Web Technologies

Fundamentals

Alex Kavvos, Stuart Gray, **Thomas Bale, Marceli Wac.**

Purpose & Objectives

Provide a focused practical introduction to web technology fundamentals. These materials directly support your Software Engineering group project. You will:

1. Understand the functions of **clients, servers, HTTP** protocol, **RESTful APIs** and **HTML**.
2. Be able to **write and adapt** a correct HTML page with associated CSS.
3. Be able to **add interactivity** to the same web page via use of JavaScript.
4. Be able to add webpage content **read interactively** from a static CSV and JSON file.
5. Be able to **dynamically** amend content called correctly from an external RESTful API.
6. Understand the role of web frameworks, such as **React and Angular**, in streamlining web development processes.

OK, so how do we do *that* ?


Month Day Year Hour Minute
JUN 01 1993 04 07

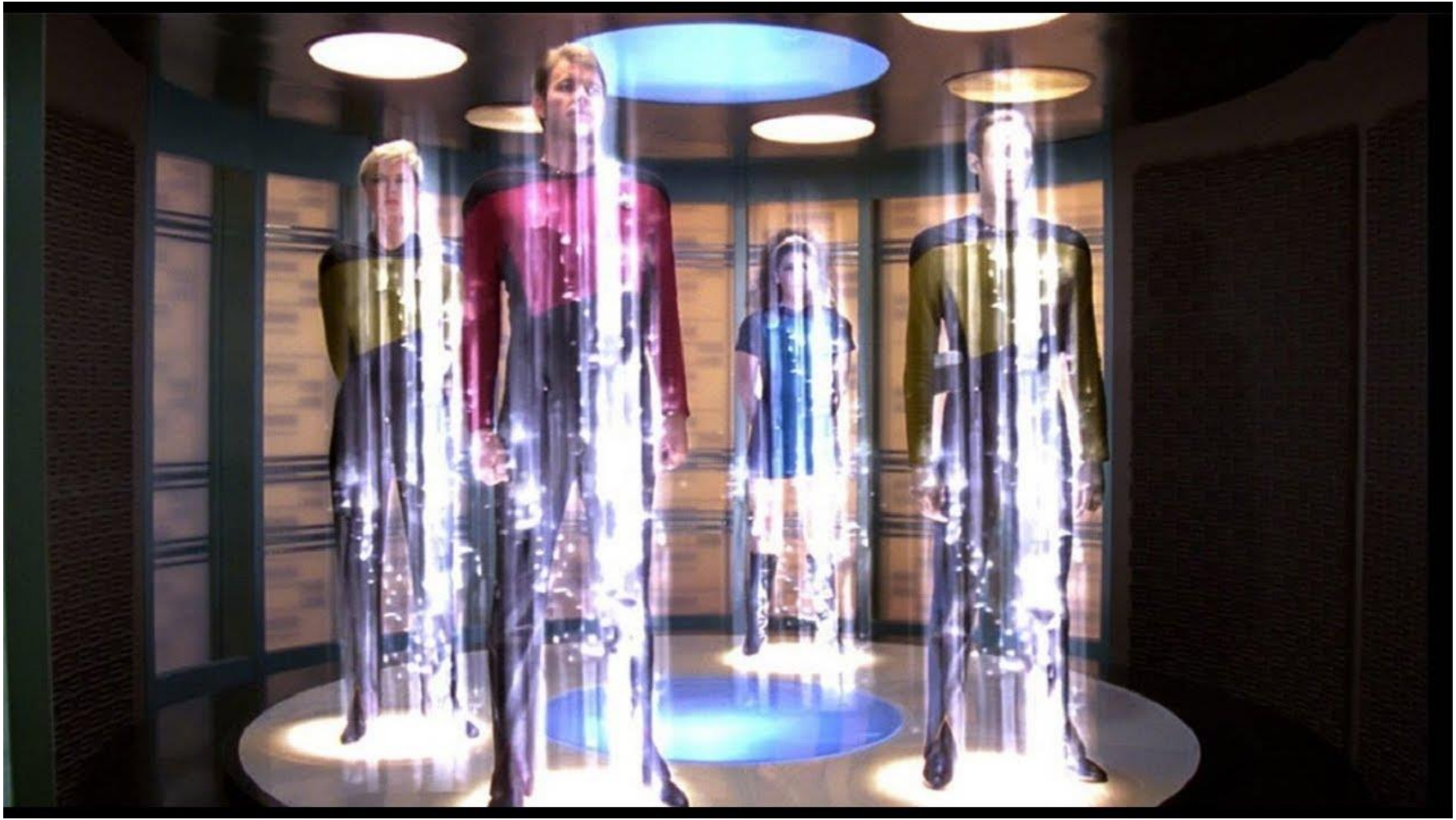

DMC
OFF

Approach

- Focused on ‘hand building’ a simple dynamic ‘web app’
- 7 x 30 minute lecture & walkthrough videos
- 4 hours of TA supported labs
- Practical worked ‘web application’ from scratch (no frameworks!)
- Demonstration/walkthrough of the same project built with React
