

FUNCTIONS

The last "big" topic



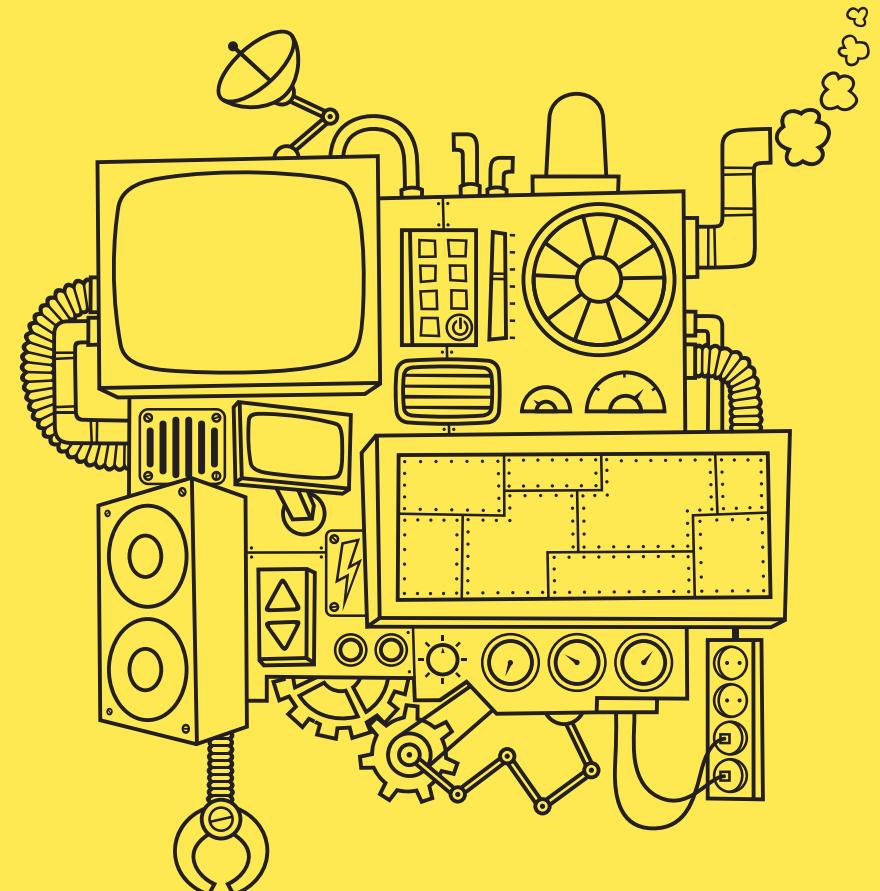
GOALS

- Write valid functions
- Write functions with arguments
- Compare function expressions & function statements
- Write a bunch of functions!

FUNCTIONS

Reusable procedures

- Functions allow us to write reusable, modular code
- We define a "chunk" of code that we can then execute at a later point.
- We use them ALL THE TIME



2 STEP PROCESS

DEFINE



RUN



DEFINE

```
function funcName() {  
    //do something  
}
```

DEFINE



```
function grumpus( ) {  
    console.log('ugh...you again...');  
    console.log('for the last time...');  
    console.log('LEAVE ME ALONE!!!!');  
}
```

RUN

```
funcName(); //run once
```

```
funcName(); //run again!
```

RUN



```
grumpus();
//ugh...you again...
//for the last time...
//LEAVE ME ALONE!!!
```

```
grumpus();
//ugh...you again...
//for the last time...
//LEAVE ME ALONE!!!
```



