

NodeJS

Part 1

jan.schulz@devugees.org

Agenda

1. What is NodeJS?
2. Modules
3. Module Patterns
4. Native Modules
5. File System Module
6. Recursion

Before we start ...

Why do we need NodeJS?

Before we start ...

Why do we need NodeJS?

Web-Site:

HTML

CSS

JavaScript

Before we start ...

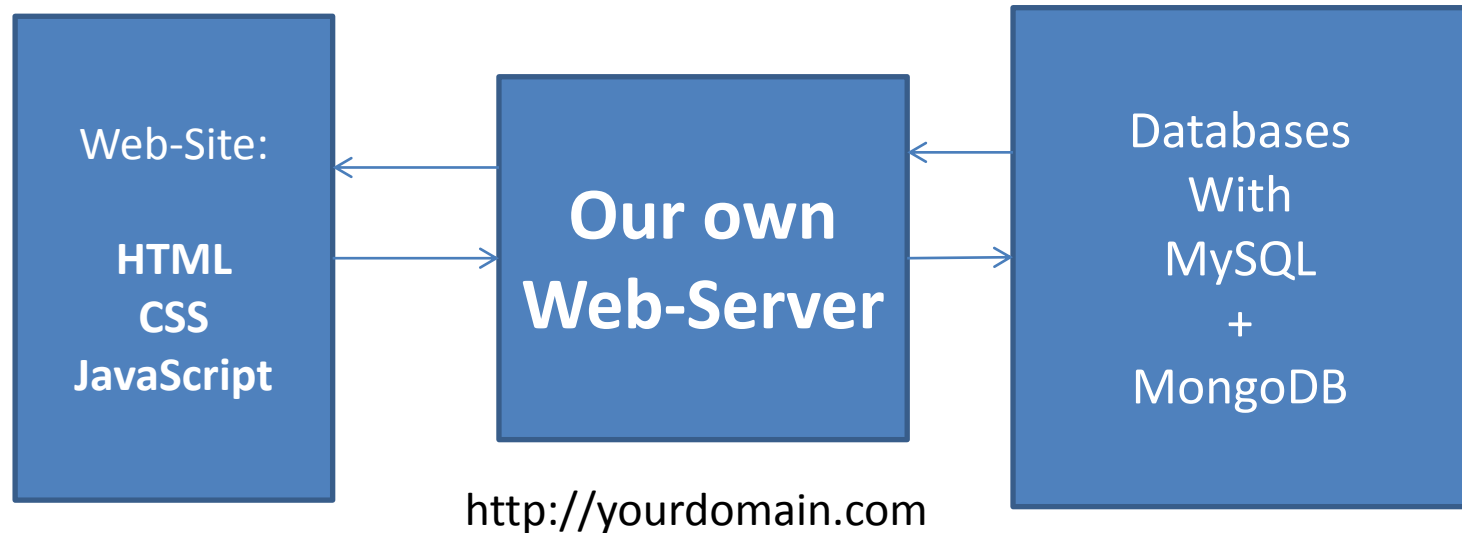
Why do we need NodeJS?



