



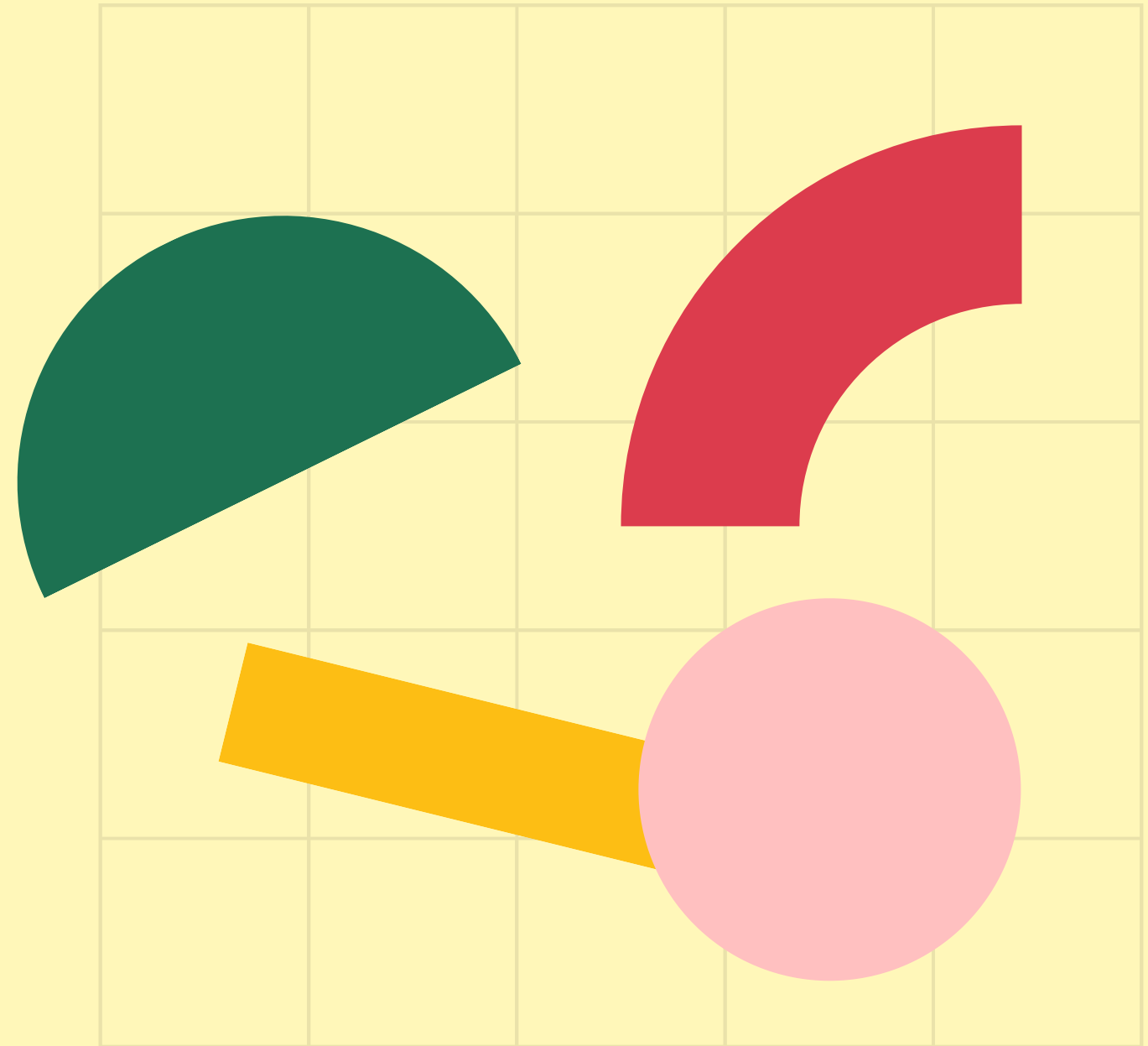
JavaScript Functions

Reusable Procedures

Unit Goals

what we'll cover

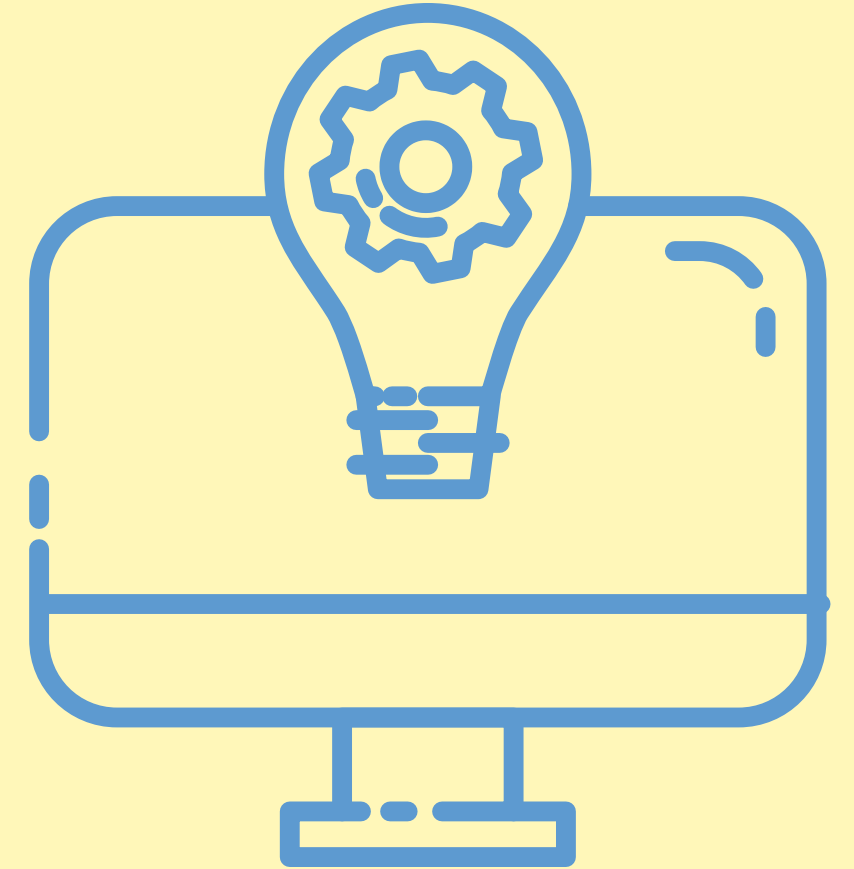
- what functions are
- function elements
 - define
 - run
 - arguments
 - inputs
 - return



