Overview of Standards and Assessments

Welcome to the Software Engineering Programme (SWE)! We are excited you are joining Multiverse to start your apprentice journey. Before you start, we want to give you a little more information on how you will be assessed and what the learning will look like. This programme is composed of the following:

- **5 week bootcamp** prepares you to enter the workplace ready for your first challenges. We cover the basics of the software development life cycle and begin to understand the role of a software developer. As well as learning the basics of programming in HTML and Javascript, you will take on coding challenges and gain practical experience of the entire software stack.
- **Module 1:** A 2 week bootcamp that builds and expands on the RESTful concept introduced in the 5-week bootcamp. Topics covered include: HTTP/HTTPS, RESTful API best practices, Building out a RESTful API, Securing an API using Basic Auth, Sessions, and Securing an API with OAuth
- **Module 2:** A two week bootcamp that will get you develop your proficiency with HTML, CSS and asynchronous JavaScript. You'll cover: WebSockets; web workers; XMLHTTPRequests; canvas elements; media elements; responsive design through media queries; and error handling in Javascript. By the end of the module you will be prepared to take the Microsoft Technology Associate Certification (MTA): Introduction to Programming Using JavaScript
- **Module 3:** A two week bootcamp to fully understand and navigate the software development lifecycle. Through a mixture of independent reading and a team project, you will learn about both agile and waterfall development methodologies and experience 2 x 1-week sprints. By the end of the module you will be prepared to take the BCS Level 4 Diploma in Software Development Methodologies
- **Module 4:** A two week bootcamp to get you to EPA-readiness for the 5-day synoptic project that forms part of the Apprenticeship EPA. Through completing two individual projects, you will be prepared for the demands of the EPA synoptic project.

How will you be assessed?

During your sessions with your coach you will work on a variety of assignments. Below is a list of some of the ways you may be assessed in the programme.

- Projects to be completed after the day of learning
- Short answer questions around understanding of software engineering concepts
- Scenario based questions around role as a developer
- Questions evaluating your role in the business with your learning
- Individual projects you will complete throughout a module
- Group projects you will complete throughout a module

Standards of Software Engineering

All the lessons you will go through during the Software Engineering Programme are directly aligned with the level 4 Software Developer programme as designed by the Institute for Apprenticeships & Technical Education. Use this page as a reference to reflect on what you are learning and areas to develop while on the job. The standards are:

Technical Competencies

- Logic: writes good quality code (logic) with sound syntax in at least one language
- User interface: can develop effective user interfaces for at least one channel
- Data: can effectively link code to the database/data sets
- Test: can test code and analyse results to correct errors found using either V-model manual testing and/or using unit testing
- Problem solving: can apply structured techniques to problem solving, can debug code and can understand the structure of programmes in order to identify and resolve issues
- Design: can create simple data models and software designs to effectively communicate understanding of the program, following best practices and standards
- Analysis: can understand and create basic analysis artefacts, such as user cases and/or user stories
- Deployment : can understand and utilise skills to build, manage and deploy code into enterprise environments
- Development lifecycle: can operate at all stages of the software development lifecycle, with increasing breadth and depth over time with initial focus on build and test.
- Can apply good practice approaches according to the relevant paradigm (for example object oriented, event driven or procedural)
- Can interpret and follow:
  - software designs and functional/technical specifications
  - company defined ‘coding standards’ or industry good practice for coding
  - testing frameworks and methodologies
  - company, team or client approaches to continuous integration, version and source control

- Can respond to the business environment and business issues related to software development
- Can operate effectively in their own business's, their customers' and the industry's environments
- Can apply the maths required to be a software developer (e.g. algorithms, logic and data structures)

Technical Knowledge and Understanding

- Understands and operates at all stages of the software development lifecycle
- Understands the similarities and differences (taking into account positives and negatives of both approaches) between agile and waterfall software development methodologies
- Understands how teams work effectively to produce software and contributes appropriately
- Understands and applies software design approaches and patterns and can interpret and implement a given design, compliant with security and maintainability requirements
- Understands and responds to the business environment and business issues related to software development
- Understands and applies the maths required to be a software developer (eg algorithms, logic and data structures)

Underpinning Skills, Attitudes and Behaviours

- Logical and creative thinking skills
- Analytical and problem solving skills
- Ability to work independently and to take responsibility
- Can use own initiative
- A thorough and organised approach
- Ability to work with a range of internal and external people
- Ability to communicate effectively in a variety of situations
- Maintain productive, professional and secure working environment

Multiverse M

Foundations

Unit-1

1.1.1 Setup (could also be done prior to course)

Before we start

You must have familiarity with basic JavaScript concepts, specifically:

- JavaScript Syntax (https://www.w3schools.com/js/js_syntax.asp)
- JavaScript Data Types (https://www.w3schools.com/js/js_datatypes.asp)
- JavaScript Variables (https://www.w3schools.com/js/js_variables.asp)
- JavaScript Arrays (https://www.w3schools.com/js/js_arrays.asp)
- JavaScript Array Methods (https://www.w3schools.com/js/js_array_methods.asp)
- JavaScript Arrow Functions (https://www.w3schools.com/js/js_arrow_function.asp)
- An overview of Node (https://www.codecademy.com/articles/what-is-node) - also watch the first 5.30 minutes of An Introduction to Node in Visual Studio Code (https://www.youtube.com/watch?v=ElQgVdoYb0M)
- There is a great article on the Community Hub (https://community.multiverse.io/topics/16826/feed) which compares statically typed languages (like JavaScript) and dynamically typed languages (like Java) - well worth a read!

You must also be confident using Git to clone, add, commit and push to your Git repository.

Overview of the Unit

Day one enables the cohort to get their computer set up so that they can write JavaScript using Node.js. We introduce the concepts of Arrays, Objects and functions in Javascript.

Install

(Note that if students are unable to install software to their computer, it is possible to use the online editor Repl.it (https://repl.it/) to create and run JavaScript and Node.js instead).

Please install the following:

- VSCode (https://code.visualstudio.com/) - this is a code editor
- Node (https://nodejs.org/en/) - Node.js is an open-source, cross-platform, back-end, JavaScript runtime environment that executes JavaScript code outside a web browser. Check if this is already installed by typing `node` on a Command Promt. If not, then select the LTS version.
- Enable `Auto Save` in VSCode via `File -> Auto Save`
- Install JsDoc (https://jsdoc.app/) (an API documentation generator for JavaScript) by typing the following command in your command prompt after you have installed Node: `npm install -g jsdoc`
- Ensure you have the Chrome browser installed

Validate your setup as follows:

1. Create a new folder called `coursework` under the `Documents` folder in your home directory
2. Open VSCode and select `File->Open Folder` - select your `coursework` folder
3. In the `coursework` folder, create a new file (`File->New File`) called `script.js` with the following content:

```
console.log('hello from JavaScript');
```

4. Check your code has been saved by looking at the timestamp for the file in `File Explorer`
5. Using a Command Prompt, type `node script.js`. You should see the console log output. Here we are running the JavaScript server-side, i.e. within Node.

Unit-2

2.1.1 The second coming

And on the next page there was more, another section indeed. With more additions and more markdown.