- Adapt & commit project to your personal git repo to improve your professional profile

Overview

1. **TCP/IP** (point to point communication protocol)
2. **HTTP** (web application protocol)
3. **HTML** (page data format)
4. **CSS** (styling instructions)
5. **DOM** (common API allowing pages to be browser neutral)

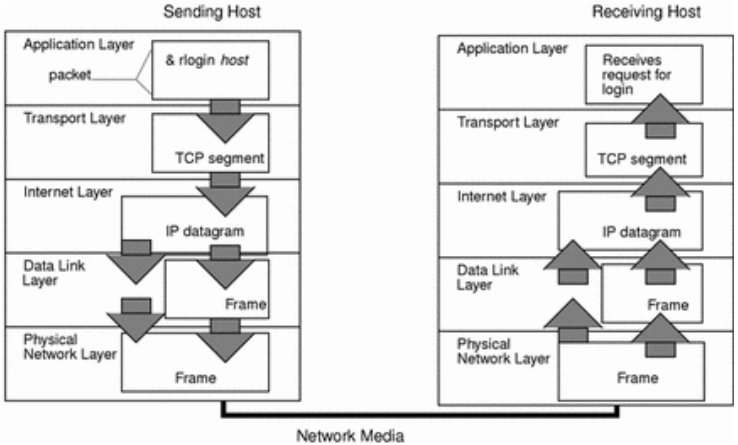


TCP/IP

- TCP/IP stands for **Transmission Control Protocol/Internet Protocol** and is a standard Internet communications protocol that allow digital computers to communicate over long distances.

TCP/IP is great because:

- **Is a very reliable, ordered & error checked method of transferring data** (we sacrifice time for reliability)
- Allows multiplexing (multiple sources of data to be transferred at the same time)
- Creates a ‘virtual circuit’ between the sender and receiver



HTTP

- **Hypertext Transfer Protocol** is an application layer protocol for distributed, collaborative, hypermedia information systems. (Wikipedia)

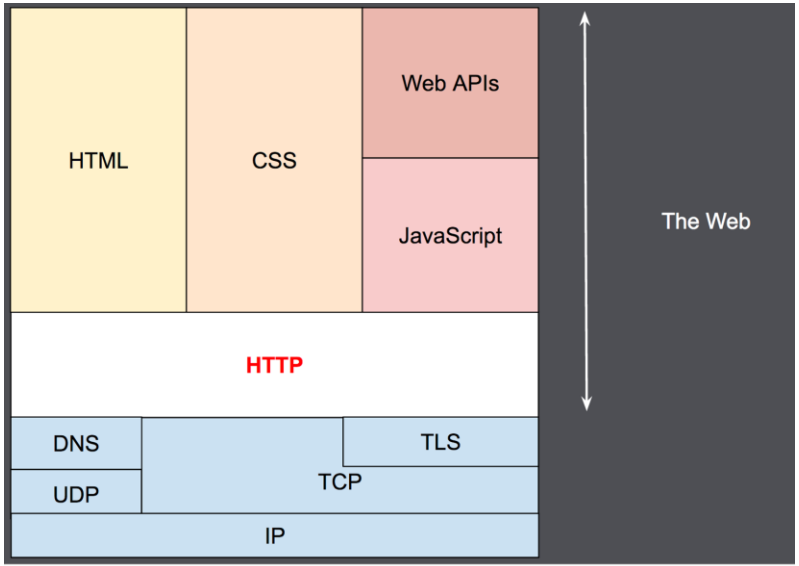
HTTP

The user makes a request...

… the **request** is sent to a server...

… the server handles it and provides an answer...,

This answer is called the ***response***.

