Updates Regarding Quiz:

Your quiz will mainly cover what we have discussed in the last class.

Total Questions: 26

Theoretical part:

https://wania-kazmi.notion.site/Q1-Batch-53-1d8b23d02262441290d46a568deabaf2?pvs=25

Practical part includes the following steps:

Steps 00, 00a, 00d, 01, 02

https://github.com/panaverse/learn-typescript/tree/master

We will see who scores the highest marks in the class.

Best of luck!

Announcement

Good News for all the students who were worried about their assignments scores. Here is another opportunity for you all:

I and my team will maintain the score board of every individual student, and here is the chance who cannot score well in the assignment. We are going to have a presentation challange and those who gave presentation will be rewarded with additional 30 marks depending on your presentation.

Your presentation should cover the following key points ( make sure all these points should interlink with one another in presentation):

1. How would you define entrepreneurship, and in your opinion, do you see it as more significant than pursuing a traditional job?

2. What are the key advantages of entrepreneurship over traditional job-seeking, and how can these benefits contribute to personal and professional growth?

3. How does the mindset of an entrepreneur differ from that of someone seeking a job, and what qualities are crucial for success in entrepreneurship?

4. Can you provide examples of successful entrepreneurs who have made a significant impact, and what lessons can be learned from their journeys compared to those who pursued traditional employment?

5. In what ways does entrepreneurship offer greater flexibility and autonomy compared to a traditional job, and how can these aspects positively influence one's career trajectory?

6. How does the potential for financial growth and independence in entrepreneurship compare to the relatively stable income associated with traditional employment, and what factors should be considered when making this decision?

7. What resources and support systems are available for aspiring entrepreneurs to cultivate the skills and mindset needed for success, and how can individuals proactively seek out these opportunities?

Note: This challange is not mendatory but an additional activity.

Medium of presentation could be Engling or Urdu

Your presentation should be on the point and donot extend more than 7mins

What matters in your presentation:

1. Presentation Quality

2. Confidence

3. Verbal Communication Style

4. Your body language

5. Interaction and engagement with audience

6. Your audience should be engage while you deliver your presentation

7. Time management

* End of fiscal celebration on July 15th
* Employee day of learning on August 14th
* Employee Yoga on September 3rd
* Seminar series begins September 10th

**Class 1**

* **What is Javascript - it’s history**
* **How Typescript came up**
* **How Typescript is superset of Javascript**
* **Difference between Javascript and Typescript**
* **What is tsconfig file and what is package.json file and what they used for ?**
* **What is module vs commonjs ?**
* **What are Transitive Dependencies?**
* **what are primitive data types**
* **What is strong typing ?**
* **Difference between const and let and var**

💡 Note: We will follow from step 0 to step 3c and 4

💡 Note: Announcement about 2 quizes

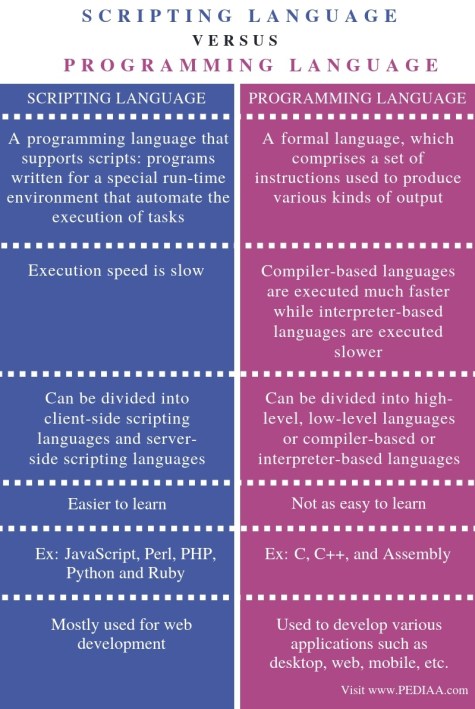
<https://docs.google.com/presentation/d/1-7Kb3laJjJ68mOTF9v0fHImk5vTol0CeE43Sg8hoUXQ/edit#slide=id.g15a453e6c68_0_408>

**History of Javascript:**

* The early to mid - 1990s was an important time for the internet
  + The internet was evolving at that time and big companies were joining the internet like Netscape navigator, Microsoft etc
* In september 1995, a Netscape programmer named Brandan Eich developed a new scripting language in just 10 days.
  + It was build to improve some features and there was a competition between Netscape and Microsoft
  + Firstly the name of this language was Mocha then changed to LiveScript and later changed to JavaScript
* JavaScript code is executed by JavaScript Engine, every browser has this engine to run the javascript such as chrome has V8 engine and even if your code is in nodejs even then Javascript engine will run it.

