Review Session #0

With your host: Brooks Townsend

About Me

- Junior
- CS BS Major, Math minor
- Hometown: Huntersville / Cornelius NC
- Hobbies: Weightlifting, hiking / camping, video games, coding for ~fun~

Objectives

- Understand basic variables
- Be able to identify what values correspond to what types
- Understand the concept of control flow

Variables

- Variables are “containers” that hold information
- We can give variables names in order to add context to our code
- The **type** of information that we are dealing with dictates the **type** of variable that we will declare.

Declaration

- “Creating” our variable
- `let <name>: <type>;`
- `let x: number;`
- `let name: string;`
- `let isWednesday: boolean;`

Initialization

- Giving our variable an “initial” value
- `x = 10;`
- `name = “Kris Jordan”;`
- `isWednesday = true;`

Declaration AND Initialization

- But what if we want to do both AT THE SAME TIME?!?
- `let <name>: <type> = <value>;`
- `let x: number = 10;`
- `let name: string = “Kris Jordan”;`
- `let isWednesday: boolean = true;`

Types

Type	Values
number	Any number (1, 2, 5.5, 6/7, etc)
string	“Strings” of characters (“hello”, “yes”)
boolean	true and false

- These are not all of the types in TypeScript, these are just the value types

Strings

- Strings of characters
- Special operation called **concatenation**
- Ex:
- `let name = await promptString(“What is your name?”);`
- `print(“Hello ” + name + “!”);`

Statements

- Statements are like the sentences of code
- Statements are a composition of expressions
- Expressions are made up of variables and operators
- Ex:
- `1 * 2  * 3;`
- `(x + y) / 100;`

Statements

- Always evaluates to a single value of a specific type
- All statements end with a semicolon
- `let num: number;`
- `num = 5;`
- `num = 7 * 2 + 1 * 6;`

Arithmetic Operators

- Standard +,-,*, / symbols
- Remember your order of operations! PEMDAS
- Can be used in variable initialization
- Ex:
- `let num: number = 15;`
- `let num2: number = (num - 10) * 5;`

Remainder operator - % *modulo*

- Divides numbers, the result is the remainder
- Ex:
- `7 % 2 = 1`
- `8 % 3 = 2`
- `15 % 3 = 0`

- `3 % 5 = 3`

modulo application

- See if you can spot the pattern
- `0 % 2 === 0`
- `1 % 2 === 1`
- `2 % 2 === 0`
- `3 % 2 === 1`
- `4 % 2 === 0`
- `5 % 2 === 1`
- `etc`

Relational Operators

- Relational operators are the basis for conditional statements
- **Always** evaluate to a boolean value
- Ex:
- `5 > 3`
- `6 !== 6`
- `“Yes” === “YES”`

Test	Math	TypeScript Operator
"is greater than?"	>	>
"is at least?"	≥	>=
"is less than?"	<	<
"is at most?"	≤	<=
"is equal to?"	=	===
"is not equal to?"	≠	!==

