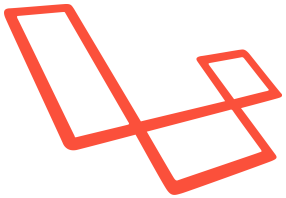
Prifysgol Abertawe
Swansea University



CSF304 – Web Application Development

Session 17: Web Sockets, Pusher & Many Other Tools

Dr. Liam O'Reilly

The Story So Far …

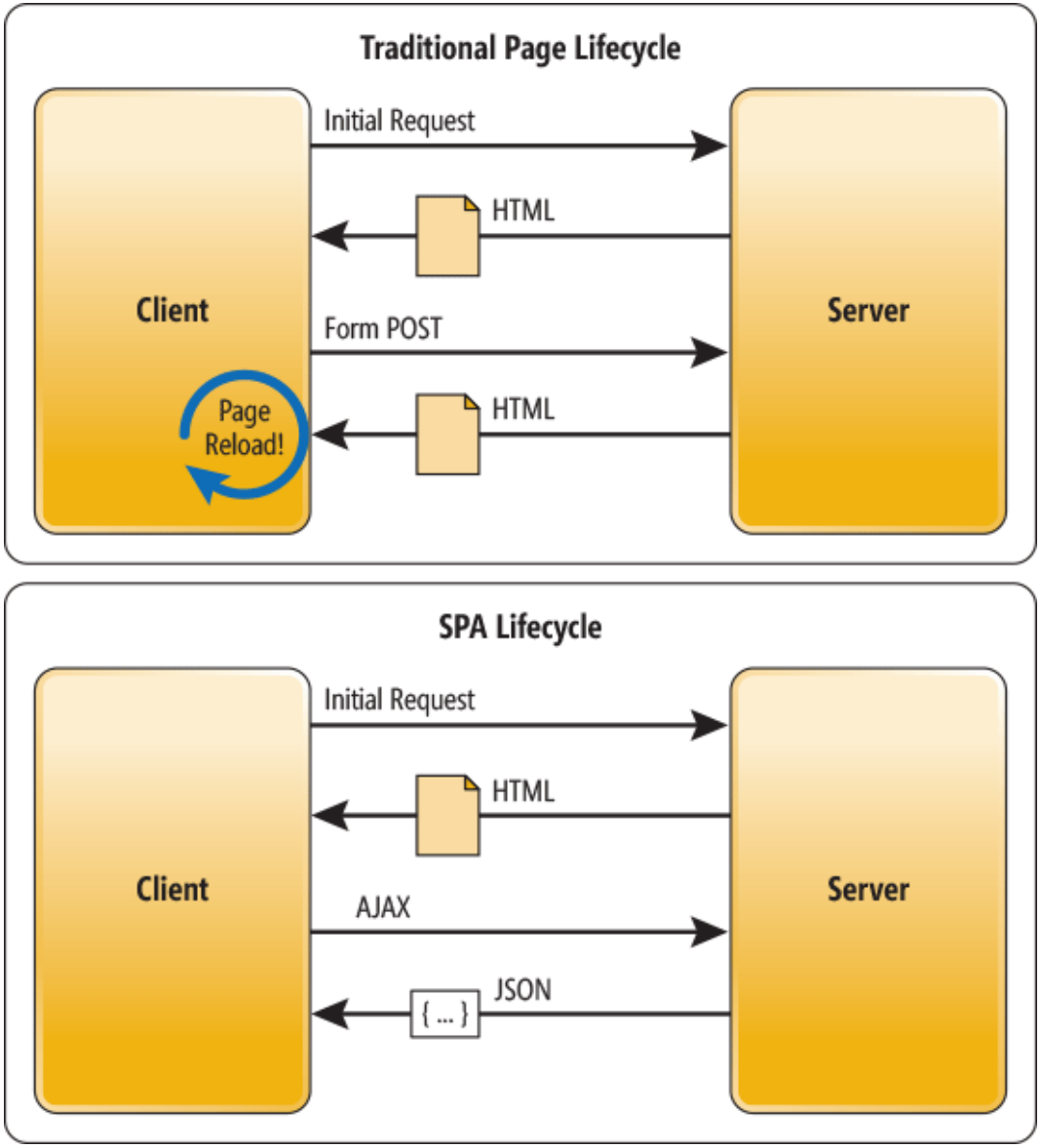
- So far we have taken a pragmatic approach to learning Laravel.
- We have leant by example mainly.
  - REST
  - CRUD
  - MVC
  - …
- We have focused on core Laravel rather than front-end.
- Reason: The technology stack and the amount of tools can be overwhelming.