**What are scripting languages?**

A scripting language is a type of programming language that interprets and executes commands one at a time. These languages are designed for tasks like automation, customization, and manipulation within existing systems



In computing, a compiler is a computer program that translates computer code written in one programming language into another language.

An interpreter is a program that directly executes the instructions in a high-level language, without converting it into machine code. In programming, we can execute a program in two ways. Firstly, through compilation and secondly, through an interpreter. The common way is to use a compiler.

Programming languages like Python, Ruby, PHP, etc. use an interpreter. These interpreted languages are also called **scripting languages**.

Compile time is the period when the programming code (such as C#, Java, C, Python) is converted to the machine code (i.e. binary code). Runtime is the period of time when a program is running and generally occurs after compile time.

<https://www.baeldung.com/cs/runtime-vs-compile-time#:~:text=Compile%20time%20is%20the%20period,generally%20occurs%20after%20compile%20time>.

**Q. Research about why it is named as JavaScript?**

Netscape and Eich designed JavaScript as a scripting language for use with the company's flagship web browser, Netscape Navigator. Initially known as LiveScript, Netscape changed the name to JavaScript so they could position it as a companion for the Java language, a product of their partner, Sun Microsystems.

A javascript engine is a computer program that execute Javascript code.

**V8 became the engine that now powers an incredible amount of server-side code written in JavaScript**.

**ECMAScript:**

* ECMAScript, also known as JavaScript, is a scripting language standardized by Ecma International.
* It sets rules, details, and guidelines for scripting languages to achieve ECMAScript compliance

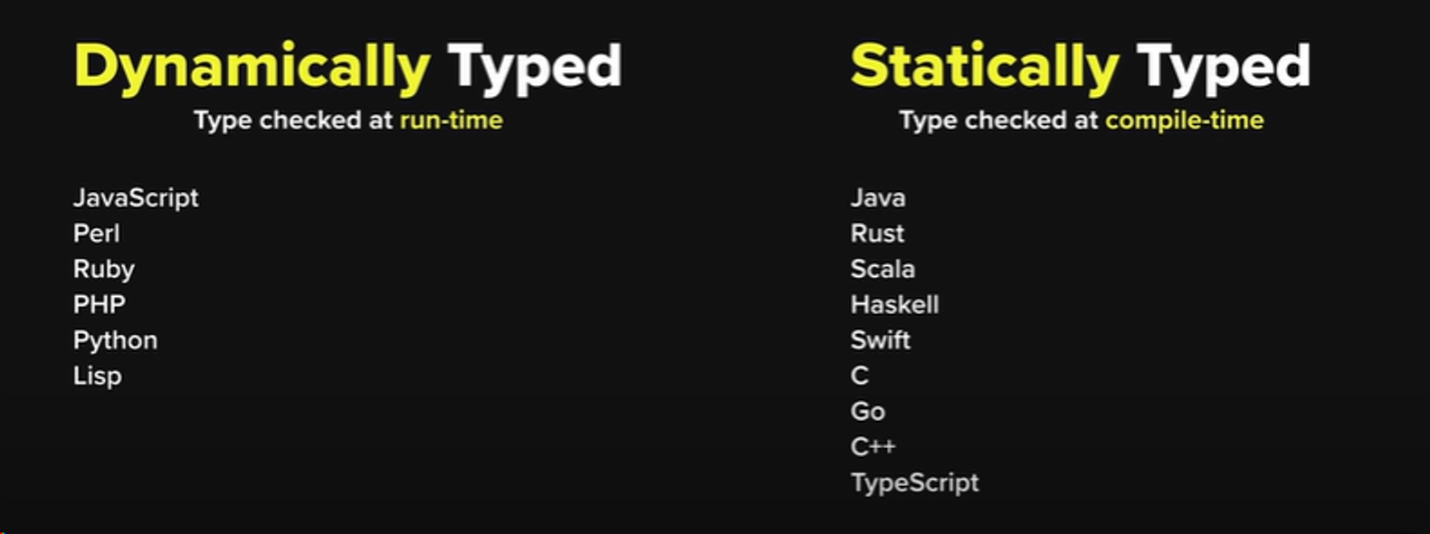
JavaScript is the coffee-flavored language with which I love to program. ECMAScript is the specification it’s based on. By reading the ECMAScript specification, you learn how to create a scripting language. By reading the JavaScript documentation, you learn how to use a scripting language.

A JavaScript engine

A program or interpreter that understands and executes JavaScript code. JavaScript engines are commonly found in web browsers, including V8 in Chrome, SpiderMonkey in Firefox, and Chakra in Edge. Each engine is like a language module for its application, allowing it to support a certain subset of the JavaScript language.

## Typescript:

There are 2 Types of languages Dynamically typed and statically typed languages



* **How Typescript came up**

TypeScript originated from the shortcomings of JavaScript for the development of large-scale applications both at Microsoft and among their external customers. Challenges dealing with complex JavaScript code led to demand for custom tooling to ease developing of components in the language.