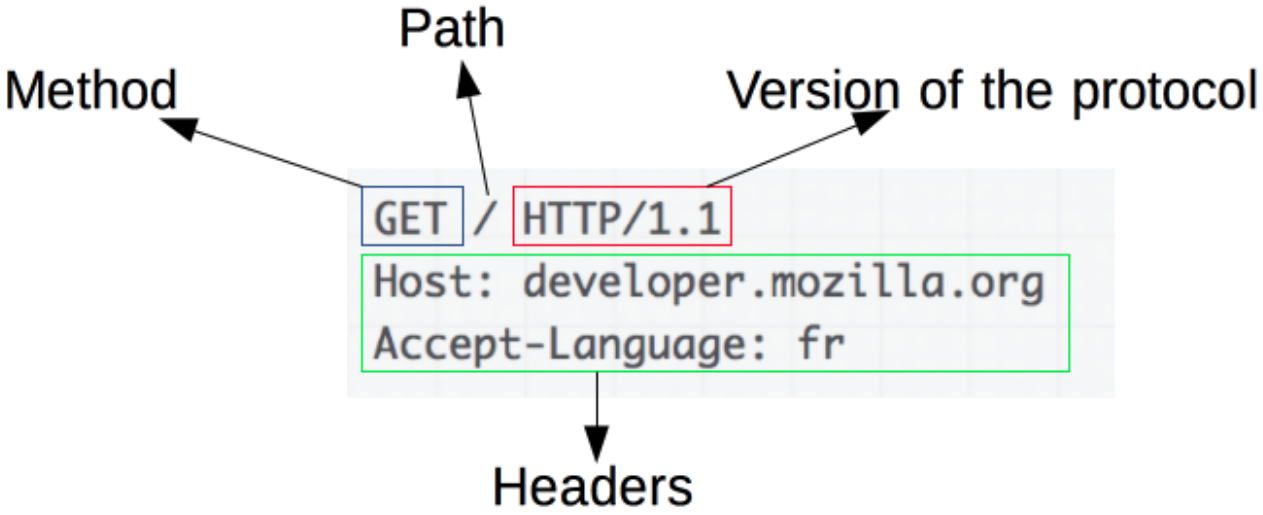


HTTPs:

- Human readable
- Extensible (think headers)
- HTTP/2 can multiplex messages over a single TCP connection

Credit: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

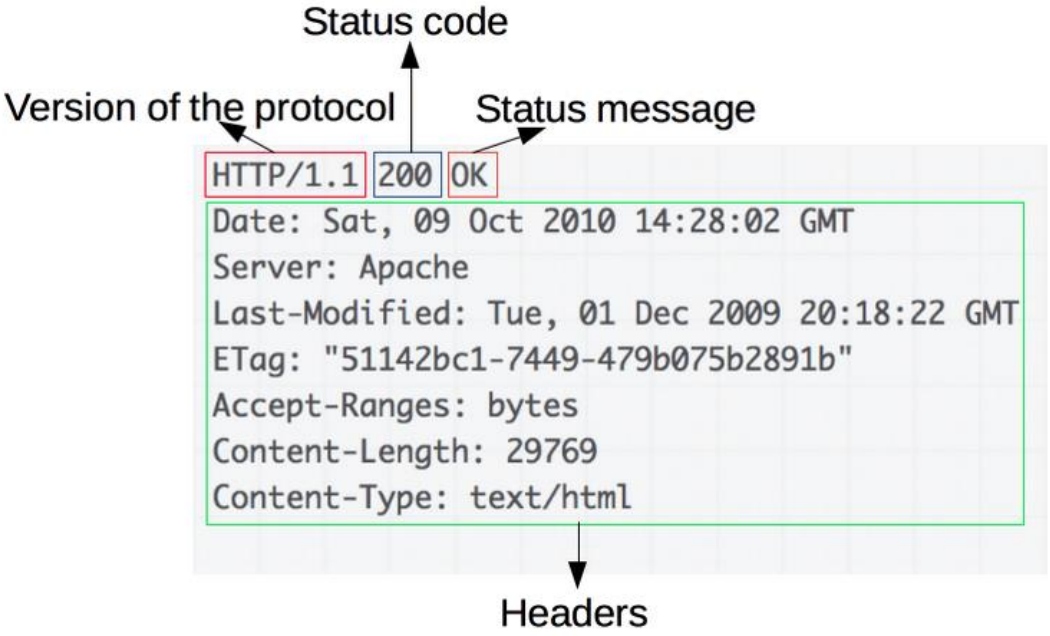
HTTP



Credit: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

What kind of response?

