      background-color: black;
      font: sans-serif;
      font-size: 15px;
      font-weight: body;
      padding: 30px;
}
```

So there are lots of different types of files and technology involved in web programming. Let's look at a typical web development stack:

- HTTP determines communication between web server and web browser.
- Web Server Distributes files for Web browser to view.
- Server based programming language such as PHP or python can to interperate the http requests coming in and dynamically create the file responses going, this involves creating...
- HTML files, which are returned and reference static files typically including images and ...
- Cascading Style Sheets (or CSS files) which specify how the HTML file should be visually displayed.
- and Javascript which is a web-bowser based programming language for editing content.

Developer's tools

Use the developer's tools!
[right click] > [inspect]

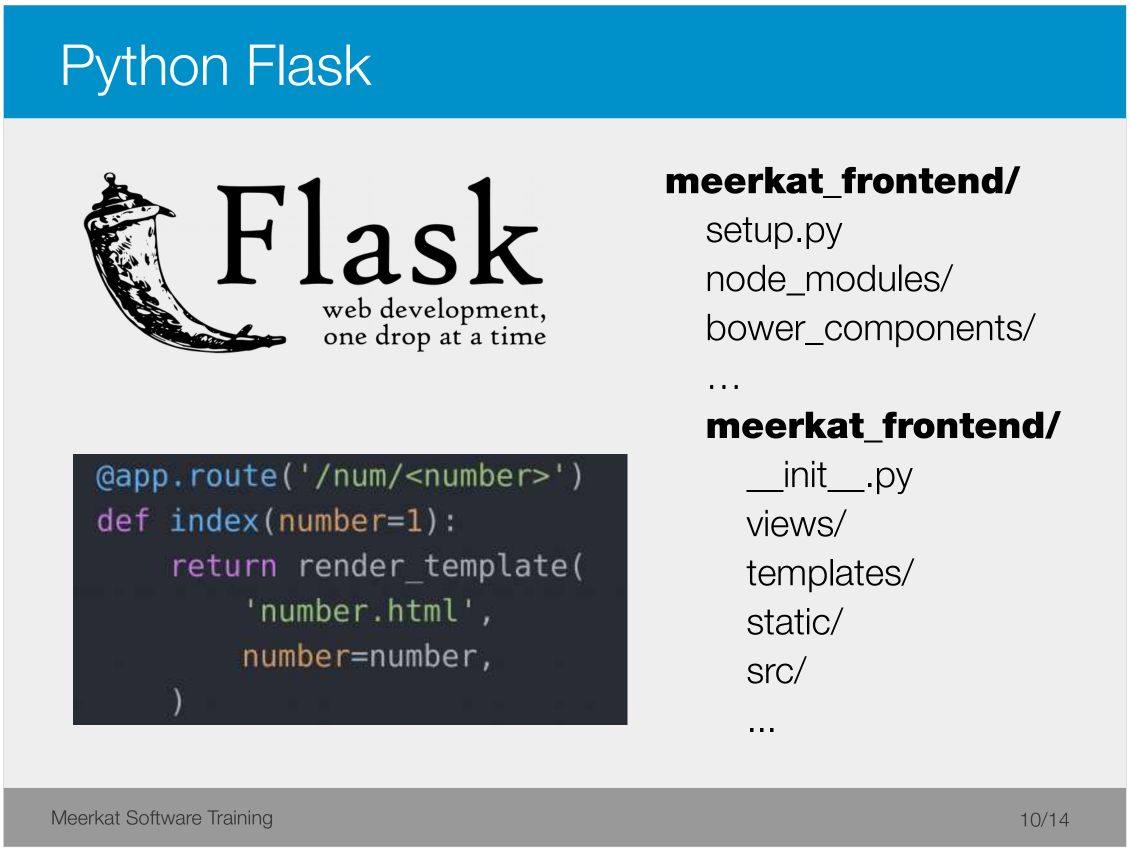


Exercise: Content and Formatting

HTML and CSS

We're going to begin by familiarising ourselves with some of the very basics of html and css.

