# CPD (Aug 29 2023)

1. Read through Prometheous Wiki (New Pages Section)
2. Learn TypeScript

## 1. Read through Prom Wiki (New Pages Section)

**Curriculum Overview Page (Useful for current DD Item):**

- Revamped version of the year group full statistics page.
- Presents high level curriculum stats.
- Allows for comparison of data over time for a year group.

  Accessability to page:

  - Via Pathfinder
  - Via the 'At A Glance' page

  Page parameters:

  - Year group (yg)
  - Optional (ay)

  Architechture:

  - Component based
  - Each component manages a number of subject stats tables.
  - Each component is responsible for loading its own data.

    Components:

    1. L3VA by Subject:
       - L3VA is a metric that measures progress made by schools / colleges, subjects and students from KS4 to Level 3 qualifications post-16.
       - Allows users to add pass rates to tables for a range of thresholds.
       - Allows users to select grade types relevent to their use case.
       - Add snapshotted data.
       - Save and load custom configs.
       - Only works with year groups 12 or 13 of an academic year from 2016 or later along with additional setting toggles.
       - Includes tool tips with information about the table results.

- Has a table explorer pop up.

2. Pass Rates by Subject:

- Revamped version of the pass rates by subject section in the detailed stats page.
- Allows users to add pass rates tables for a range of thresholds, select grade types and add snapshotted data.
- Always visible on the CO page.
- Has a counts or percentages toggle (percentages is the default on load)
- Includes tooltips and data explorer pop ups.

3. Average Points by Subject

- Revamped version of the average points by subject on the full stats page.
- This component allows the user to add average points tables for any point type (GCSE, school score, APS, new APS).
- Select grade types and snapshotted data.
- Always visible on page.
- Stats presented as decimal averages.
- Includes tooltips and data explorer pop up.

4. Common Commponent Features:

- Add Tables
- Select Grade Type
- Visualise Trends
- Select Snapshots
- Subject Sort
- Percentages and Counts
- Saved Selections
- Student Features
- Subject Filters

**Curriculum Analysis Page**

- The page where the year group ribbons and residual progress indicator will go.

  Accessability to page:

  - Via Pathfinder
  - Via the 'At A Glance' page

  Page parameters:

  - Year group (yg)
  - Optional (ay)

  Full Residual Progress Indicator (Full RPI):

  - This measure is the average of the distance between each student's grade and their overall average grade.
  - Better than average is scored positive and worse it negative.
  - A subject residual of 0.00 tells you that they did as well in this subject as they did in all subjects.
  - Includes RPI caching layer.

  Year Group Ribbons:

  - Interative ribbons to view curriculum subjects on year group pages.
  - Ribbons option to add aditional columns by grade type.
  - Ability to filter by subjects with the option to save the filter.
  - Sorting abilities (by department, by qualification)
  - Filter by focus group
  - Table view popout
  - Has a caching layer

**Attainment Over Time (AOT) Page**

- To see the changes in data throughout the academic years, and across 3 academic years, easily.
- Review changes in the data by reviewing key performance measures.
- Drill into data to view specific student data over time.
  Filter information by focus groups for performance measures and subject data.

  Workflow (year group page):

  - Select one or more KPI's to view
  - Select primary grade type
  - Select snapshots (read-only historical data from a given time)
  - Display a line graph and provide options for:

    1. Viewing the table
    2. Downloading as CSV
    3. Drilling into data via year group explorer.

### At A Glance Page

- A overview of the data for a year group through components

  Accessability to page:

  - Via Pathfinder
  - Via the 'Curriculum Analysis' page

  Page parameters:

  - Year group (yg)
  - Optional (ay)

  Components:

  - Attendance
  - Detentions
  - Behavior
  - RPI
  - KPI
  - Ribbons

**Key Statistics Page**

- An extension to the at a glance component with more statistics.

  Accessability to page:

  - Delagated privilage from the access table.
  - Via the Path finder
  - Via the 'At a glance' page

  Page parameters:

  - Year group (yg)

  Content displayed:

  - Statistics data
  - There is no snapshot data (AOT page does this)
  - KPI's drawn from the year groups page.
  - Mirrored stats from the year groups page.

