Quiz 2 Review Session

Outline

- For loops
- While vs for loops
- Arrays / Appending elements to arrays
- For loops with arrays
- **Hot date and check-in**
- Linear search
- Testing and Edge cases
- Practice questions

For Loops: What are Those??

A conditional iterative statement! Just another kind of loop

> Repeats a block of code as long as some conditional statement is true

General form:

```
for (let i: number = 0; i < 5; i++) {
    // repeat block
}
```

Bbbbbbreak it down

3 parts

1. Declaration & initialization of counter variable

2. Conditional statement

3. Modify counter variable (still happens at end of repeating block!)

```
for (let i: number = 0; i < 5; i++) {
    // repeat block
}
```

While Loops vs. For Loops

Counter variable scoped **outside** the repeating block

Good for when you **don't** know how many times you'll want to loop

(or, if you're checking a boolean variable as your condition)

Counter variable scoped **inside** the repeating block

Good for when you **do** know how many times you'll want to loop

While vs. For Loops example

```
let str: string = "";
let i: number = 0;
while (i < 10) {
    str = str + "yah ";
    i++;
}
print(str);
```

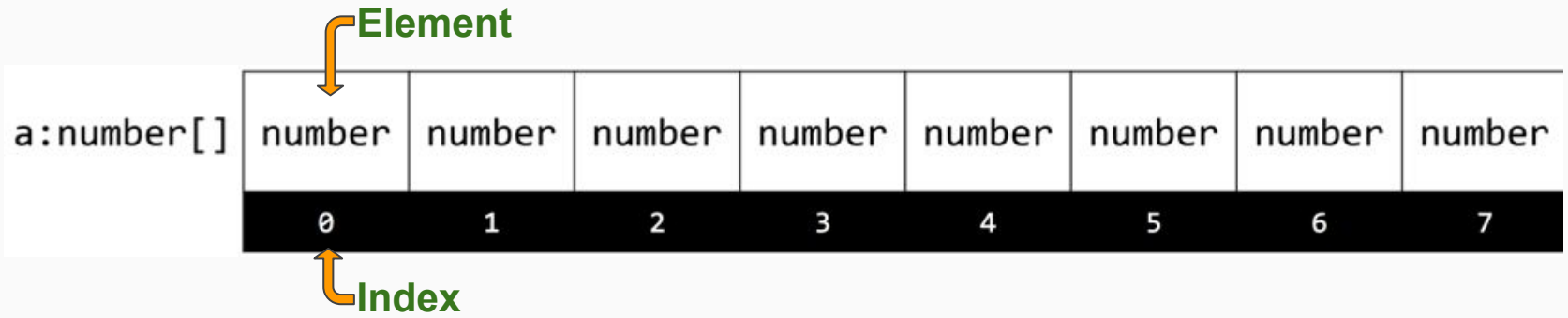
```
let str: string = "";
for (let i: number = 0; i < 10; i++) {
    str = str + "yah ";
}
print(str);
```

Both print:

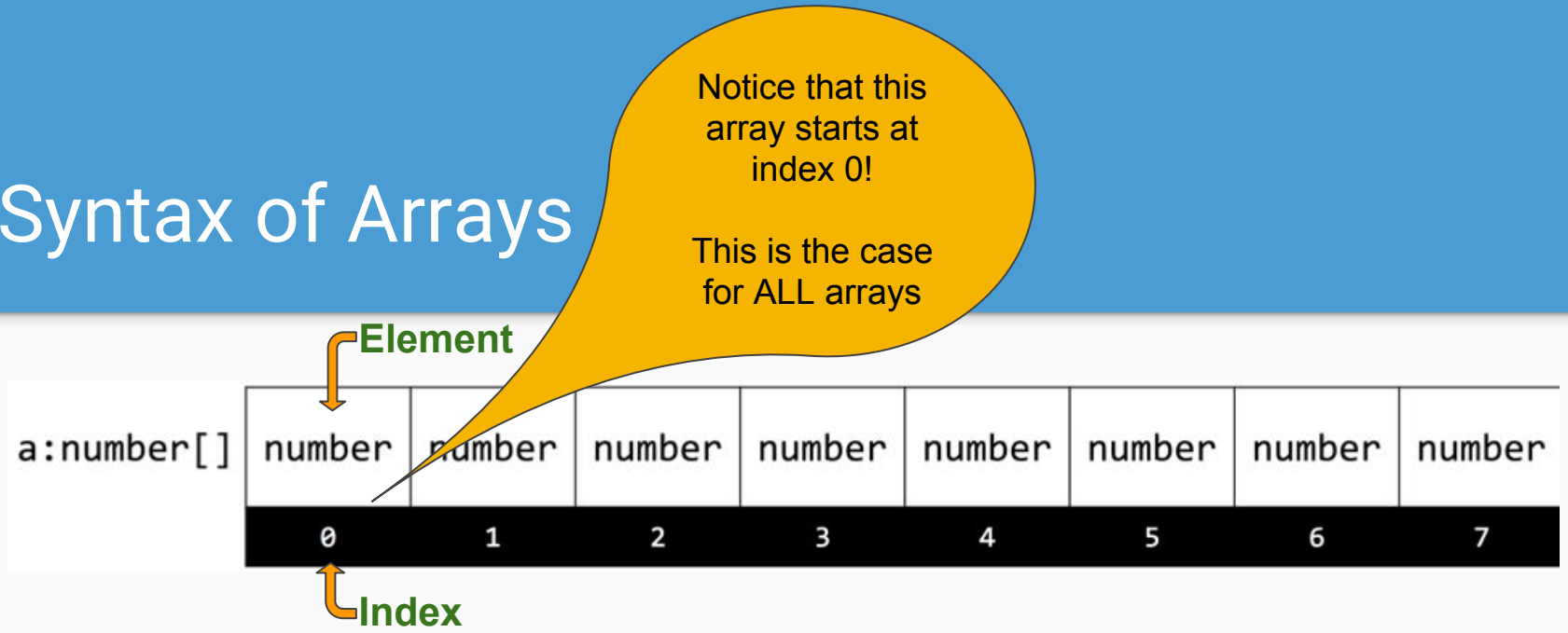
yah yah yah yah yah yah yah yah yah yah
string