Meerkat Frontend
A Python Flask Application



Python Flask

```
@app.route('/num/<number>')
def index(number=1):
    return render_template(
        'number.html',
        number=number,
    )
```

meerkat_frontend/
- setup.py
- node_modules/
- bower_components/
- …
- **meerkat_frontend/**
  - __init__.py
  - views/
  - templates/
  - static/
  - src/
  - ...

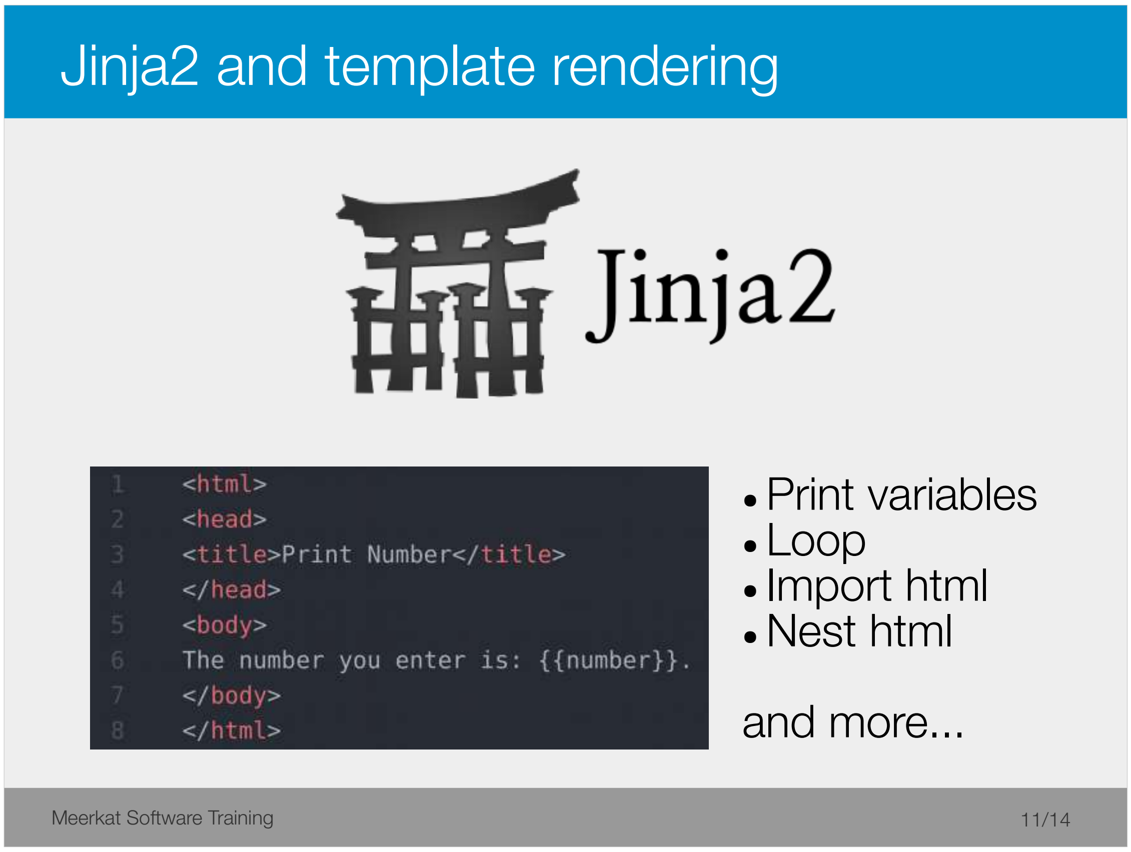
Meerkat Software Training 10/14

As I said before, we use Python as a server-side programming language to interpret and dynamically assemble responses from the web server.

There are a few pieces of Python software we could use to do this, we have chosen Python Flask because it is light weight, giving lots of flexibility without significant overheads.

Meerkat Frontend is structured first and foremost as a Python package, with an __init__.py file and folder for python views and a folder for static resources.

Amongst many other things, python flask allows us to specify functions that are called when a http request is made to a particular url. It allows us to encode variables into a page's url, process the url's information and from it dynamically assemble a html file to return. This is what we'll be doing in the next exercise.



