Recursion Preview

Lecture 11 – Spring 2019

VSCode: npm run pull / npm start

PollEv.com/compunc

Path Forward

- Quiz Thursday - Emphasis on Environment Diagrams and Reference Types (Objects/Arrays)

- PS03 - Weather Stats - Due Monday 3/4

- **Whiteboard Only Office Hours** are in Effect through 3/4!
  - If you have a black screen of death or nothing is showing up and you and can't figure out why, the TAs can help you get your program compiling again.
  - Otherwise, laptops will be closed and discussion will revolve around notes or code you bring in on paper and/or produce on the whiteboard.
  - Every time we have done this in past semesters the feedback has been:
    "It felt annoying at first, but I actually wound up understanding the concepts I was confused about and it helped me on the next quiz."

- Next Up:
  - Object-oriented Programming
  - Recursion (Preview Today)
  - Functional Programming

```
3  export let main = async () => {
4      print(c4());
5  };
6
7  let c1 = (): number => {
8      return 1;
9  };
10
11 let c2 = (): number => {
12     return 1 + c1();
13 };
14
15 let c3 = (): number => {
16     return 1 + c2();
17 };
18
19 let c4 = (): number => {
20     return 1 + c3();
21 };
22
23 main();
```

CQ0. Draw an Environment Diagram

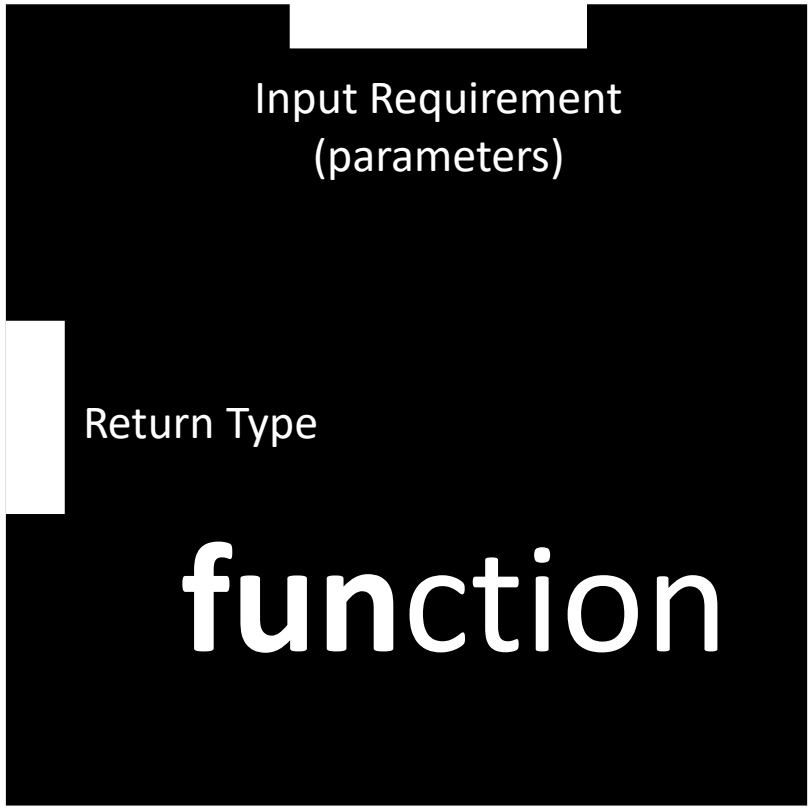
Answer the following questions on PollEv:

- How many names are bound globally?
- Not including the globals frame, how many frames on the stack are there?
- What is printed?