Today

- Now we will briefly look at the tools such as Composer, NPM, Laravel Mix, …
- And also WebSockets and Pusher.

WebSockets and Pusher

Recap: Traditional Web Apps vs SPA



https://msdn.microsoft.com/en-us/magazine/dn463786.aspx

Recap: AJAX

- AJAX (Asynchronous JavaScript and XML):
  - AJAX is not a programming language.
  - It is a technique for accessing web servers from a web page.

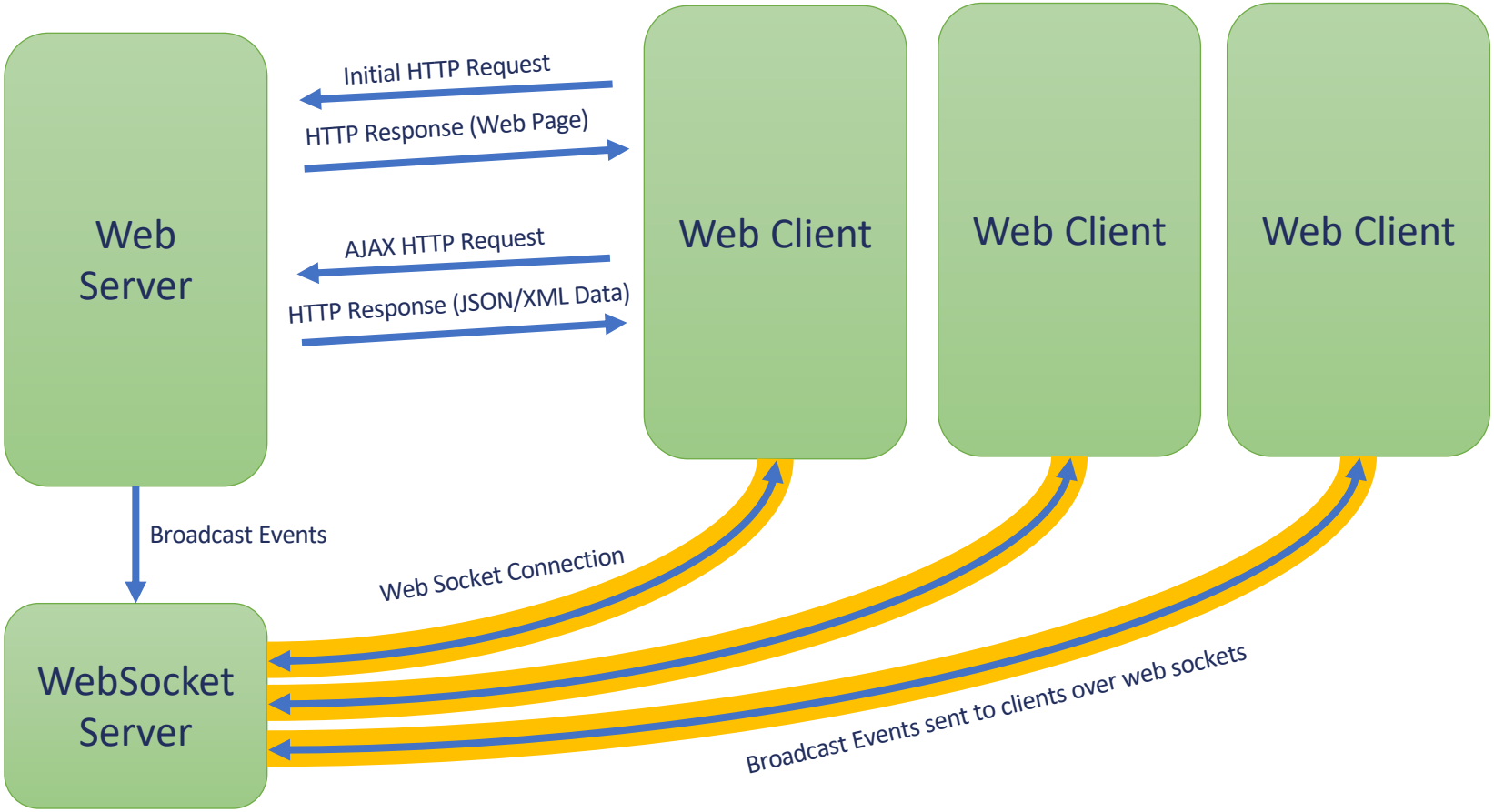
- We load a HTML page with JavaScript.
- The JavaScript then executes asynchronous HTTP requests in the background to communicate with servers.