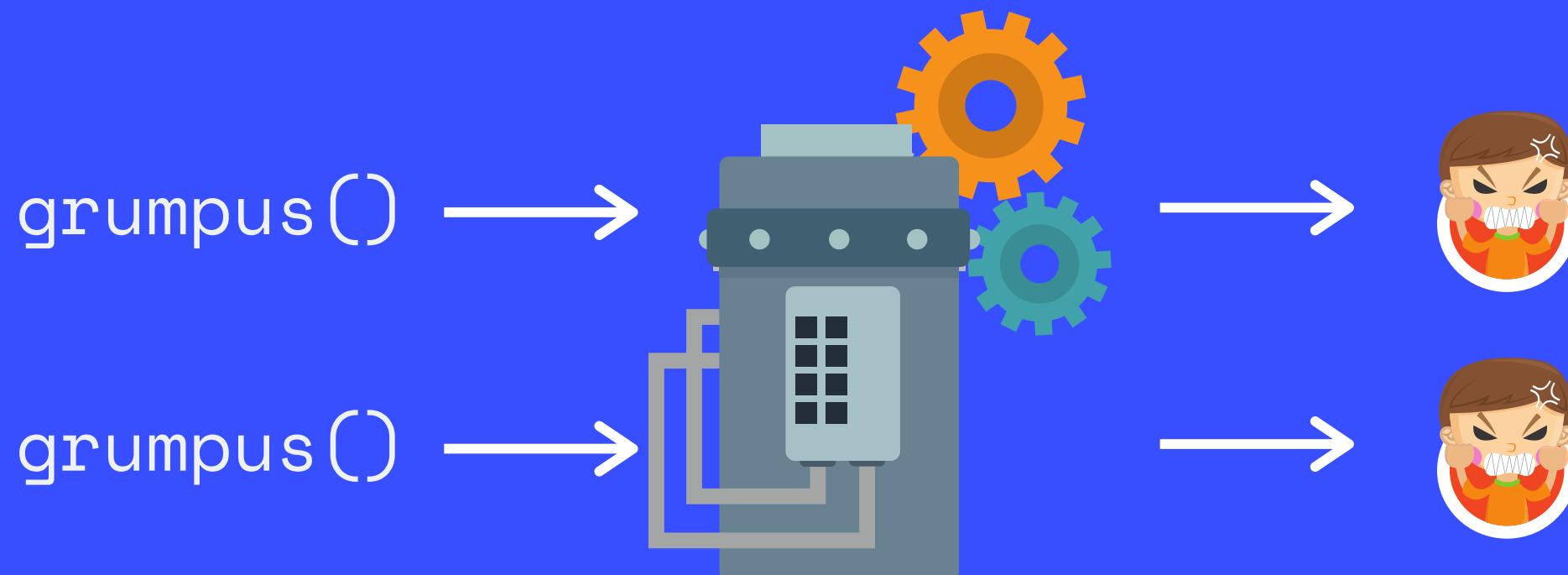
ARGUMENTS

A photograph of a man and a woman shouting at each other. The man, on the left, has a beard and is wearing a white t-shirt. The woman, on the right, has dark hair pulled back and is wearing a gold hoop earring. They are positioned against a background split into two colors: teal on the left and pink on the right. The word "ARGUMENTS" is overlaid in large, bold, black capital letters across the top of the image.