* **How Typescript is superset of Javascript**

TypeScript is a programming language that's a superset of JavaScript, which means it understands all of JavaScript's syntax and capabilities, while adding additional features. TypeScript's primary value add over JavaScript is static typing. This means that type-checking happens at the time of code compilation

* **Difference between Javascript and Typescript**

When JavaScript was developed, the JavaScript development team introduced JavaScript as a client-side programming language. But as people were using JavaScript, developers also realized that JavaScript could be used as a server-side programming language. However, as JavaScript was growing, JavaScript code became complex and heavy. Because of this, JavaScript wasn’t even able to fulfill the requirement of an Object-Oriented Programming language. This prevented JavaScript from succeeding at the enterprise level as a server-side technology. So **TypeScript** was created by the development team to bridge this gap.

**Features of TypeScript:**

* **TypeScript Code is converted into Plain JavaScript Code:** TypeScript code can’t be natively interpreted by browsers. So if the code was written in TypeScript, it gets compiled and converted into JavaScript. This process is known as **Trans-piled**. With the help of JavaScript code, browsers are able to read the code and display it.
* **JavaScript is TypeScript**: Whatever code is written in JavaScript can be converted to TypeScript by changing the extension from **.js** to **.ts**.
* **Use TypeScript anywhere:** TypeScript can be compiled to run on any browser, device, or operating system. TypeScript is not specific to any single environment.
* **TypeScript supports JS libraries:** With TypeScript, developers can use already existing JavaScript code, incorporate popular JavaScript libraries, or call the TS Code from native JavaScript code.

**Difference between TypeScript and JavaScript:**

* TypeScript is known as an Object-oriented programming language whereas JavaScript is a prototype-based language.
* TypeScript has a feature known as Static typing but JavaScript does not support this feature.
* TypeScript supports Interfaces but JavaScript does not.

**Advantages of using TypeScript over JavaScript**

* TypeScript always points out the compilation errors at the time of development (pre-compilation). Because of this getting runtime errors is less likely, whereas JavaScript is an interpreted language.
* TypeScript supports static/strong typing. This means that type correctness can be checked at compile time. This feature is not available in JavaScript.
* TypeScript is nothing but JavaScript and some additional features i.e. ES6 features. It may not be supported in your target browser but the TypeScript compiler can compile the **.ts** files into ES3, ES4, and ES5 also.

| **Feature** | **TypeScript** | **JavaScript** |
| --- | --- | --- |
| Typing | Provides static typing | Dynamically typed |
| Tooling | Comes with IDEs and code editors | Limited built-in tooling |
| Syntax | Similar to JavaScript, with additional features | Standard JavaScript syntax |
| Compatibility | Backward compatible with JavaScript | Cannot run TypeScript in JavaScript files |
| Debugging | Stronger typing can help identify errors | May require more debugging and testing |
| Learning curve | Can take time to learn additional features | Standard JavaScript syntax is familiar |

**Installation Process:**

**What is json?**

1. **Json stands for Java script object notation,** JSON is a text-based data format that is used to store and transfer data.

But wait, Is JSON is similar to javaScript objects?

The Answer is No.

1. JavaScript objects can contain functions but JSON not.
2. JavaScript objects can only be used in JavaScript but JSON can be created and used by other programming languages.

**JSON Data**

1. JSON data consists of key/value pairs similar to JavaScript object properties.
2. The key and values are written in double quotes separated by a :.
3. Example :

// JSON data

"name": "Vipin"

JSON data requires double quotes for the key.

## JSON Object

1. The JSON object is written inside curly braces { }.
2. JSON objects can contain multiple key/value pairs.
3. Example :
4. // JSON object

{ "name": "Vipin", "age": 21 }

## JSON Array

1. JSON array is written inside square brackets [ ].

## Accessing JSON Data

1. We can access JSON data using the dot notation.
2. We can also use square bracket syntax [] to access JSON data.

**Use of JSON**

1. JSON is the most commonly used format for transmitting data (data interchange) from a server to a client and vice-versa.
2. JSON data are very easy to parse and use.
3. JSON is language independent(We can create and use JSON in other programming languages too).

**Examples of JSON**

1. [package.json](https://docs.npmjs.com/cli/v9/configuring-npm/package-json)
2. [tsconfig.json](https://www.typescriptlang.org/docs/handbook/tsconfig-json.html)

**What is tsconfig.json file?**

**Create tsconfig.json file** - tsc —init

The **tsconfig.json** file in TypeScript is a configuration file that specifies the settings and options for the TypeScript compiler (**tsc**). It stands for "TypeScript Configuration," and its primary purpose is to help manage the compilation process for TypeScript projects.