- We use the data returned by AJAX to update the DOM of the webpage.

WebSockets

- So far all the HTTP requests originate from the clients.
- What if we want the client to wait/know about some events to happen on the server.
  - E.g., Webmail: check for new emails arriving.
  - E.g., Chat system: New messages or someone typing.
- Client could poll the server:
  - keep asking “anything updates?”
  - This is slow and resource intensive – especially as you increase the number of clients.
- Solution WebSockets .

WebSockets - Overview



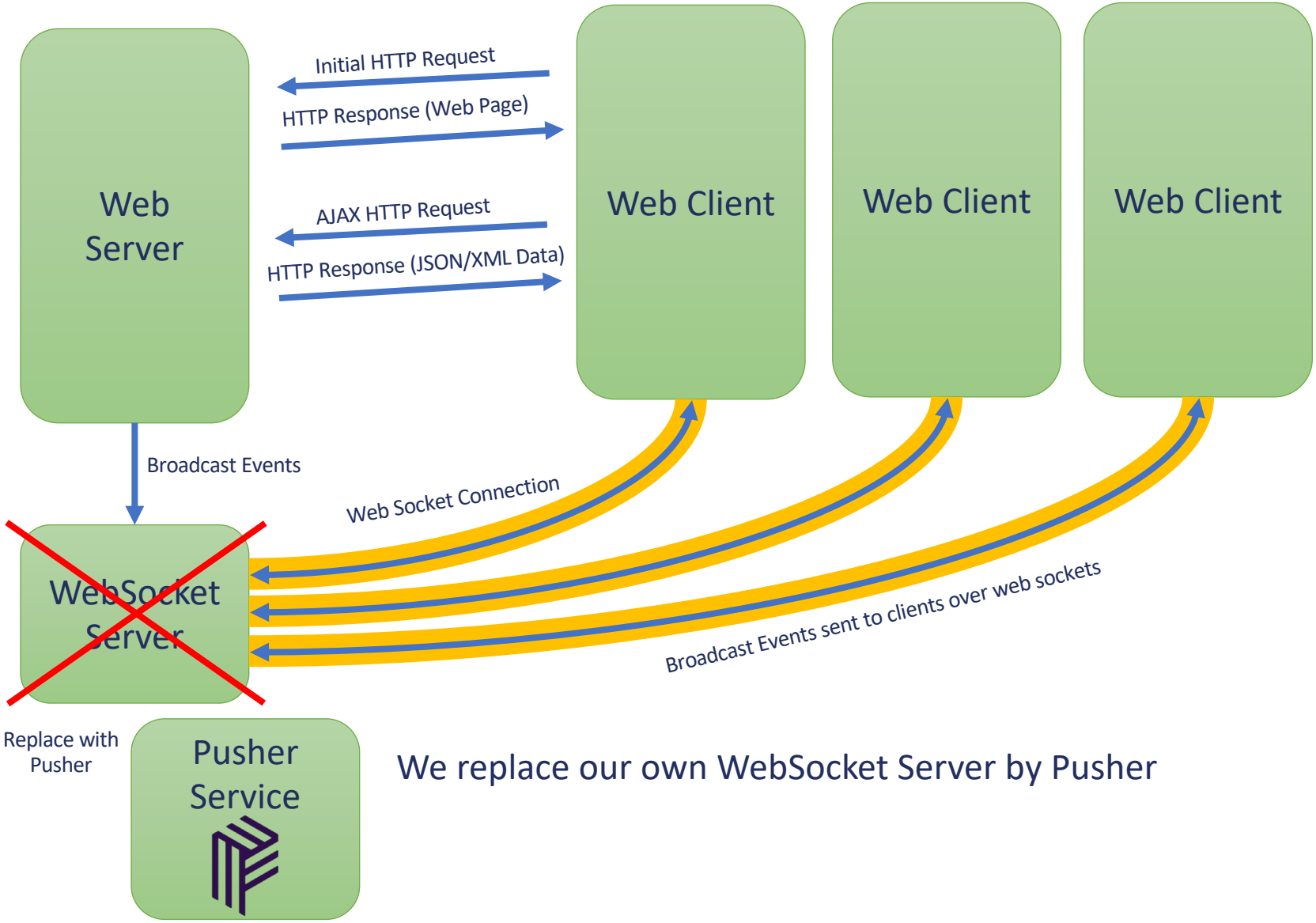
Note: the Web Server and WebSocket Server are software and can run on the same physical machine (especially for development).