Arrays

- Named variable that can hold multiple values
- The **element** (value) can be addressed by an **index** number
  - First index is **always 0!**
- Elements in an array all have to be of the **same type**
  - Can be any type -- number, string, boolean, object, etc.



Syntax of Arrays



Syntax for declaring an array:
`let <name>: <type>[];`

```
let students: string[];

let grades: number[] = [];
```

Construct an empty array by initializing with empty square brackets: **`let <name>: <type>[] = [];`**

Construct an array with elements: **`let <name>: <type>[] = [<value>, <value>, <value>];`**

Appending Elements to Arrays

To add an element at a specific index of an array:

<name>[<index>] = <value>;

To add an element to the **end** of an array:

<name>[<name>.length] = <value>;

Determining the next available index to append:

- For an empty array [], next available index is 0
- Array of 1 element, next available index is 1
- Array of 7 elements, next available index is 7

Appending Elements to Arrays

Ex 1) To put the string “Jim” as the **7th element** in an array named office:

```
office[6] = "Jim";
```

… why index 6?
0-based indexing!

Ex 2) To append the string “Oscar” to the end of the same array:

```
office[office.length] = "Oscar";
```

For Loops with Arrays

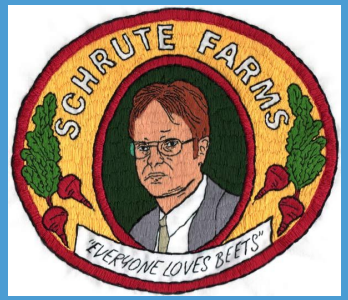
Useful when:

- Changing every element in an array
- Executing a piece of code based on the value of every element
- Collecting user input

Common syntax for an array a:

```
for (int i = 0; i < a.length; i++) {
        // code using a[i] to access each
            element, one by one
}
```

For Loops with Arrays: Example



The following array includes Dwight's monthly beet harvest amounts for 2018:

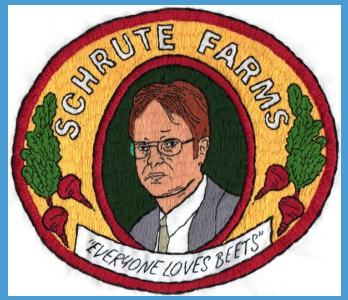
```
let a: number[] = [77, 95, 100, 123, 96, 68];
```

He expects for monthly beet harvests to increase by 5% in 2019. What should he expect for his harvest amounts to be?

```
for (let i = 0; i < beets.length; i++) {
    beets[i] = beets[i] * 1.05;
}
```

Results: 80.85, 99.75, 105, 129.15, 100.8, 71.4

For Loops with Arrays: Example



Dwight is only happy with his beet season if there are at least 3 months wherein he harvests at least 100 beets. He wasn't able to achieve this goal in 2018. Will he be able to achieve it in 2019?

80.85, 99.75, 105, 129.15, 100.8, 71.4

Yes! (goodMonth is 3)
Strong beet szn

```
let goodMonth: number = 0;
for (let i = 0; i < beets.length; i++) {
    beets[i] = beets[i] * 1.05;
    if (beets[i] > 100) {
        goodMonth += 1;
    }
}
if (goodMonth >= 3) {
    print("Strong beet szn");
} else {
    print("Maybe next year");
}
```

Reference for Arrays!