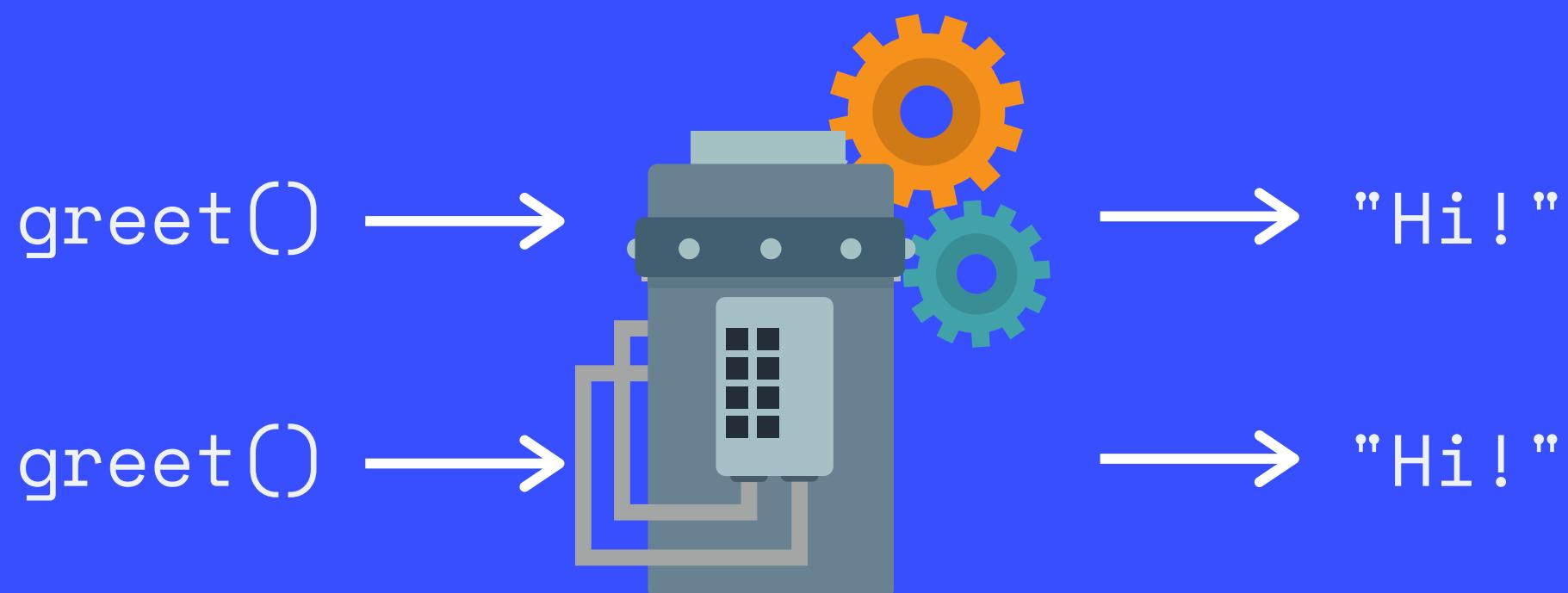
INPUTS

Right now, our simple functions accept zero inputs. They behave the same way every time.

NO INPUTS



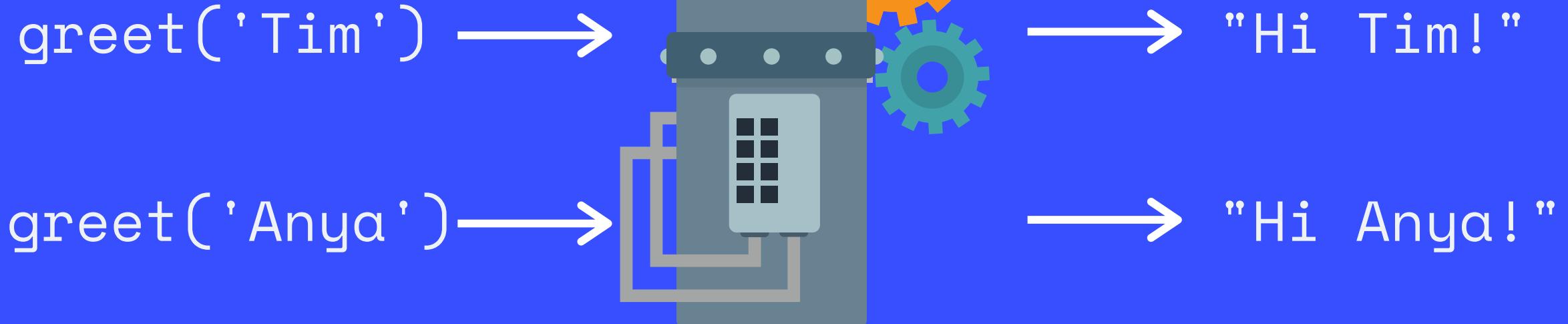
NO INPUTS



ARGUMENTS

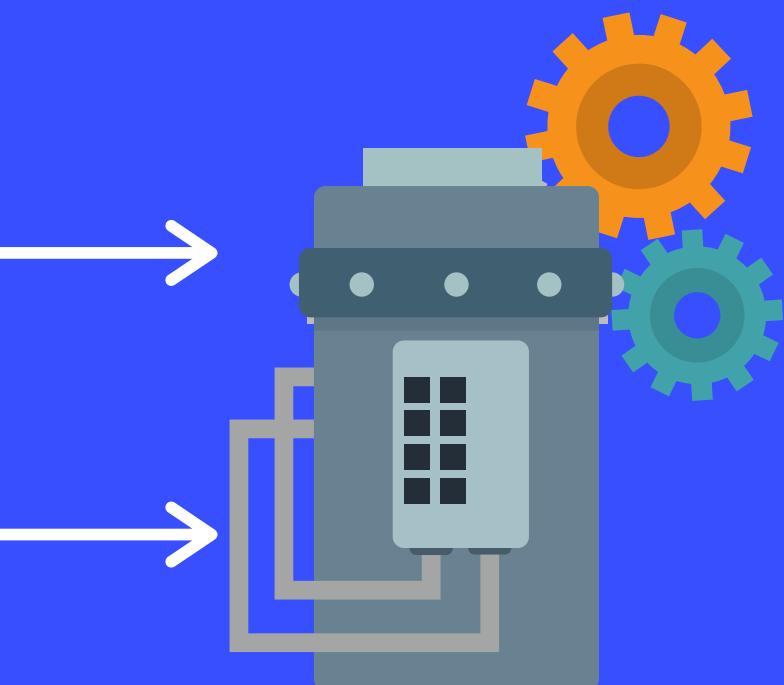
We can also write functions that accept inputs, called *arguments*

ARGUMENTS



ONE MORE

avg(20, 25)



22.5

avg(3, 2, 5, 6)



4

We've seen this before

No inputs

```
//No input  
"hello".toUpperCase();
```

Arguments!

```
//Different inputs...  
"hello".indexOf('h'); //0  
//Different outputs...  
"hello".indexOf('o'); //4
```

GREET TAKE 2



```
function greet(person) {  
  console.log(`Hi, ${person}!`);  
}
```



```
greet('Arya');
```



"Hi, Arya!"



```
greet('Ned');
```



"Hi, Ned!"