WebSockets

- So now we can broadcast events to our clients.
  - The clients can respond to the events or chose to ignore them.
  - Of course, we have to provide even more JavaScript code to handle all this.

- It turns out that setting up that little “WebSocket Server” can be hard work.
  - Third Party Services to the rescue.

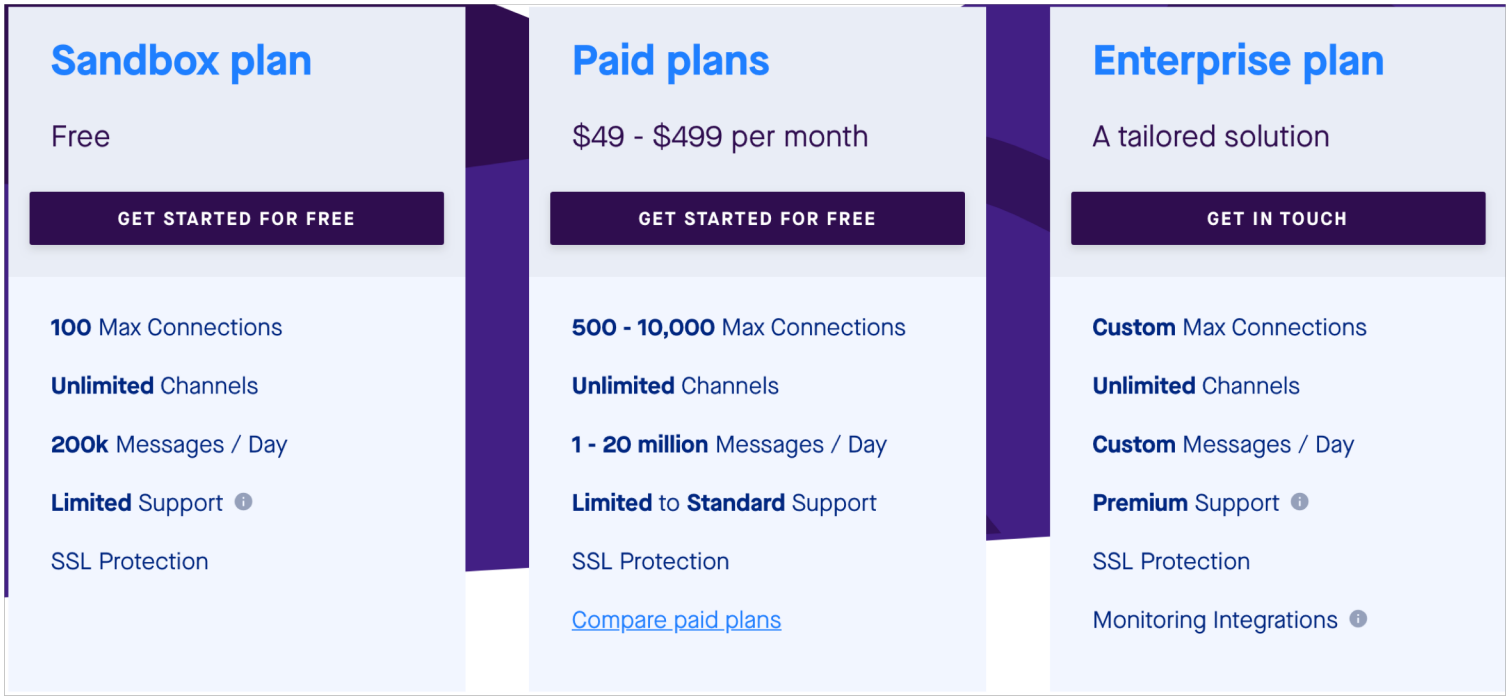
WebSockets - Pusher



We replace our own WebSocket Server by Pusher

Pusher

- Obviously Pusher is not free:



- The free plan look okay for small apps (??). 200,000 messages a day.
  - If you have 100 connections established then broadcasting a single event costs 101 messages.

The Code

- You have to write code to Broadcast Events.
  - Laravel has all this built in. You would need to learn about Broadcasting and Queues in Laravel.
- You have to write the front-end JavaScript to:
  - Establish the WebSocket Connection and process received events.
  - Laravel comes with a first-party JavaScript library called Laravel Echo that makes this all quite easy.

Composer

Composer

- Composer is the package manager for PHP.
- Use it to install and manage packages installed locally into your Laravel projects.
- Mainly used in Laravel to install server-side packages.

- Use the command “composer” to interact with it.
- There are two main configuration files for composer:
  - composer.json
  - composer.lock
  - Both should be placed in version control.

composer.json

```
{
    "name": "laravel/laravel",
    "description": "The Laravel Framework.",
    "keywords": ["framework", "laravel"],
    "license": "MIT",
    "type": "project",
    "require": {
        "php": "^7.1.3",
        "fideloper/proxy": "^4.0",
        "laravel/framework": "5.7.*",
        "laravel/tinker": "^1.0"
    },
    "require-dev": {
        "beyondcode/laravel-dump-server": "^1.0",
        "filp/whoops": "^2.0",
        "fzaninotto/faker": "^1.4",
