Keep Sitting with YOUR PARTNER at your Assigned Tables!

Destructuring, Rest Syntax, Spread Syntax

COMP423 / Fall 2024 / Lecture 11

Diagram the following code listing...

```
1    const nums = [1, 2, 3, 4];
2
3    const [a, b, c, d] = nums;
4
5    console.log(a, b, c, d);
```

Destructuring Assignment Statements

Compare the following equivalent code listings...

```
1    const nums = [1, 2, 3, 4];
2
3    const a = nums[0];
4    const b = nums[1];
5    const c = nums[2];
6    const d = nums[3];
7
8    console.log(a, b, c, d);
```

```
1    const nums = [1, 2, 3, 4];
2
3    const [a, b, c, d] = nums;
4
5    console.log(a, b, c, d);
```

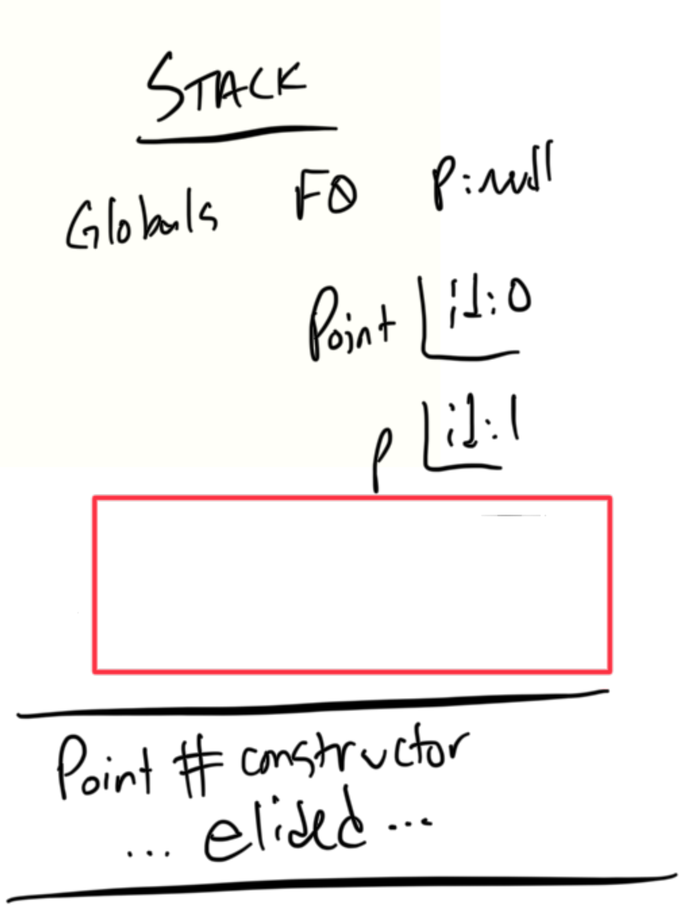
Diagram the following code listing...

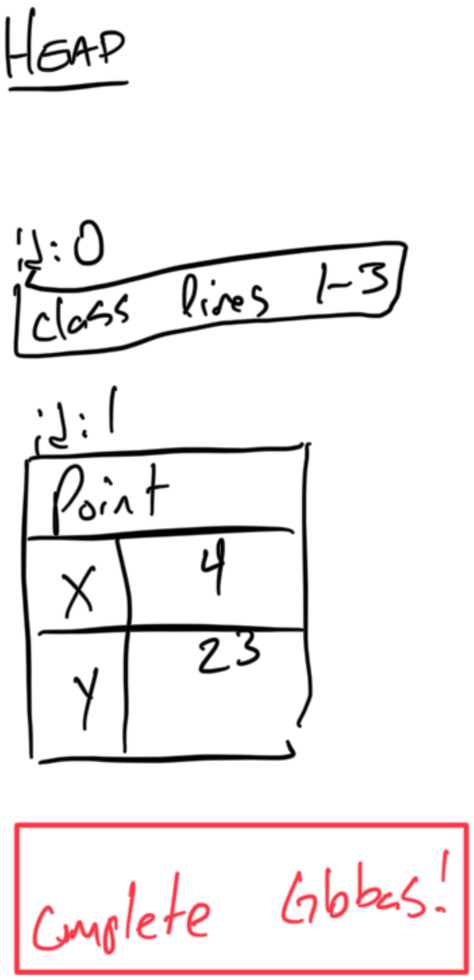
```
1    let a = 590;
2    let b = 423;
3
4    [a, b] = [b, a];
5
6    console.log(a, b);
```

Diagram the Following Code Listing

Use the starting point to the right and just complete what changes in Globals...

```
1    class Point {
2      constructor(public x: number, public y: number) {}
3    }
4
5    const p = new Point(4, 23);
6
7    const {x, y} = p;
8
9    console.log(x + y);
```





Rest Syntax ...

Diagram and then Discuss: What data types are `nums`, `first`, and `rest`?

```
1    let nums = [1, 2, 3, 4];
2
3    let [first, ...rest] = nums;
4
5    console.log(first);
6    console.log(rest);
```

Functions with Variadic Arguments (varargs)

Defining functions with rest parameters

```
let multiply = (n: number, ...xs: number[]) => xs.map(x => n * x);
```

```
let multiply = (n: number, ...xs: number[]): number[] => {
  let results: number[] = [];
  for (let x of xs) {
    results.push(n * x);
  }
  return results;
};
```

1. There are two implementations of a `multiply` function. Are they equivalent?

2. Produce an example function call to `multiply` that returns the array [2, 4, 6]?

Spread Syntax ...

These ellipses have a *very* different meaning!

```
1    const odds = [1, 3];
2    const evens = [2, 4];
3    const nums = [0, ...odds, ...evens, 5];
4    console.log(nums); // What is the output?
```

Spread Arguments ...

Trace the output of the following snippet:

```
1    const mul = (n: number, ...xs: number[]) => xs.map(x => n * x);
2    console.log(mul(3, 4, 5));
3
4    const nums: number[] = [4, 5];
5    console.log(mul(3, ...nums));
```

Bringing it all together...

Which of these ellipses are *Rest?* Which are *Spread?*

```
interface AnyFunction {
  (...args: any[]): any;
}

const logged = (f: AnyFunction) => {
  return (...args: any[]) => {
    console.log("args:", args)
    let rv = f(...args);
    console.log("rv:", rv);
    return rv;
  };
};

let add = (x: number, y: number) => {
  return x + y;
};
add = logged(add);

console.log(add(1, 2));
```