Here are some key aspects of the **tsconfig.json** file and its purpose:

1. **Compiler Options:** The **tsconfig.json** file allows you to specify various compiler options, such as the target ECMAScript version (**target**), module system (**module**), source map generation (**sourceMap**), and more. These options define how the TypeScript code should be transpiled into JavaScript.
2. **File Inclusion and Exclusion:** You can specify which files should be included or excluded from the compilation process using the **include** and **exclude** options. This helps define the scope of your TypeScript project.
3. **Module Resolution:** TypeScript supports different module resolution strategies (e.g., CommonJS, AMD, System). The **moduleResolution** option in **tsconfig.json** allows you to specify how module imports should be resolved.
4. **Type Declaration Files:** TypeScript allows the use of type declaration files (**.d.ts**) to provide type information for JavaScript libraries. The **types** and **typeRoots** options in **tsconfig.json** enable you to manage type declarations.
5. **Output Configuration:** You can configure the output directory for transpiled JavaScript files using the **outDir** option. This helps organize the output structure of your project.
6. **Strict Checks:** TypeScript offers strict type-checking options to catch more potential errors. The **strict** option enables a set of strict type-checking options collectively.
7. **Project References:** For large projects that consist of multiple sub-projects, the **tsconfig.json** file supports the **references** option, allowing you to specify project dependencies and establish a build order.

**What we are going to change in our tsconfig file?**

"target": "ES2022”

"module": "NodeNext",

"moduleResolution": "NodeNext",

**package.json file:**

**Create package.json file** - npm init -y

**Install types of node.js** - npm i @types/node -D

**What we are going to change in our package.json file?**

"type": "module",

**What is difference between dependency and devdependencies? H/w**

A dependency is a library that a project needs to function effectively. DevDependencies are the packages a developer needs during development. A peer dependency specifies that our package is compatible with a particular version of an npm package

"dependencies" : Packages required by your application in production. "devDependencies" : Packages that are only needed for local development and testing.

The dependencies object specifies the packages that you need to run your code. For example React, Vue, Firebase, etc. When you run npm install some-package , npm installs the package and adds it to the dependencies object in the package

dependencies are required to run , devDependencies only to develop , e.g.: unit tests, Coffeescript to Javascript transpilation, minification, ... React is a dependency because it is included in the final build. In case of a React App, all your JSX is converted to a syntax similar to React.

**What is package.json file?**

The **package.json** file is a metadata file used in Node.js projects to define various project-related configurations, dependencies, scripts, and other information. It serves as a central configuration file and is a standard part of Node.js projects.

1. **Project Metadata:** The **package.json** file includes metadata about the project, such as the project name, version, description, author, and license. This information helps identify and document the project.
2. **Dependencies and DevDependencies:** The **package.json** file lists the project's dependencies and devDependencies. Dependencies are the libraries and tools that the project depends on to run, while devDependencies are dependencies used during development (e.g., testing frameworks, build tools). TypeScript itself is often listed as a devDependency.
3. **Scripts:** The **package.json** file allows you to define custom scripts that can be executed using npm or yarn. These scripts can be used for various tasks such as building, testing, and running the project. In TypeScript projects, scripts are commonly used for compiling TypeScript code to JavaScript.
4. **Configuration for TypeScript:** The **package.json** file often includes TypeScript-specific configurations. This can include a **"type"** field set to **"module"** if you are using ECMAScript Modules in TypeScript.
5. **Version Management:** The **package.json** file helps manage the version of the project and its dependencies. It allows developers to use tools like npm or yarn to install the correct versions of dependencies based on the specified version ranges.

## ****Fun Fact:****

## ****Q. What is the shortest program of javascript?****

Ans: An empty file.

Because even then, JS engine does a lot of things. As always, even in this case, it creates the GEC which has memory space and the execution context. JS engine creates something known as 'window'.

## Q What are dependency / dev dependencies and their difference:

Our project is dependent on some packages (called as dependencies) and these packages are dependent on some other dependencies and those dependencies are further dependent on some other dependencies and so on these are called as **transitive dependencies**

**Q. Difference between package.json and package.lock.json file?**

package.lock does for you is:

1. It locks the version for you and maintain the exact version of your project.
2. It maintain the hash of it as well. This hash ensures that the exact version is running on my system is exactly the same on production or not. It is maintained by integrity.

**Q. What is node module?**

It is like a database for your npm and when ever you install any package it get installed into your node\_module.

Our application has dependencies on some package (turbo, parcel etc). Now parcel also has dependencies on something else and those dependencies can also have dependencies on something else.

Project (dependent on) ———> parcel (dependent on) ——————→ minifier and so on…

This is known as **transitive dependencies.**

* **What is module vs commonjs ?**
* **What are Transitive Dependencies?**
* **what are primitive data types**
* **What is strong typing ?**
* **Difference between const and let and var**