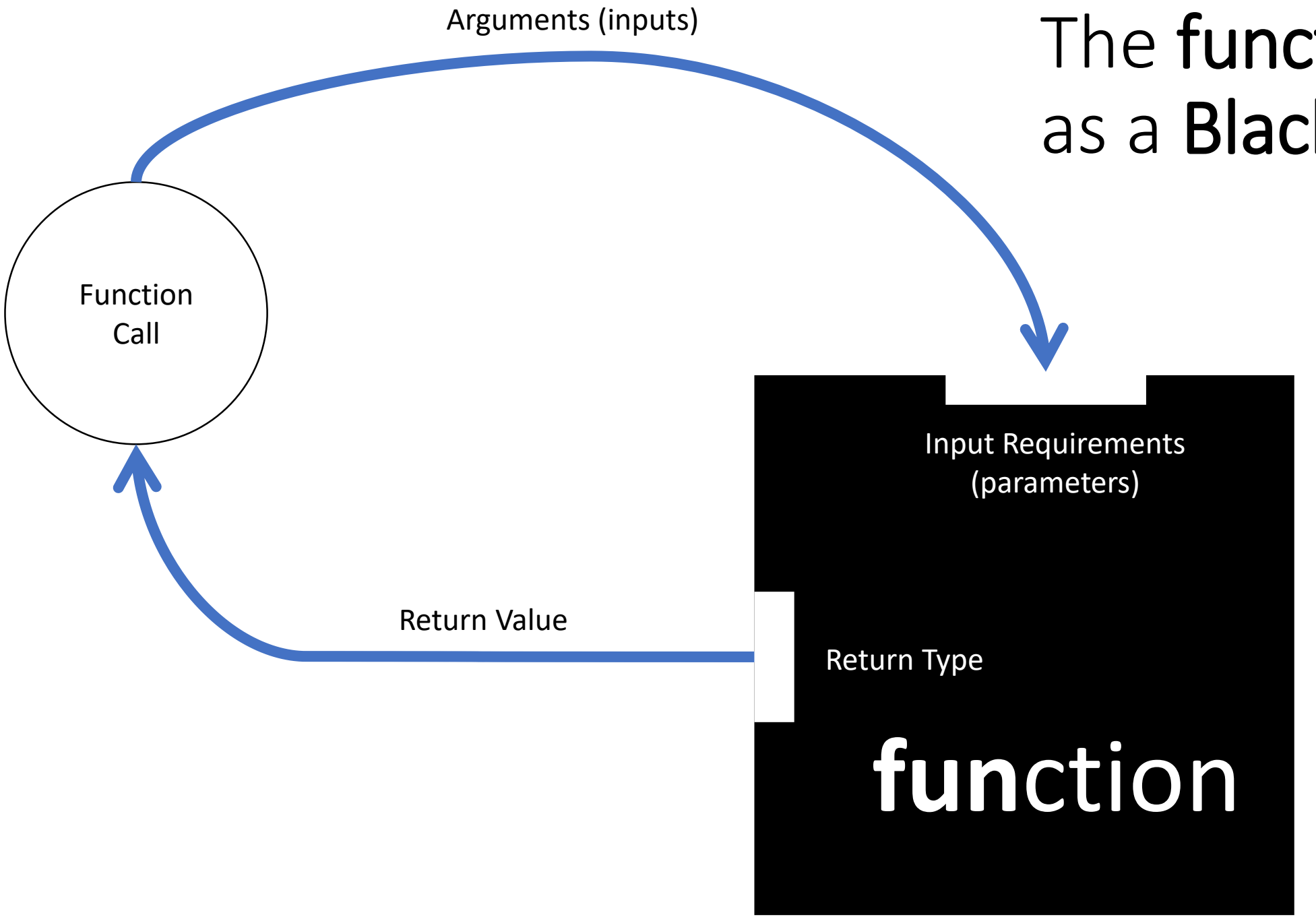
```
export let main = async () => {
    print(c4());
};

let c1 = (): number => {
    return 1;
};

let c2 = (): number => {
    return 1 + c1();
};

let c3 = (): number => {
    return 1 + c2();
};

let c4 = (): number => {
    return 1 + c3();
};

main();
```

The **function** as a **Black Box**

- Once a function is correctly implemented, we can think of it as a **"black box"**
- We do not need to know or see what happens inside of the black box...
  that's magic
- All we need to know is:
  1. What inputs does it need?
  2. What does it return back to us?
     (Or what effect does it have if `void`?)



The **function** as a **Black Box**



CQ1. Draw an Environment Diagram for the Following Code

```
1  import { print } from "introcs";
2
3  export let main = async () => {
4      print(sum(2));
5  };
6
7  let sum = (x: number): number => {
8      if (x < 1) {
9          return 0;
10     } else {
11         return x + sum(x - 1);
12     }
13 };
14
15 main();
```

Answer the following three questions on PollEv:

- How many names are bound globally?
- How many different values of the parameter x exist on the stack?
- What is printed?

```
import { print } from "introcs";

export let main = async () => {
    print(sum(2));
};

let sum = (x: number): number => {
    if (x < 1) {
        return 0;
    } else {
        return x + sum(x - 1);
    }
};

main();
```

Playing with Graphical Procedures

- Today we'll introduce a simple graphics library called Turtle Graphics
  - It's a style of teaching introductory computer science that dates back to 1967!