        "mockery/mockery": "^1.0",
        "nunomaduro/collision": "^2.0",
        "phpunit/phpunit": "^7.0"
    },
    ...
```

What is always required.

What is required only for development.

Flexible (complex?) version syntax.

Any version of Laravel 5.7. E.g., 5.7.1 or 5.7.2 or …

- Composer.json specifies what PHP packages you want to install.
- Composer will solve the dependencies for you.
  - Find versions that match your spec and fit together.

composer.lock

- When using composer it will produce a file composer.lock.
- This specifies the actual version of the packages installed.
- You want to place this in your version control system so whole team is using the same versions.

```
...
"packages": [
    {
        "name": "dnoegel/php-xdg-base-dir",
        "version": "0.1",
        "source": {
            "type": "git",
            "url": "https://github.com/dnoegel/php-xdg-base-dir.git",
            "reference": "265b8593498b997dc2d31e75b89f053b5cc9621a"
        },
        "dist": {
            "type": "zip",
            "url": "https://api.github.com/repos/dnoegel/php-xdg-base-dir/
            zipball/265b8593498b997dc2d31e75b89f053b5cc9621a",
            "reference": "265b8593498b997dc2d31e75b89f053b5cc9621a",
            "shasum": ""
        },
        ...
```

npm

npm

- npm is the package manager for Node.js
  - Node.js is a JavaScript runtime system for running JavaScript without a web browser.
    - Can be used to build web servers.
- npm started off just for Node.js
  - But now it contains general tools to working with various web front-end develop tools.
- Use it with the “npm” command.

packages.json

- packages.json is where you define what packages npm should install.

Install axios and bootstrap so that we can provide axios to the client ourselves.

Can also specify small scripts to run. These are used to transpile code and produce website assets. See next section of slides.