Jinja2 and template rendering

Jinja2

```
<html>
<head>
<title>Print Number</title>
</head>
<body>
The number you enter is: {{number}}.
</body>
</html>
```

- Print variables
- Loop
- Import html
- Nest html

and more...

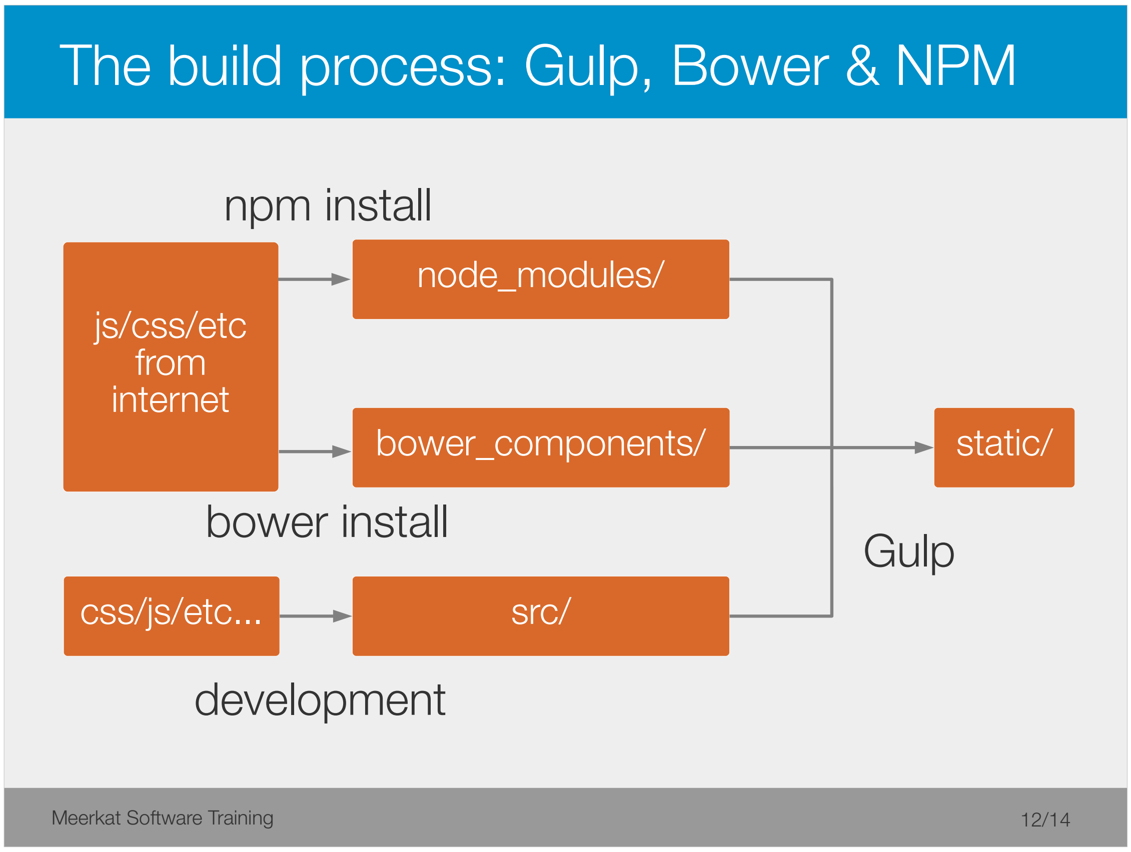
Meerkat Software Training 11/14

So how does python flask dynamically create html files?

Python flask provides a function called render template. This function takes a Jinja2 template and some data that you define, and combines the two into a static html file response to be returned by the web server.

A Jinja2 template looks a lot like a html file, except is also uses it's own language to do some limited editing of the html content using the data structures that you supplied to the render_template function.

You can print variables out, loop through variables, and perform conditional statements, “if A is true, then add B to the html file.