- We have a number of procedures available to us to guide an invisible "turtle" on the screen who is dragging around a marker…

  `forward(n: number): void` – Moves the turtle forward by **`n`** pixels
  `left(rad: number): void` – Turns the turtle left by **`rad`** in radians
  `right(rad: number): void` – Turns the turtle right by **`rad`** in radians
  `moveTo(x: number, y: number): void` - Moves turtle to **`x`**, **`y`** coordinate

- You can import these functions by:
  - **`import { forward, left, right, moveTo } from "introcs/turtle";`**

Hands-on: Draw a Square

- Open 11 / 00-square-app.ts

- Reference for the imported procedures:
  - **`forward(n: number): void`**
  - **`left(radians: number): void`**

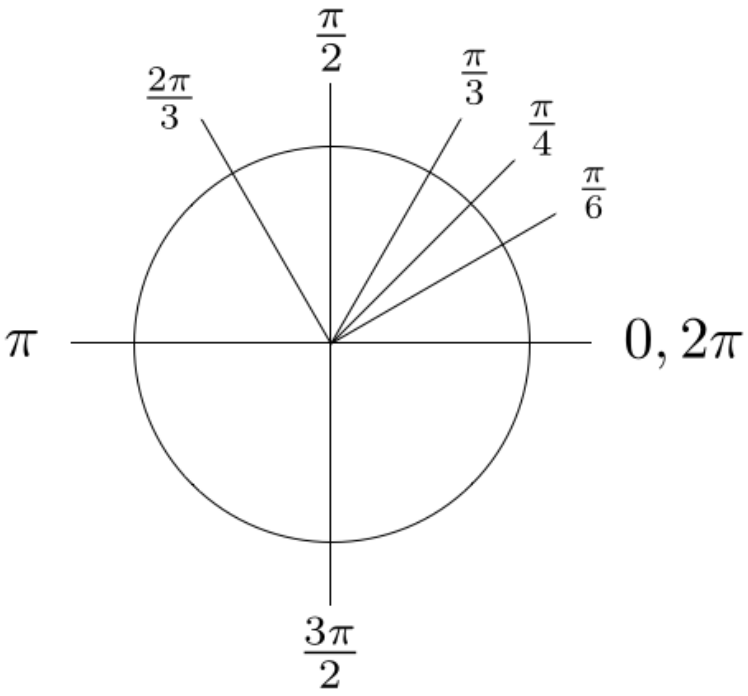
1. Implement the forwardTurn function based on its comments.
   Hint: The left function is in terms of radians. Use Math.PI for a reasonably precise value of PI and the chart right.

2. Implement the square function based on its comment.

3. Call the square function from main with a width argument of 100. You should see a 100-pixel wide

- Check-in on PollEv.com/compunc when complete

Radians Chart



Hands-on: Draw a Spiral

- Open 11 / 01-spiral-app.ts

- The logic for spiral will be defined *in terms of itself!*

- Draw a spiral by:
  1. Moving **forward** by **width** amount
  2. Turning **left** by 90 degrees (Math.PI / 2)
  3. **If** the **width** is **greater than 10**,
     - ***then*** call the **spiral** function with an **argument** of **97% of the current width**

- Check-in on PollEv.com/compunc when your spiral is spiraling out of control.

Recursive Tree Intuition

- Before we paint a happy, little tree recursively, let's explore the idea intuitively first…

- http://recursivedrawing.com/

- Add a branch
  - Recursively add another branch angled a bit to the left and smaller
  - Recursively add another branch angled a bit to the right and smaller

- Stop adding branches when the size of a branch gets too small to see

Follow-Along: Painting a Happy, Little Tree

- The **lineOut** function turns the turtle by some angle and traces a line along a branch.

- The **lineBack** function turns the turtle around, traces back along the branch, and resets its orientation to where it began.

- To draw a branch, we'll draw a line out and a line back.
  - The recursive branches will happen between the line out and the line back.