HTTP



Credit: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

HTML & CSS

“To present a Web page, the browser sends an original **request** to **fetch** the HTML document that represents the page. It then **parses this file**, making additional requests corresponding to execution scripts, layout information (CSS) to display, and sub-resources contained within the page (usually images and videos). The Web browser then **mixes these resources** to present to the user a **complete document**, the Web page.”

Credit: Mozilla Firefox Foundation

HTML & CSS

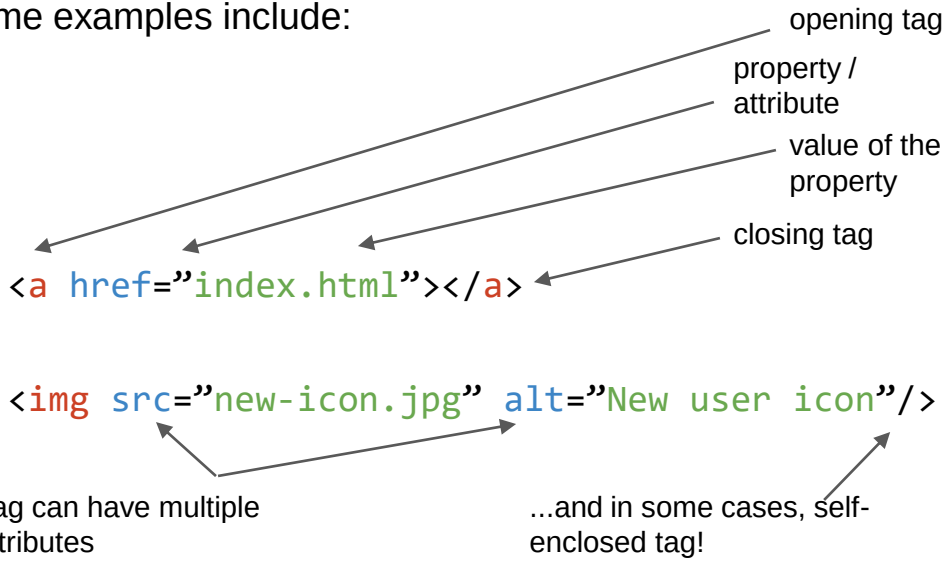
- **HTML** stands for Hyper Text Markup Language. **HTML** is the standard markup language for creating Web pages. **HTML** describes the structure of a Web page. (Mozilla.org)
- **CSS** stands for Cascading Style Sheets and is the language we use to style a Web page. (w3.org)

What is HTML used for?

HTML is a language used to **structure the information in a form of a 'website'**. It contains both the **content** (such as the body of a blog post or its title) and **metadata** (such as the name of the author and the website's title).

The primary building block of HTML is a **tag**. Some examples include:

```
<html lang=”en”>
<head>
    <meta type=”charset” value=”utf-8”>
    <!-- This is a comment -->
</head>
<body>
    <a href=”https://bristol.ac.uk/”>
        <h1>Visit University’s page</h1>
    </a>
</body>
</html>
```



What is CSS used for

CSS is the language used to **style the HTML pages**. It **applies the styles to the elements**, based on their relative position within the page, classes or ids, but also states and attribute values.

This means that selecting the element with CSS can be done at a very granular level. Some examples of **CSS selectors** include:

```
.someClass {
   color: red;
}

#someId {
   color: blue;
}

p {
   color: black;
}
```

```
h1.someClass {
   color: yellow;
}

@media(min-width: 768px) {
   .tabletsAndUp {
      color: pink;
   }
}
```

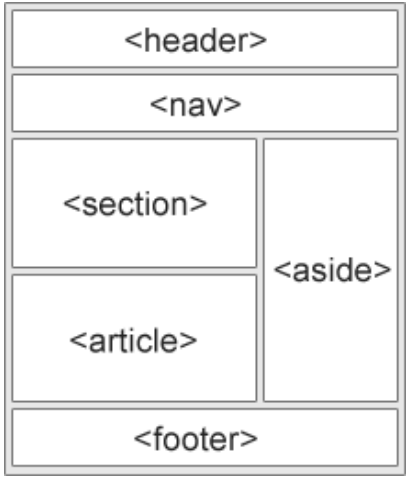
```
button:hover {
   color: gray;
}

a[href^="http://"] {
   color: green
}
```

Semantic HTML

Recall that HTML is used to both **describe** and **structure** the information. As such, while some of the tags are used for description of the content, others can be used purely for structuring the document. Using appropriate tags for certain types of content such as figures and their captions is important, because it allows the browser and search engines to extract this information and provide additional functionalities.