Run them with, e.g.,

npm run dev

```
{
    "private": true,
    "scripts": {
        "dev": "npm run development",
        "development": "cross-env NODE_ENV=development node_mo
        "watch": "npm run development -- --watch",
        "watch-poll": "npm run watch -- --watch-poll",
        "hot": "cross-env NODE_ENV=development node_modules/we
        "prod": "npm run production",
        "production": "cross-env NODE_ENV=production node_modu
    },
    "devDependencies": {
        "axios": "^0.18",
        "bootstrap": "^4.0.0",
        "cross-env": "^5.1",
        "jquery": "^3.2",
        "laravel-mix": "^2.0",
        "lodash": "^4.17.5",
        "popper.js": "^1.12",
        "vue": "^2.5.17"
    }
}
```

SASS & LESS

Problems with CSS

- When writing CSS you will end up repeating code.
  - E.g., Main colour of website.
- CSS Does not have variables / constants.

- SASS and LESS allow you to write higher level CSS code that uses variables, constants, inheritance, etc.\
- This code must be compiled down to basic normal CSS before being severed to client web browsers.

Polyfilling, Transpiling, Minification and Laravel Mix

Evolution of JavaScript

Ver	Official Name	Description
1	ECMAScript 1 (1997)	First Edition.
2	ECMAScript 2 (1998)	Editorial changes only.
3	ECMAScript 3 (1999)	Added Regular Expressions. Added try/catch.
4	ECMAScript 4	Never released.
5	ECMAScript 5 (2009) Read More: JS ES5	Added "strict mode". Added JSON support. Added String.trim(). Added Array.isArray(). Added Array Iteration Methods.
5.1	ECMAScript 5.1 (2011)	Editorial changes.
6	ECMAScript 2015 Read More: JS ES6	Added let and const. Added default parameter values. Added Array.find(). Added Array.findIndex().
7	ECMAScript 2016	Added exponential operator (**). Added Array.prototype.includes.
8	ECMAScript 2017	Added string padding. Added new Object properties. Added Async functions. Added Shared Memory.
9	ECMAScript 2018	Added rest / spread properties. Added Asynchronous iteration. Added Promise.finally(). Additions to RegExp.

- JavaScript has evolved over time.
- New syntax has been added. Much nicer syntax too.
- It has become more powerful.

- Version of JavaScript are known as (equivalent to)
  - ECMAScript
  - ES5, ES6

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

Browser Support

- Different browsers support different versions of JavaScript.
  - For example. Chrome supported ES7 (ECMAScript 2016) in May 2018.
- We want to use the nice new syntax for writing JavaScript.
  - Really you do.
- But we want older browsers to run the code too.
  - We are not talking about ancient browsers here.

Polyfill

Pollyfill is the process of taking a new feature (e.g., set of methods) and producing code which emulates this new feature with code that only used the older standard.

Example:

- ES1 defines a method called isNaN(value) to determines whether a value is an **illegal number** (Not-a-Number).
- ES1 is supported by all browsers.
- ES6 defines a method called Number.isNaN(value) too determine whether a value is **NaN** (Not-a-Number).
- ES6 is not supported by all browsers (yet).
- We can produce code which emulates the new method in ES1

```
if (!Number.isNaN) {
        Number.isNaN = function isNaN(x) {
                return x !== x;
         };
}
```

https://stackoverflow.com/questions/31205640/what-is-the-difference-between-polyfill-and-transpiler