  Interactivity:

  - Select grade types
  - Remember grade types
  - CSV Download
  - Print view

## 2. Learn TypeScript

**What is it?**

- A solution to JavaScript's messy dynamically typed code with improved strong type checking and compile-time error checks, embracing OOP.
- By definition, “TypeScript is JavaScript for application-scale development.”
- Compiles down to plain JavaScript.
- Developed by Microsoft.
- Has enhanced IDE support (Intelli Sense)

**The Basics:**

Installation - `npm install -g typescript`

Simple Hello World Program:

- TypeScript:

```
export {}
let message = 'Hello World';
console.log(message);
```

- Compiled into JavaScript:

```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
var message = 'Hello World';
console.log(message);
```

Variable Declarations:

- TypeScript variable declarations stop values such as a const being redeclared, this helps values you want to be kept the same have a contraint over it. The same goes for 'let' variables, they can be declared without a value and assigned it later.

```
let x;
const y = 20;
x = 30;
```

- Compiled to JavaScript:

```
var x;
var y = 20;
x = 30;
```

Variable Types:

- Boolean:
  - TypeScript:

```
let isBeginner: boolean = true;
```

  - Compiled JavaScript:

```
var isBeginner = true;
```

- Number:
  - TypeScript:

```
let total: number = 0;
```

  - Compiled JavaScript:

```
var total = 0;
```

- String:

- TypeScript:

```
let name: string = 'Ben';
```

- Compiled JavaScript:

```
var name = 'Ben';
```

- Concatenated String:
  - TypeScript:

```
let sentence: string = `My name is ${name}.`
```

  - Compiled JavaScript:

```
var sentence = "My name is ".concat(name, ".");
```

- Null & Undefined Types
  - These are subtypes of all other types.
  - Example:

    TS:

```
let n: null = null;
let u: undefined = undefined;
let isNew: boolean = null;
let myName: string = undefined;
```

    JS:

```
var n = null;
var u = undefined;
var isNew = null;
var myName = undefined;
```

Lists and Arrays:

- Single Type Array Example:

  TS:

```
let list1: number[] = [1,2,3,4];
let list2: Array<number> = [1,2,3,4];
```

  JS:

```
var list1 = [1, 2, 3, 4];
var list2 = [1, 2, 3, 4];
```

    - As you can see in this example, there is no difference in both of the TypeScript implementations.

- Mixed Type Array Example:

  TS:

```
let person1: [string, number] = ['Christ', 22];
```

  JS:

```
var person1 = ['Christ', 22];
```

    - TS enforces that the `person1` object needs to have a string followed by a number in the tuple, otherwise it wont compile.

Enum Type:

- Colours Example:

  TS:

```
enum Colour {Red,Green,Blue};
let c: Colour = Colour.Green;
console.log(c);

// outputs: 1

enum Colour2 {Red=5,Green,Blue};
let c2: Colour2 = Colour2.Green;
```

```
console.log(c2);

// outputs: 6
```

JS:

```
var Colour;
(function (Colour) {
    Colour[Colour["Red"] = 0] = "Red";
    Colour[Colour["Green"] = 1] = "Green";
    Colour[Colour["Blue"] = 2] = "Blue";
})(Colour || (Colour = {}));
;
var c = Colour.Green;
console.log(c);

// outputs: 1

var Colour2;
(function (Colour2) {
    Colour2[Colour2["Red"] = 5] = "Red";
    Colour2[Colour2["Green"] = 6] = "Green";
    Colour2[Colour2["Blue"] = 7] = "Blue";
})(Colour2 || (Colour2 = {}));
;
var c2 = Colour2.Green;
console.log(c2);

// outputs: 6
```

- Its clear that TS also provides abstraction over JS and provides types that can be defined with readability in mind.

The Any Type:

- Allows you to assign random types of values to a variable.
- Example:

  TS:

```
let randomValue: any = 10;
randomValue = true;
randomValue = 'Ben';
```

  JS:

```

```

```
var randomValue = 10;
randomValue = true;
randomValue = 'Ben';
```

Type Inference:

- TypeScript Example:

```
let a; // no value assigned on declaration.
a = 10;
a = true; // works fine.

let b = 20; // assign value on declaration.
b = true; // gives a TS error as type was inferred.
```

Multi Type:

- Example:

  TS:

```
let multiType: number | boolean;
multiType = 20;
multiType = true;
```

  JS:

```
var multiType;
multiType = 20;
multiType = true;
```