The build process: Gulp, Bower & NPM

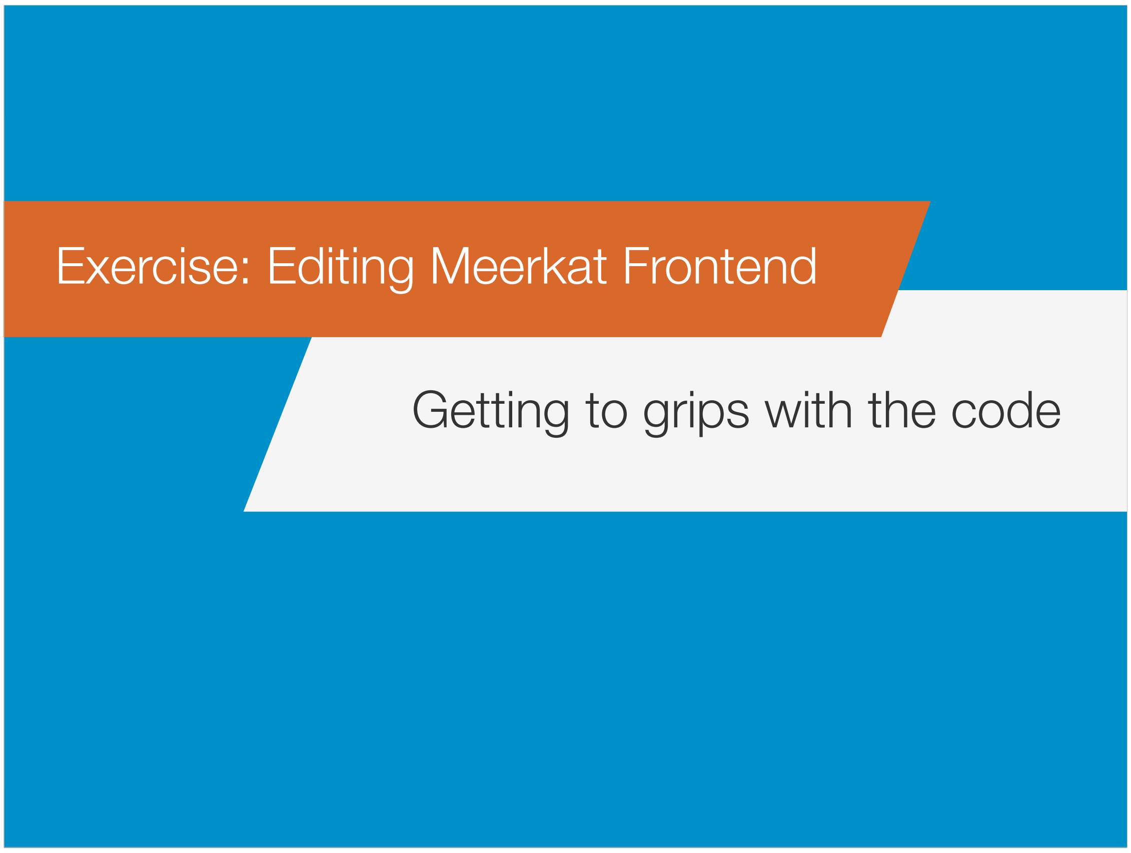
These static resources, such as css, javascript, images etc... are drawn from a mixture of resources developed by us and by other people.

This introduces a problem: How do we install and manage so many different resources efficiently?

Well the answer is to use “Bower” and “Node Package Manager” to install and manage the many packages created by other people.

A third piece of software called Gulp to merge their software with our own software in a structure that can be referenced by our html responses.

IMPORTANT: 'static' is built by Gulp from 'source'.
This means that in order to make changes to...



Exercise: Editing Meerkat Frontend

Getting to grips with the code

Summary

A web application uses many different technologies.

Meerkat Frontend requires understanding:
HTTP, Python, Jinja2, HTML, CSS (SASS), Javascript

Meerkat Frontend is built using Gulp, Bower and NPM

Countless tutorials/docs online to help you e.g...

www.w3schools.com
flask.pocoo.org
jinja.pocoo.org
and of course: google.com