Examples of **semantic** HTML tags include:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



Non-responsive

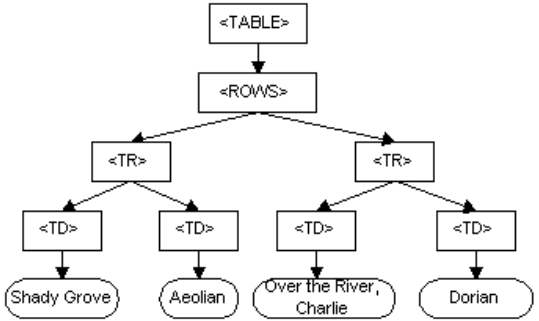
Responsive

DOM

- The **Document Object Model** (DOM) is a programming API for HTML and XML documents. (w3.org)
- **DOM** is a model of a document with an associated API for manipulating it. **HTML** lets you represent a certain kind of **DOM** in text (as we already know).
- The DOM is a logical structure which provides a standard programming interface that can be used in a wide variety of environments and applications

DOM

- The DOM is vendor and platform neutral
- Documents have a logical structure which is very much like a tree
- Nodes in the ‘forest’ do not represent a data structure, they represent **objects**, which have functions and identity



Activity 1

- ❏ Watch the walkthrough video
- ❏ Using a text editor write and test a basic web page using HTML and CSS which meets the stated guidelines (semantic HTML).
- ❏ Copy and paste the data from the JSON file.
- ❏ Present the data in new ways not covered in the walkthrough: introduce new HTML tags and CSS.
- ❏ Extension: add Bootstrap styling from CDN (content delivery network).

Interactivity

Javascript


Month Day Year Hour Minute
NOV 01 1996 00 01

Javascript

- Javascript can change HTML content
- It can also change HTML attributes
- ...And CSS styles…!
- … and it can hide and show things...!!

Javascript

- We insert JS between the <script> </script>
- You can put it anywhere in the HTML (e.g. head and body)
- We use functions and a function could be called when an event occurs (e.g. onclick)
- Scripts can be placed in external files… and there are advantages to doing so

Javascript: displaying output

- Writing into an HTML element, using **innerHTML**.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

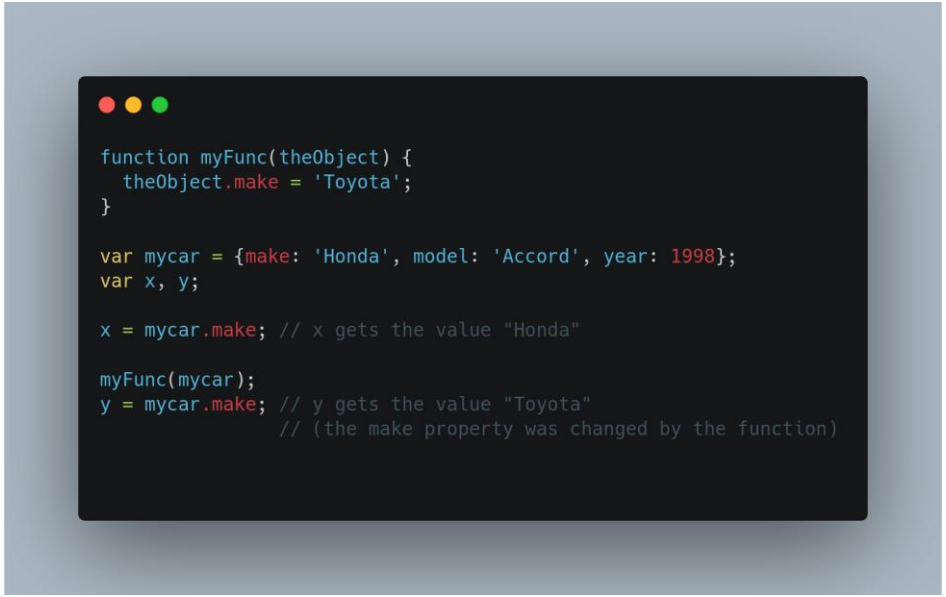
Javascript fundamentals:

- Optional: Complete Javascript Tutorial if you want a step by step guide
- **Bonus**: you know C already … so you will recognise a lot.
- Remeber - you **know a lot** more already than you think!

https://www.w3schools.com/js/default.asp

Javascript demonstration:

- Lots of similarities to C
- Interpreted vs Compiled
- No memory management



```
function myFunc(theObject) {
  theObject.make = 'Toyota';
}

var mycar = {make: 'Honda', model: 'Accord', year: 1998};
var x, y;

x = mycar.make; // x gets the value "Honda"

myFunc(mycar);
y = mycar.make; // y gets the value "Toyota"
                // (the make property was changed by the function)
```

Activity 2

- ❏ Watch the walkthrough video
- ❏ Make your applications interactive via a load data button event and manipulate data in the JSON data protocol.
- ❏ Extension: Write a ‘save data’ button to export the page data (pets only) as a CSV/JSON file.

Dynamic loading

RESTful APIs


Month Day Year Hour Minute
JAN 01 2008 00 01

RESTful APIs

Representational state transfer (**REST**) is a software architectural style which uses a subset of HTTP.

1. Client-server separation of concerns
2. Stateless
3. Cacheability
4. Uniform interface

HTTP method	Description
GET	Get a representation of the target resource's state.
POST	Let the target resource process the representation enclosed in the request.
PUT	Set the target resource's state to the state defined by the representation enclosed in the request.
DELETE	Delete the target resource's state.

Try one out!  https://petstore.swagger.io/#/store/placeOrder

RESTful APIs

1. The endpoint
2. The method
3. The headers
4. The data (or body)

Activity 3

- ❏ Watch the walkthrough video
- ❏ Now adapt JavaScript to now make a properly a formed RESTful API GET call for data.
- ❏ Extension: Adapt the JavaScript again to POST data to the API.

Web Application Frameworks

React

Purpose & Objectives

Provide a focused practical introduction to web technology fundamentals. These materials directly support your Software Engineering group project. You will:

1. Understand the functions of **clients, servers, HTTP** protocol, **RESTful APIs** and **HTML**.
2. Be able to **write and adapt** a correct HTML page with associated CSS.
3. Be able to **add interactivity** to the same web page via use of JavaScript.
4. Be able to add webpage content **read interactively** from a static CSV and JSON file.
5. Be able to **dynamically** amend content called correctly from an external RESTful API.
6. Understand the role of web frameworks, such as **React and Angular**, in streamlining web development processes.