Operation	Form	Example
Declaration	`let <name>: <type>[];`	`let scores: number[];`
Construction (Empty)	`<name> = [];`	`scores = [];`
Construction (Non-empty)	`<name> = [<values>];`	`scores = [12, 0, 9];`
# of Elements	`<name>.length`	`scores.length`
Access Element	`<name>[<index>]`	`scores[0]`
Assign Element	`<name>[<index>] = <expression>;`	`scores[1] = 12;`
Append Element	`<name>[<name>.length] = <expr>;`	`scores[scores.length] = 13;`

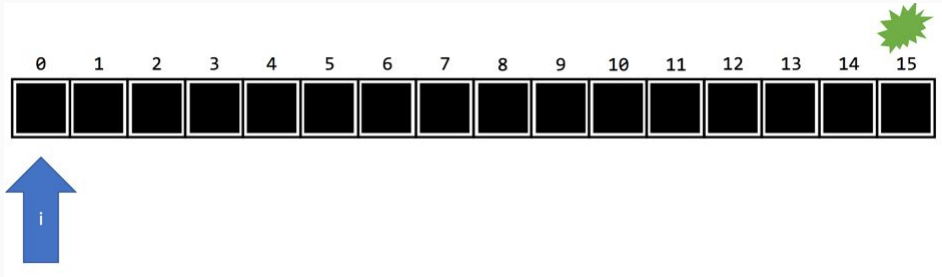
Hot date and Check-in

Check-in Code: **69D26**

Hot date: Talk to your neighbor about your favorite movie you've seen recently

Linear Search

- Searching for something, starting at the beginning of a line and moving one by one until you find it (or don't)
  - ***Loop*** until you find it, or reach the end of the array!
- Return the first occurrence of what we're looking for
  - If we reach the end of the array without finding it, return -1



Linear Search: Example

Help me find my friend Jules in a line of people!

```
let findFriend = (people: string[], name: string): number => {
    for (let m = 0; m < people.length; m++) {
        if (people[m] === name) {
            return m;
        }
    }
    return -1;
};
```

Linear Search: Example

```
let findFriend = (people: string[], name: string): number => {
    for (let m = 0; m < people.length; m++) {
        if (people[m] === name) {
            return m;
        }
    }
    return -1;
};
```

```
let a: string[] = ["Brooks", "Kaki", "Madison", "Kiet", "Carly", "Hank"];
```

What will this return? `findFriend(a, "Jules");` -1

```
let b: string[] = ["Brooks", "Kaki", "Madison", "Kiet", "Jules", "Hank"];
```

What will this return? `findFriend(b, "Jules");` 4

Testing

How can we know if our code works the way we want it to?

What are some things that our code **doesn't** do?

Testing

Testing is the act of ensuring that the code you have written performs the functionality you intend it to.

Testing can be either manual or programmatic.

Use Cases

Things your code **should** do.

Edge Cases

Things your code **should** do, but might be use cases that are a little out of the ordinary.

Try and think about what might cause something incorrect to happen to your program, **what are the boundaries of your conditions?**

Finding Cases

What are some use and edge cases for this function?

This function takes in a string array and a string, and returns a boolean value that indicates if the string is contained in the array.

```
let edgeUse = (a: string[], s: string): boolean
```

Finding Cases

```
let edgeUse = (a: string[], s: string): boolean => {
  for (let i = 0; i < a.length; i++) {
    if (a[i] === s) {
      return true;
    }
  }
  return false;
};
```

Summary

Testing helps verify functionality, but doesn’t guarantee it.

The act of thinking of how you would test your code is *almost* as useful as actually testing it (at this level.)

Practice #1

```
/*
 * What's the expected output of arrayThing(10)?
 */
let arrayThing = (n: number): number[] => {
  let a: number[] = [1];
  for (let i = 1; i < n; i++) {
    a[i] = i % 3;
  }
  a[a.length - 1] = 3;
  return a;
};                    // [1, 1, 2, 0, 1, 2, 0, 1, 2, 3];
```

Practice #2

```
/*
 * First off, what does this function do?
 * What does it return?
 * What are input values that break this function?
 */
let err = (a: number[], b: number): void => {
  let i = 0;
  let count = 0;
  while (i < a.length) {
    print("Here's an element: " + a[i] + "!");
    i += b;
    count++;
  }             // Functionality: Prints every b'st / b'nd / b'rd number
};              // Breaks With: Any negative number as b
```

Practice #3

Write a function that takes in 2 number arrays and adds the elements at corresponding indices together

Ex: [1, 2, 3], [4, 5, 6] added together is [5, 7, 9]

Practice #3 answer

What's the edge case that I handled in this function? How did I handle it?

```
let addArrays = (a: number[], b: number[]): number[] => {
  if (a.length !== b.length) {
    return [];
  }
  let c = [];
  for (let i = 0; i < a.length; i++) {
    c[i] = a[i] + b[i];
  }
  return c;
};
```