2 ARGUMENTS!



```
function findLargest(x, y) {  
    if (x > y) {  
        console.log(`${x} is larger!`);  
    }  
    else if (x < y) {  
        console.log(`${y} is larger!`);  
    }  
    else {  
        console.log(`${x} and ${y} are equal!`);  
    }  
}
```



findLargest(-2, 77)

"77 is
larger!"



findLargest(33, 33);

"33 and 33
are equal"

RETURN

Built-in methods **return** values when we call them.
We can store those values:



```
const yell = "I will end you".toUpperCase();

yell; // "I WILL END YOU"

const idx = ['a', 'b', 'c'].indexOf('c');

idx; // 2
```

NO RETURN!

Our functions print values out,
but do NOT return anything



```
● ● ●  
  
function add(x, y) {  
  console.log(x + y);  
}  
  
const sum = add(10, 16);  
sum; //undefined
```

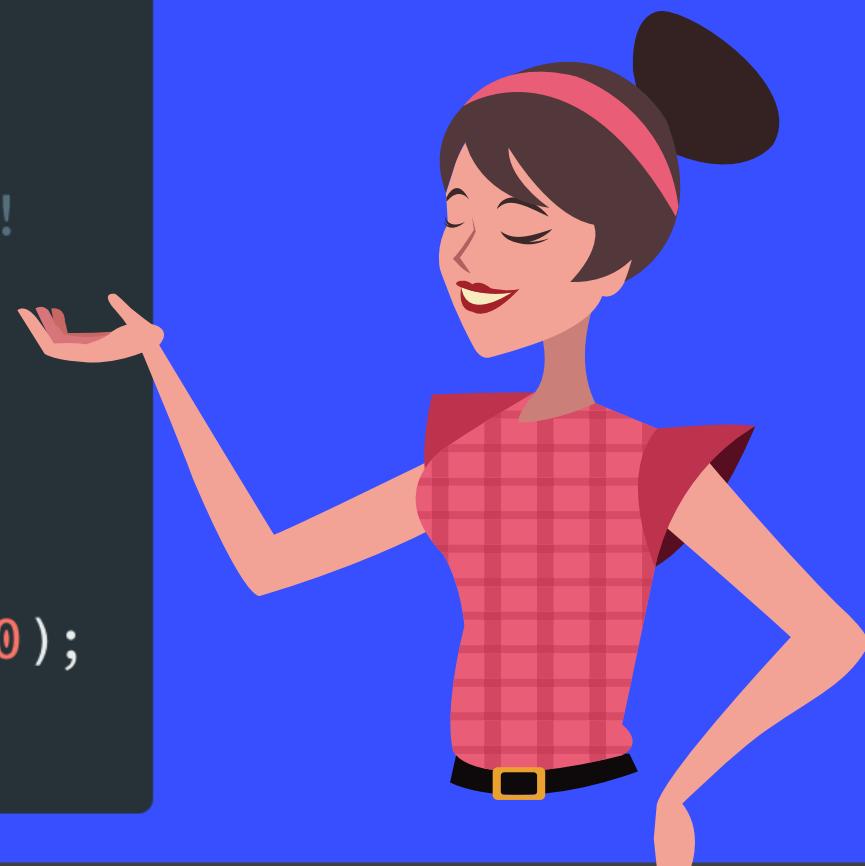


FIRST RETURN!

Now we can capture a return value in a variable!



```
function add(x, y) {  
    return x + y; //RETURN!  
}  
  
const sum = add(10, 16);  
sum; //26  
  
const answer = add(100, 200);  
answer; //300
```

A dark grey rectangular box containing a snippet of JavaScript code. The code defines a function named 'add' that takes two parameters, 'x' and 'y', and returns their sum. It then shows three examples of calling this function: first with arguments 10 and 16, resulting in the output 26; second with arguments 100 and 200, resulting in the output 300.

RETURN

The *return* statement ends function execution AND specifies the value to be returned by that function

PRACTICE