<http://yourdomain.com>

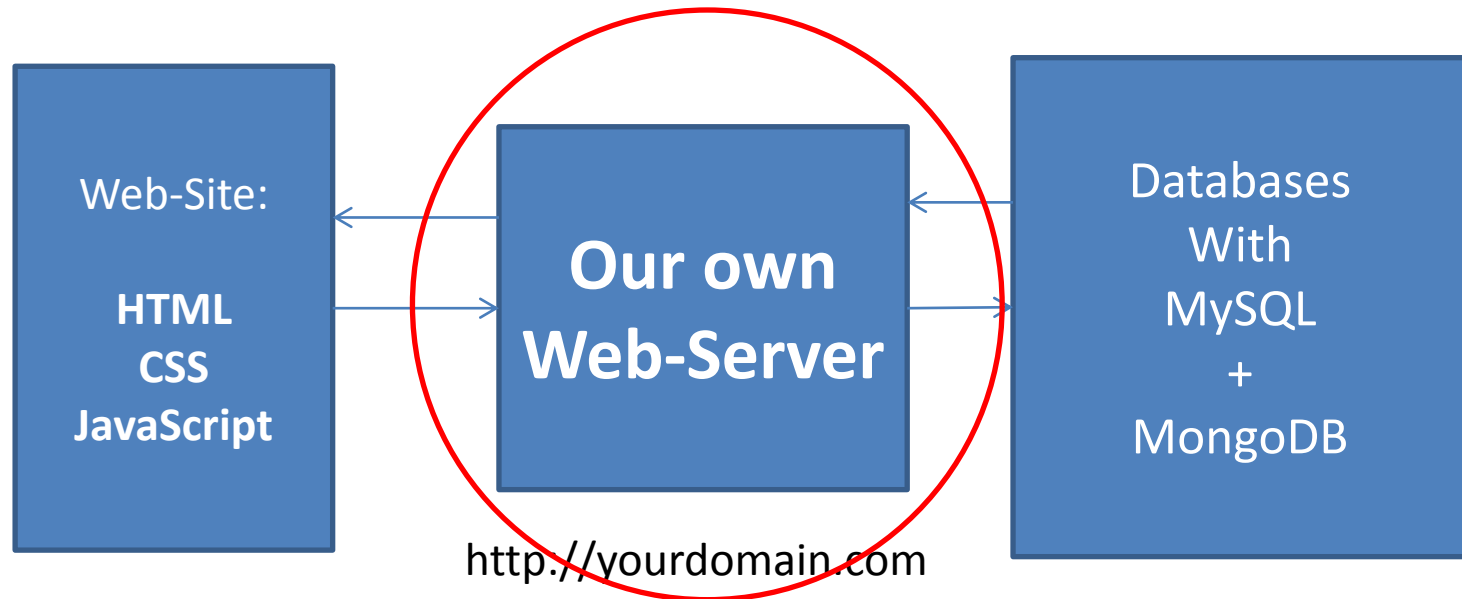
Before we start ...

Why do we need NodeJS?



Before we start ...

Why do we need NodeJS?



1. What is NodeJS?

NodeJS is an **run-time environment** for executing JavaScript code **server-side**.

- Non-blocking, event-driven programming paradigm

(More on that later in **Part 2/Asynchronicity**)

1. What is NodeJS?

- NodeJS operates JavaScript on the server-side, including
 - File Access
 - Database Access
 - Process Access
 - Network Access

1. What is NodeJS?

- However,

NodeJS is mostly used as HTTP-WebServer

2. Modules

Module:

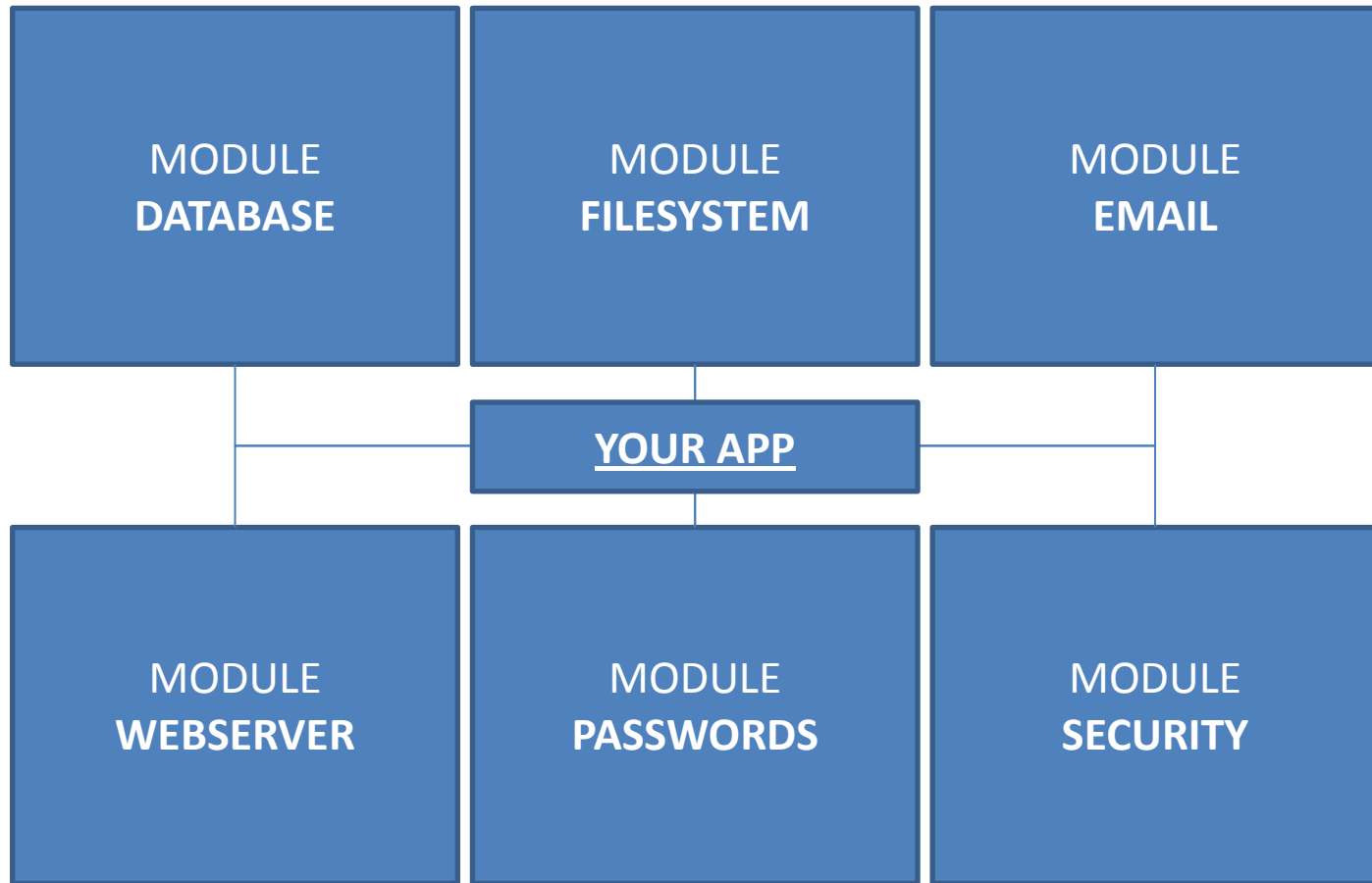
A reusable block of code whose existence does not accidentally impact other code.

= One of the fundamentals of NodeJS

2. Modules

**YOUR NODEJS APPLICATION
(i.e. Your Web-Server)**

2. Modules



2. Modules

- Lets build our first module ...

2. Modules

- Lets build our first module ...
- **require()** is a **function**, that you pass a 'path'
- *module.exports* is the **return statement of require()**

3. Module Patterns

- There are multiple ways of defining a module

4. Native Modules

- A list of native modules for NodeJS
- <https://nodejs.org/api>

5. File System Module

- Deals with files
 - Read: **readFileSync(path);**
 - Write: **writeFileSync(path, string);**
 - Append: **appendFileSync(path, string);**
 - Delete: **unlinkSync(path);**
 - Show contents of a folder: **readdirSync(path);**

6. Recursion

Recursion:

A function that calls itself.

```
function f(x) {  
    if(x === 0)  
        console.log('x is 0, end of recursion stack');  
    else  
        console.log('x is ' + x);  
        f(x-1);  
}
```

6. Recursion

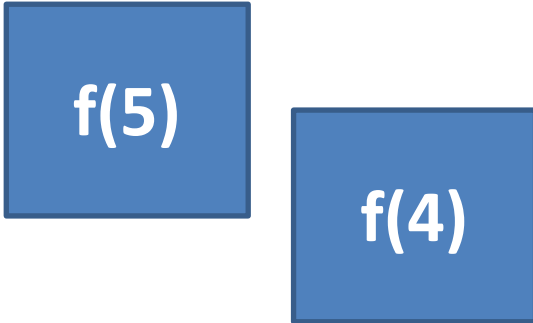
$f(5)$ is called.



$f(5)$

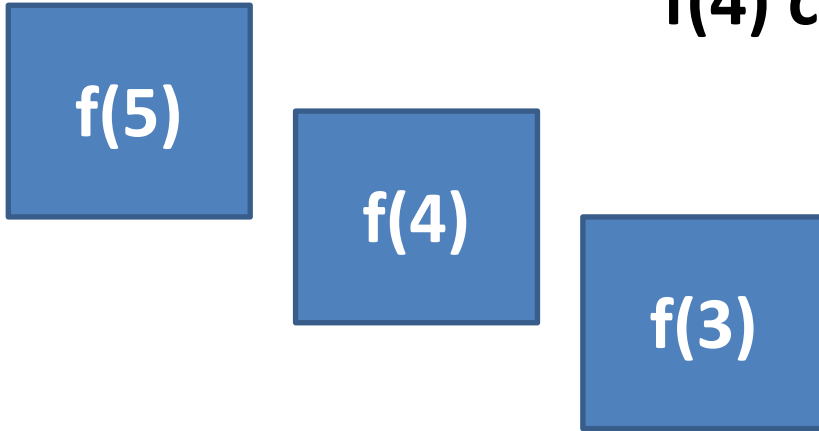
6. Recursion

f(5) calls f(4)