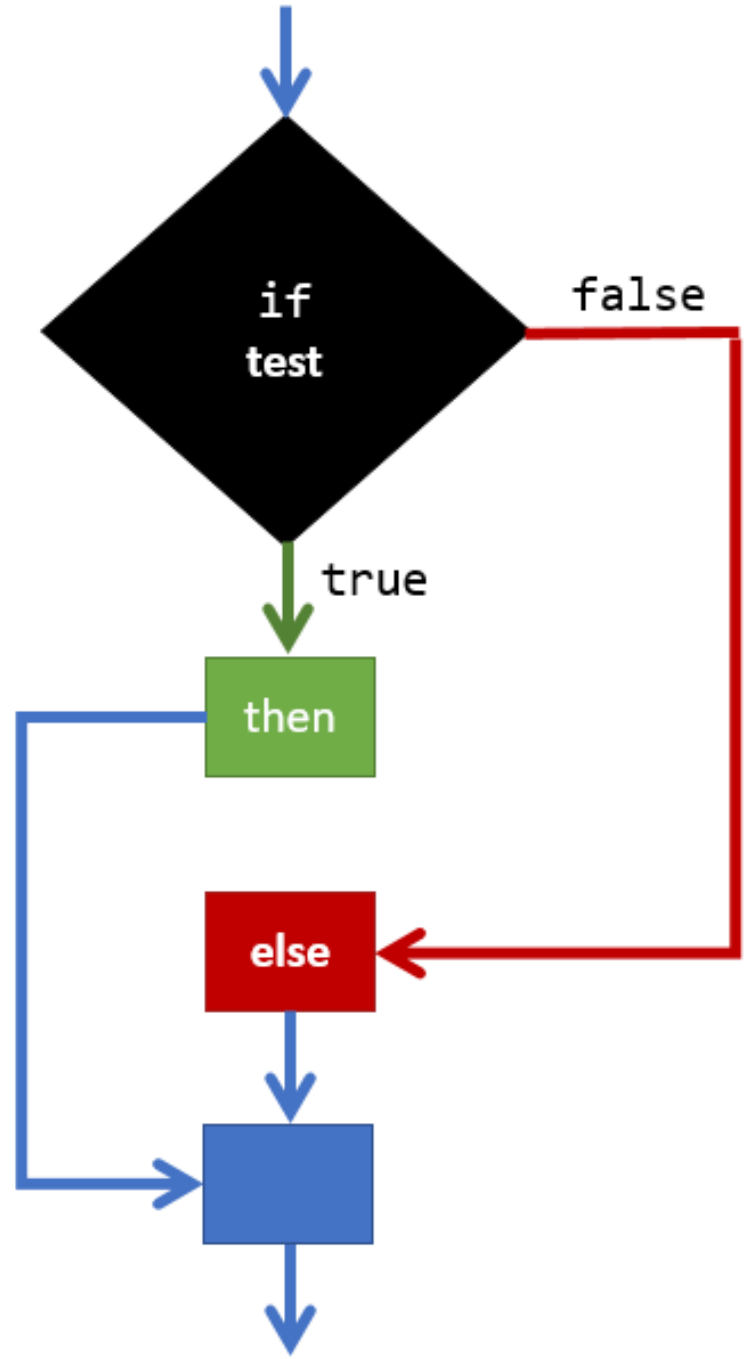
Assignment vs Equality

- Two similar operators, two totally different functions
- = is different from ===
- = is the **assignment** operator (`let i: number = 0;`)
- === is the **equality** operator
- `“COMP110” === “COMP” + 110;`

Control flow

- Allows your program to do different things (without your direct interference)

```
if (<condition>) {
   //then run this code
} else {
   //then run this code
}
```



If – then – else

```
let isRaining: boolean = await promptBoolean(“Raining?”);

if (isRaining === true) {
      print(“Bring your umbrella!”);
} else {
      print(“Don’t bring your umbrella!”);
}

print(“Don’t be late for class!”);
```

If – then – else

```
let isRaining: boolean = await promptBoolean(“Raining?”);

if (isRaining) {
      print(“Bring your umbrella!”);
} else {
      print(“Don’t bring your umbrella!”);
}

print(“Don’t be late for class!”);
```

isRaining	!isRaining	isRaining === true	isRaining === false
true	false	true	false
false	true	false	true

|| and &&

- Two more operators, used in conditional logic
- Or and And
- Ex
- `let num: number = 5;`
- `num === 6 || num > 3;`
- `num >= 5 && num < 100;`
- `true || false;`
- `true && false;`

Nested if

- You can have multiple if - then - else statements for more complex conditions

```
if (<conditional>) {
     if (<conditional>) {
          //do this
     } else {
          //do this instead
     }
} else {
     if (<conditional>) {}
}
```

```
let dayOfWeek: number;
dayOfWeek = await promptNumber("What day is it?");
if (dayofWeek === 7 || dayOfWeek === 1){
      print("Wow it's the weekend this is great news!");
      if (dayOfWeek === 1){
            print("Finish your homework!!!");
      } else {
            print("Relax and catch up on sleep");
      }
} else {
      if (dayOfWeek === 2){
            print(":(((((((");
      }
      else {
            print("Need more coffee");
      }
}
```

Sunday = 1
Monday = 2
Tuesday = 3
Wednesday = 4
Thursday = 5
Friday = 6
Saturday = 7

Hands-on

- Let’s write some code! Break out a pencil and paper.
- Write a nested if statement to tell us what greeting is appropriate
- Declare a variable `time` (a number from 0 to 24)
- Print “Good Morning” if `time` is between 0 and 12
- Print “Good Afternoon” if `time` is between 12 and 17
- Print “Good Evening” if `time` is between 17 and 21
- Print “Good Night” if `time` is between 21 and 24

Solution

```
let time: number = await promptNumber("What time is it?");
if (time >= 0 && time < 12) {
  print("Good Morning");
} else {
  if (time >= 12 && time < 17) {
    print("Good Afternoon");
  } else {
    if (time >= 17 && time < 21) {
      print("Good Evening");
    } else {
      if (time >= 21 && time <= 24) {
        print("Good Night");
      } else {
        print("This is not a time.");
      }
    }
  }
}
```