**Names:** Adam Kraus, Chami Senerath, Hunter Hufstutler

**Language:** Typescript

**Coding environment**: Node.JS w/ TypeScript 4.5.2

### **Distinct pieces of functionality (that the team personally wrote for the project part):**

1. Database Connection
   1. /src/data-source.ts
   2. Everything in /src/entity
2. Management
   1. /src/scripts/ts/add.ts
   2. /src/html/add.html
   3. /src/scripts/ts/list.ts
   4. /src/html/list.html
3. Searching
   1. /src/scripts/ts/watchlist.ts
   2. /src/html/watclist.ts

### **Section 1: Comparison to C++, Java, Python, and/or Scala**

1. **What is the language’s philosophy?**

JavaScript with type safety

1. **Compare and contrast your language in terms of the location it is used.**

JavaScript/TypeScript/PHP/C#

1. **Compare and contrast your language in terms of where it excels and where it fails**
   * **Excels**

Static typing, spotting early bugs, predictable, OOP

* + **Fails**

Bloated code, added transpiling step

1. **Compare and contrast your language in terms of portability, simplicity, orthogonality, AND reliability.**
   * **Portability**

TypeScript is very portable since it transpiles to JavaScript. If it can run JavaScript, it will work on that system.

* + **Simplicity**

TypeScript keeps much of the same structure as JavaScript, and adds additional type safety. That adds a little more to learn, and can help further down in development.

* + **Orthogonality**

JavaScript is already orthogonal with classes and OOP, but TypeScript adding static typing makes it a lot easier and safer to do.

* + **Reliability**

Since TypeScript adds static typing for variables, it is very unlikely that a bad value makes it into a variable to cause an issue further into runtime.

### **Section 2: Syntax, OOP**

1. **A) Write an example of one type of assignment expression in the language.   
   B) Then write the generic format of an expression. Eg. Pre-fix, post-fix, curly brackets, indentation requirements, etc. The goal here is the general appearance of a line or block of code.**

let str: string = “Hello World!”;

let val = 4; // inferred typing to number

let arr: any[] = [2, “test”];

let name: type = value;

1. **A) How does the language support extension, etc. (single inheritance, interfaces, root object, class OOP, prototype OOP, other OOP, file importing, file extension, plugins, piping, module linking, etc.)?**TypeScript has class OOP, file importing

**B) Give an example.**

Class Animal {

Name: string;

Constructor(newname: string) {

This.name = newname;

}

Speak(speech: string) {

Console.log(`Animal spoke ${speech})`);

}

}

Class Frog {

Constructor(name: string) {

Super(name);

}

Speak(speech: string) {

Console.log(`Frog spoke ${speech})`);

}

}

1. **A) How does the language handle module/namespace/packages/etc.?**
2. *import* { Rating } *from* "./entity/Rating";
3. *import* { Role } *from* "./entity/Role";
4. *import* { WatchlistFilter } *from* "./entity/WatchlistFilter";
5. *import* \* *as* bodyparser *from* "body-parser";
6. *import* \* *as* express *from* "express";
7. *import* \* *as* path *from* "path";

You can import things from other files, as well as other packages that you have installed.

**B) What is the scope operator(s)? Alternatively, explain how to pick which variable if two code courses contain the same name?**

The scope operator for TypeScript is .

**A) Does the language allow function overloading (name repetition), function redefinition, and/or function overriding?**TypeScript does provide function overloading, unlike JavaScript, but not with different number of parameters and types with the same name. display(a: string, b: string) and display(a: number) would not work. Overriding is allowed, since TypeScript has OOP functionality with classes.

**B) Give example syntax if it does.**

function add(a: string, b: string): string;

function add(a: number, b: number): number;

function add(a: any, b: any): any {

return a + b;

}

add(“Hello ”, “Steve”) would be “Hello Steve”

add(10, 20) would be 30

### **Section 3: Binding, type system, and data type range**

1. **Is the language static or dynamically typed? Give example syntax in code.**

Static typing

1. **Is the language static or dynamically scoped? Give an example in code.**

Static scoping

1. **Describe the type system (equivalence, compatibility, etc.).**

TypeScript has structural subtyping for compatibility, so the following would work without explicitly stating that Dog implements Pet.

Interface Pet{

}

Class Dog{

}

Let pet: Pet;

Pet = new Dog();

In general, TypeScript says that *x* is compatible with *y* if *y* has at least the same members of *x*.

1. **What are the built-in data types and their ranges? (list 4-10, or send me a note if you believe that there are less than 4)**

Boolean

String

Number

Array

Tuple

Enum

Unknown

Any

Void

### **Section 4: Control flow, functions, specialties**

1. **What are the selection and repetition structures of the language, and what are their syntax?**

If(statement1) {

} else if(statement2) {

} else {

}

Switch(value) {

case “A”:

break;

default:

break;

}

For(let i = 0; i < 10; i++) {

}

Array.forEach(value => {

});

1. **Are functions pass-by-value, pass-by reference, etc.? Give example syntax in code.**

Primitives are pass by value, and objects and arrays are pass by reference.

Add(a: number) {

a += 5 // wouldn’t change outside value

}

Add(a: number[]) {

a[0] = 5; // would change outside value

}

1. **Describe at least two of the language specialties.**

Specialty one:

Optional static typing is the main point of TypeScript and improvement over JavaScript.  
Specialty two:  
 Generic type functions allow for flexibility and repeatability

function generic<Type>(arg: Type): Type {

return arg;

}