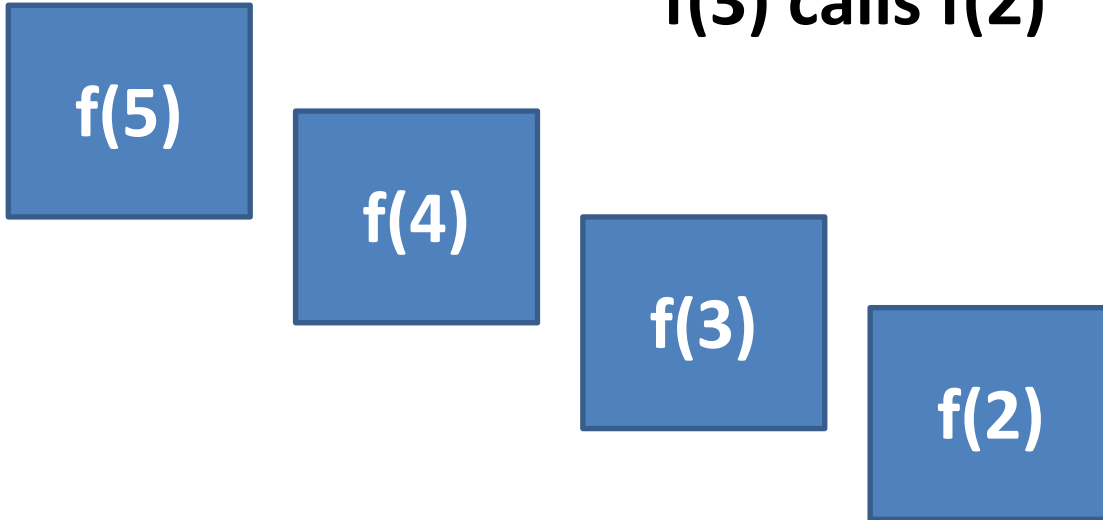
6. Recursion

f(4) calls f(3)