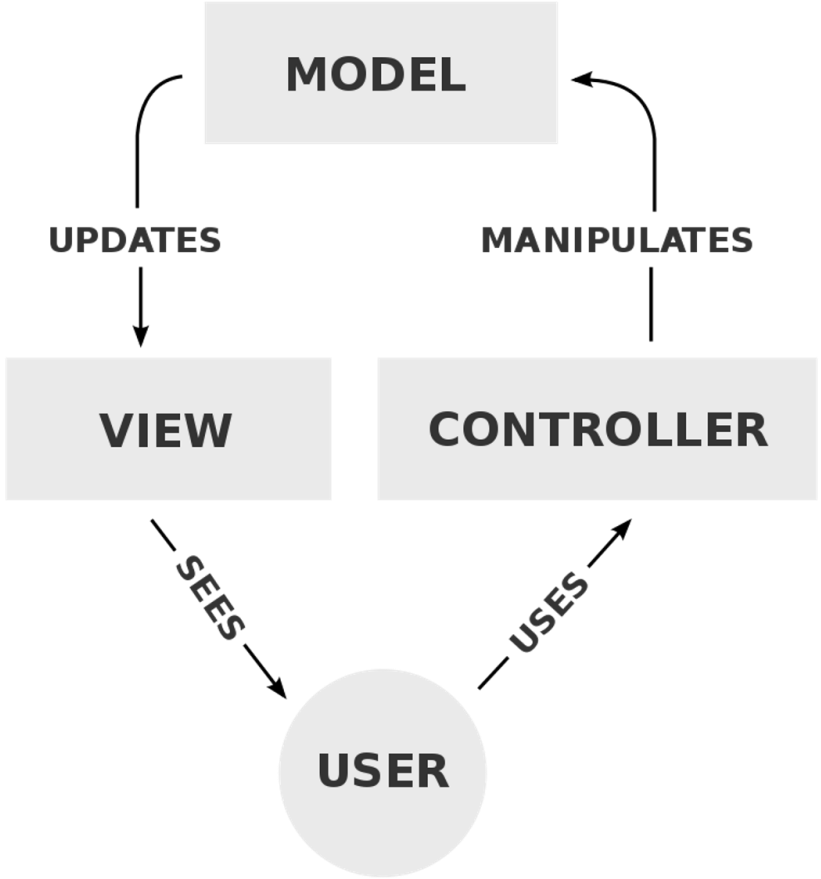

Month Day Year Hour Minute
MAY 29 2013 00 01

Advantages of using frameworks

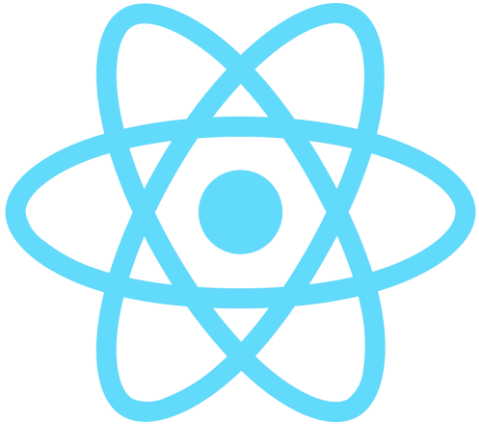
1. Open Source
2. Manual tasks are automated (e.g. data binding)
3. Performance
4. Security
5. Developer community
6. Documentation
7. Better developer experience
8. Support from cloud and hosting providers

Disadvantages of using frameworks

1. Learning curve
2. Can be heavy for simple tasks
3. Potential known vulnerabilities
4. Developer expertise/cost needed to maintain (vs Wordpress)
5. Fixed programming paradigm


MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER

Feature	Angular package	React library
Data binding, dependency injection (DI)	@angular/core	MobX
Computed properties	rxjs	MobX
Component-based routing	@angular/router	React Router v4
Material design components	@angular/material	React Toolbox
CSS scoped to components	@angular/core	CSS modules
Form validations	@angular/forms	FormState
Project generator	@angular/cli	React Scripts TS



Activity 4

- ❏ Watch the walkthrough video
- ❏ Now run the React example of the same petstore.
- ❏ Extension: Re-write the existing HTML-CSS-JS website as React website.

Activity 4: Web frameworks

We are going to reuse the code and create the application from scratch using the React framework. This activity is the most challenging as it makes use of a wider set of tooling and relies on new concepts. You may therefore need to read around how to use React (see reading list for materials). Based on your prior experience you may wish to use the boilerplate code provided as a start or begin again. You will find two versions of the source file directories:

1. **web-softwaretools-plain:** A plain version which is similar to the extended output of Activity 3
2. **web-softwaretools-react:** A fully re-written React version of the Pet-store application.

Complete these steps:

1. Watch the walkthrough video.
2. At this point, you will need to install node if you have not already! You can access NodeJS's website here: https://nodejs.org/en/
3. Download the react version of the project (web-softwaretools-react) or clone the **web-softwaretools-react** repository from https://github.com/marceliwac/web-softwaretools-react/ using git.
4. Navigate to the project's directory and install the dependencies of the project by running "npm install". Then Launch the local version of the website using "npm run start".
5. Experiment with the website, try adding components and routes that the user can navigate to, modify the styles and expand on the application.

Extension if you have time and want to learn extra skills:

1. Create a boilerplate react application using the node create-react-app module.
   You can access the documentation for running the command on the module's npmjs page:
   - https://www.npmjs.com/package/create-react-app
2. Try running your application using the "npm run start" command from your projects root directory. At that point, your browser should automatically launch the website (commonly hosted at http://localhost:3000/
3. Think about how you could split your application into components and modify your React website's codebase accordingly.
4. Write a custom hook that connects to the API and fetch data from the pet-store API.

---

5. Write a function that loads the when the component is mounted (loaded). You can use the React's useEffect() hook for that.

Think about:

- What difference does it make now you can use React to speed up your development?
- How might these tools be helpful when building a larger scale site?
- What are the limitations?