Transpilling

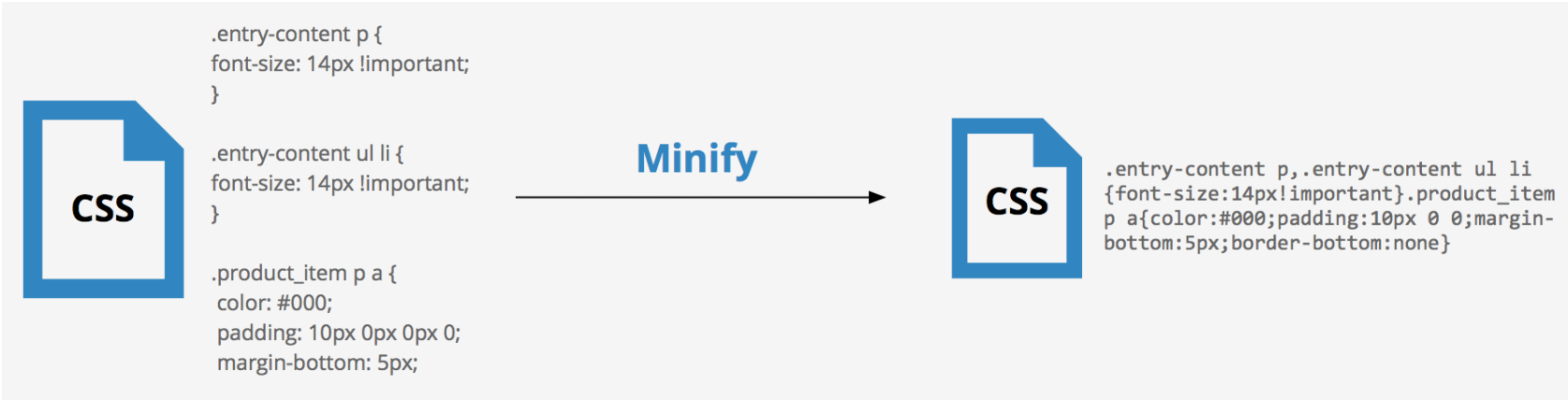
- When a new version of JavaScript introduces new syntax, then polyfilling is of no use.
  - The browser must run the new syntax which it won't understand.

- Transpilling is the process of converting the code to use an old version of the syntax but with the same semantical behavior.

- We could even be clear:
  - Provide the new versions of JavaScript to new browsers (might might be optimized for the new syntax).
  - Provide transpilled versions to older browsers.

Minification

- Why bother sending JavaScript/CSS files to the client which contains comments and white space.
  - Remember JavaScript code is interpreted and not compiled.
- Minification is the process of taking all comments and white space out.



https://www.keycdn.com/support/how-to-minify-css-js-and-html

- This helps deliver smaller files making our website faster.

How To Do All This?

- Managing all this pollyfilling, transpillation and minification can be tough.
  - We need a tool to do that for us.
- Gulp, WebPack are tools for doing this.
- LaravelMix is a wrapper around WebPack to make this easier.

Laravel Mix

- Laravel uses Laravel Mix out of the box.
- The file webpack.mix.js contains the code to produce “compiled” assets (JavaScript and CSS) from your written version.
- Default file:

```
mix.js('resources/js/app.js', 'public/js')
   .sass('resources/sass/app.scss', 'public/css');
```

  - This “compiles” your app.js file down to a public version and does the same with your CSS.
- This can apply polyfill, and transpilling and minification (based on your environment (production vs dev).

Back to npm

- Npm is used to kick off Laravel Mix.
- Just run

  npm run dev
- This will run your Laravel Mix code and produce your front end assets.
- The command
  npm run watch
  will (should?) constant watch for changes and “compile” on the fly.

```
{
    "private": true,
    "scripts": {
        "dev": "npm run development",
        "development": "cross-env NODE_ENV=development node_mo
        "watch": "npm run development -- --watch",
        "watch-poll": "npm run watch -- --watch-poll",
        "hot": "cross-env NODE_ENV=development node_modules/we
        "prod": "npm run production",
        "production": "cross-env NODE_ENV=production node_modu
    },
    "devDependencies": {
        "axios": "^0.18",
        "bootstrap": "^4.0.0",
        "cross-env": "^5.1",
        "jquery": "^3.2",
        "laravel-mix": "^2.0",
        "lodash": "^4.17.5",
        "popper.js": "^1.12",
        "vue": "^2.5.17"
    }
}
```

Okay That Was A Lot

- That was a bit too much.
- What is the point?
  - Just to let you know these tools exist and a rough idea of what they can do.

End?

- So we have reached the end.
- This course was quite full and bring together so many concepts covered through the course.
- We only touched the top of the Web Development Ice Berg.
- It is a never ending evolution of tools, techniques and technology.
- It requires constant learning and adapting to be a good web Developer.
- Hope you have had some fun at least ☺
  - And now know a bit more.