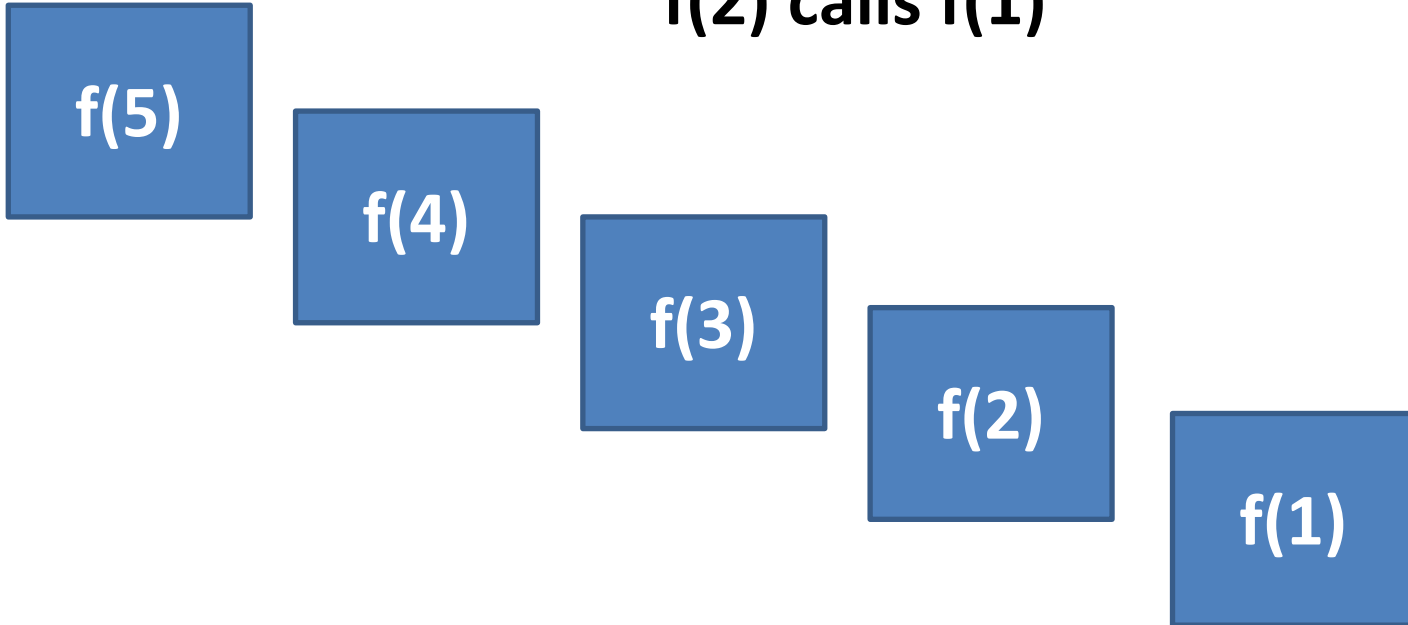
6. Recursion

f(3) calls f(2)



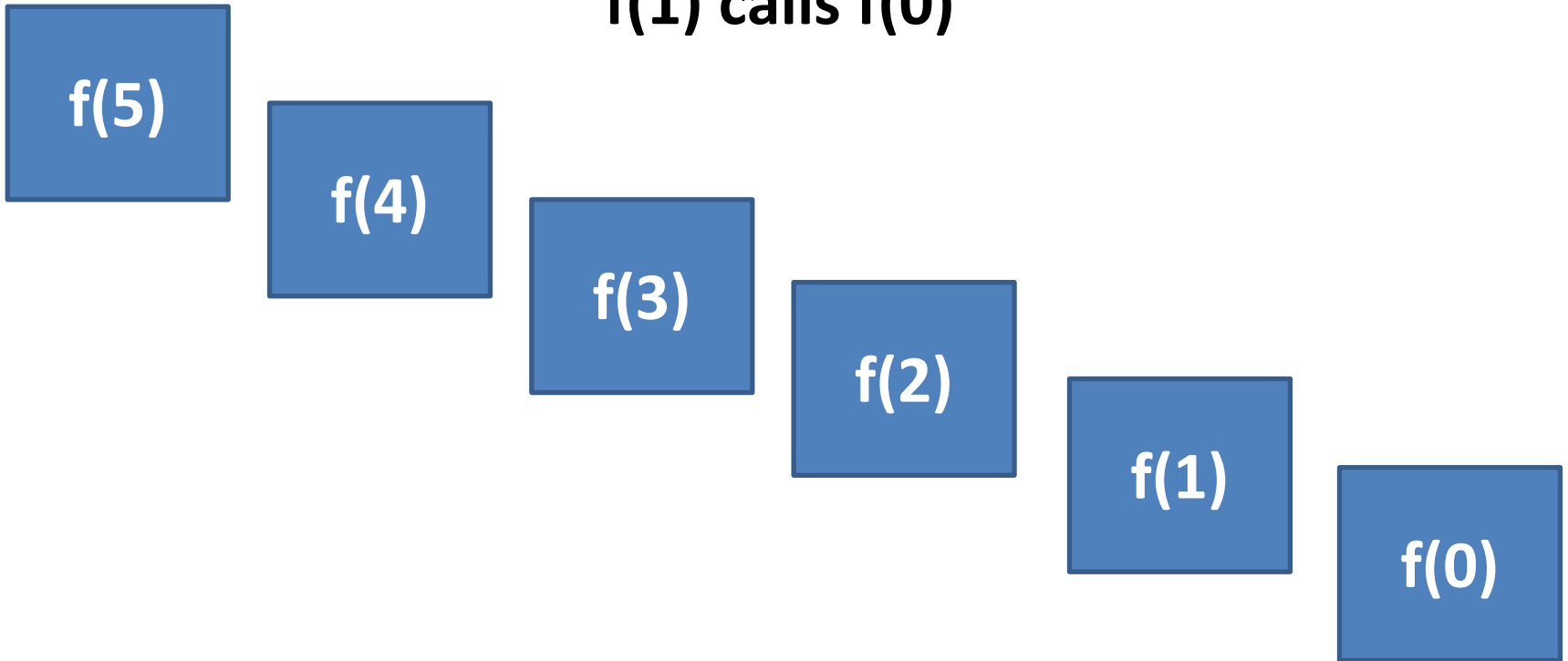
6. Recursion

f(2) calls f(1)



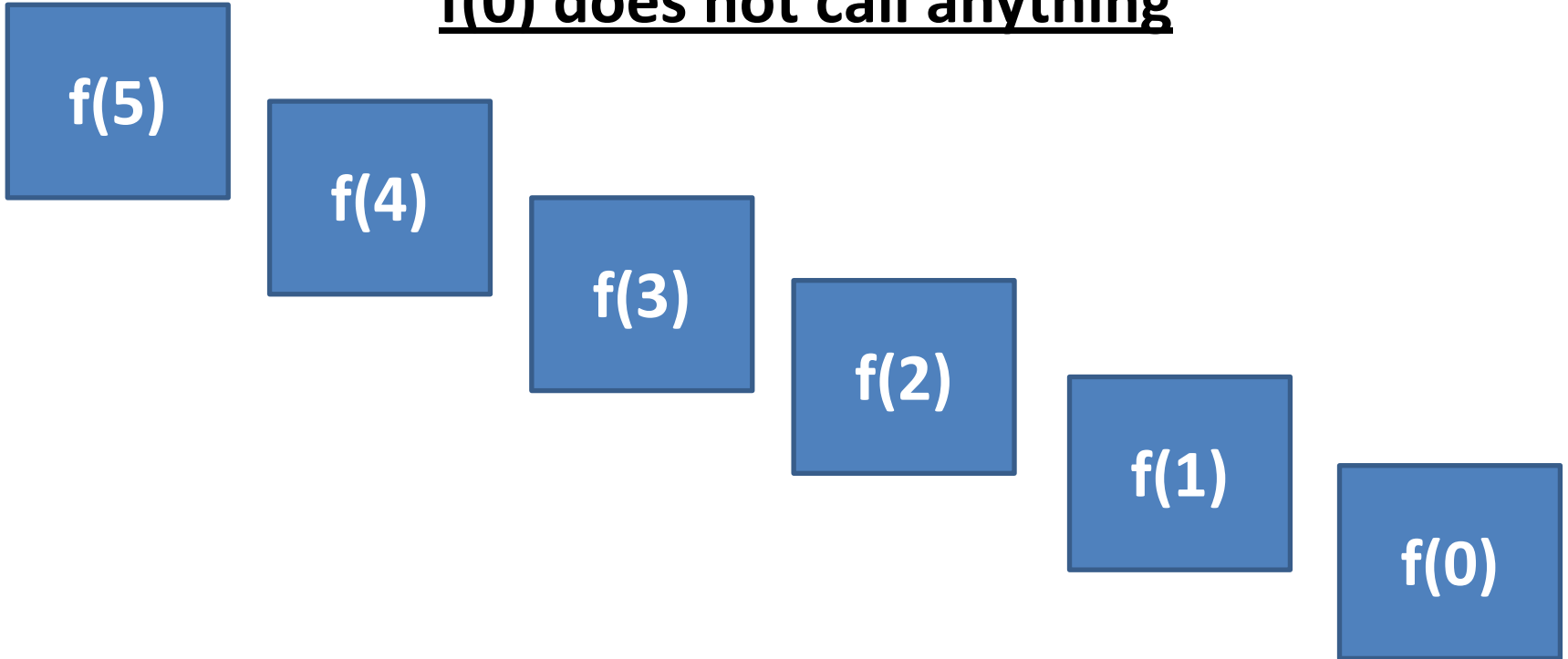
6. Recursion

f(1) calls f(0)



6. Recursion

f(0) does not call anything



end of recursion stack

6. Recursion

Task:

1. Write a function `sum(x)` which sums up all values from 1 to x. I.e. `sum(5)` would add up $1+2+3+4+5=15$. Use recursion for that.
- 2a. Write a function ***listFiles(path)*** which lists all files and folders of a given **path**.
- 2b. Extend **`listFiles(path)`** by also showing the subfolders. Implement that using recursion.