FUNCTIONS

REUSABLE PROCEDURES

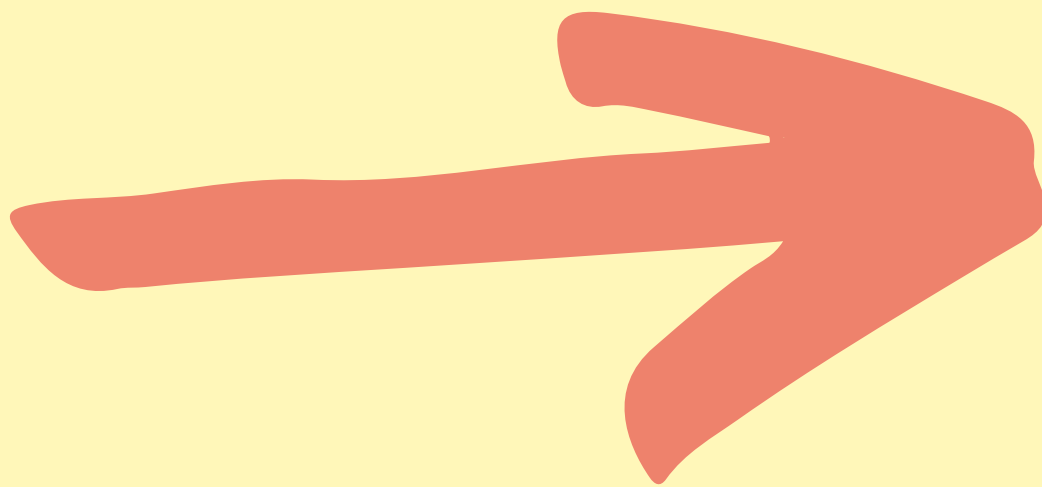
- Functions allow us to write reusable, modular code
- We define a "chunk" of code that we can then execute at a later point.
- We use them ALL THE TIME



FUNCTIONS

HAVE A 2 STEP PROCESS:

1) DEFINE




2) RUN



1) DEFINE

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

1) DEFINE



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

2) RUN

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

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

2) RUN



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

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

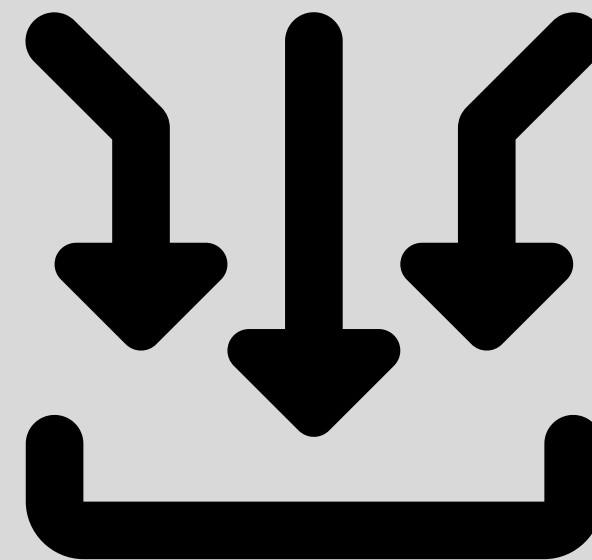

ARGUMENTS

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

ARGUMENTS

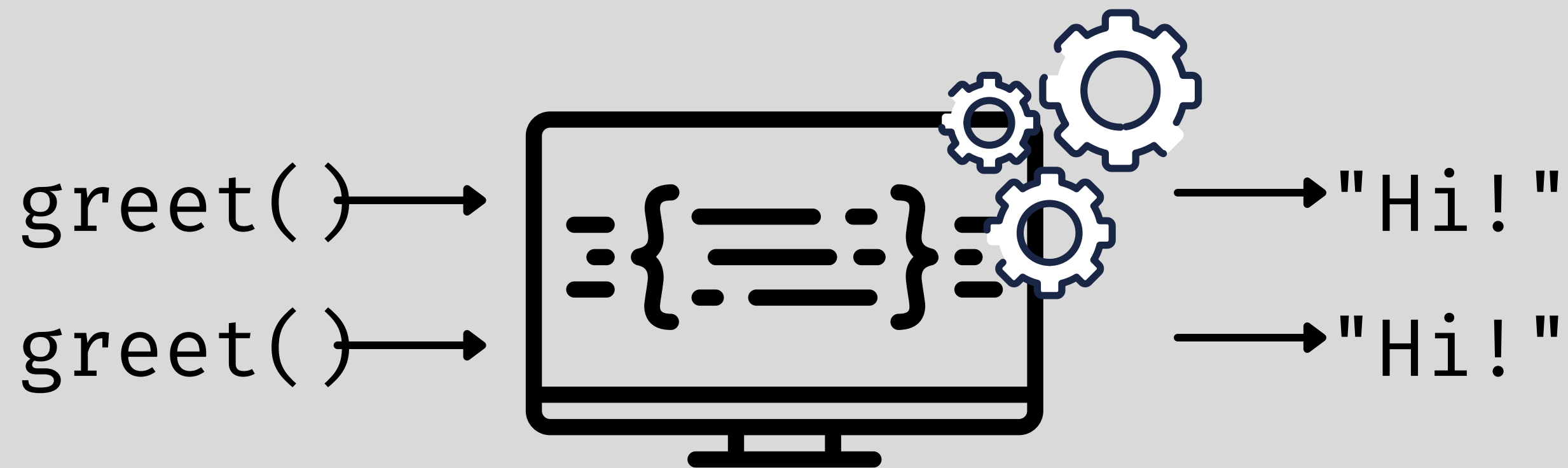
NO INPUTS

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



ARGUMENTS

NO INPUTS

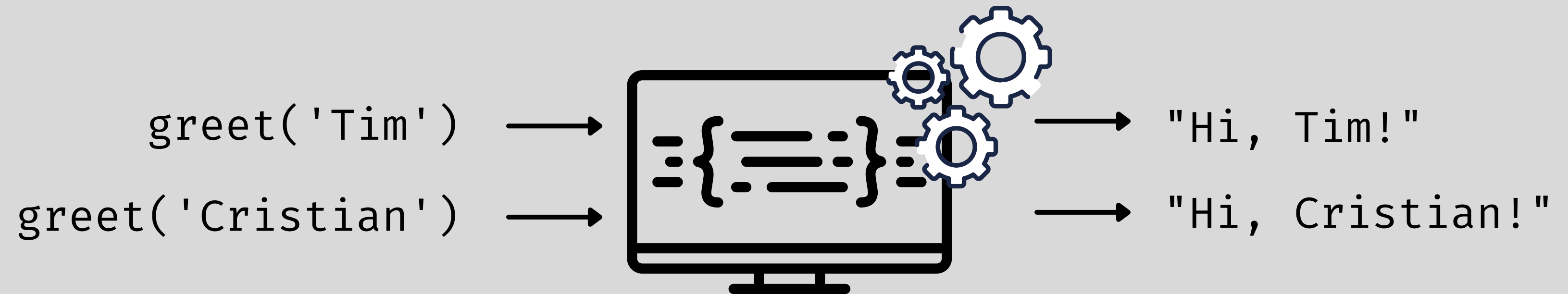


ARGUMENTS

INPUTS

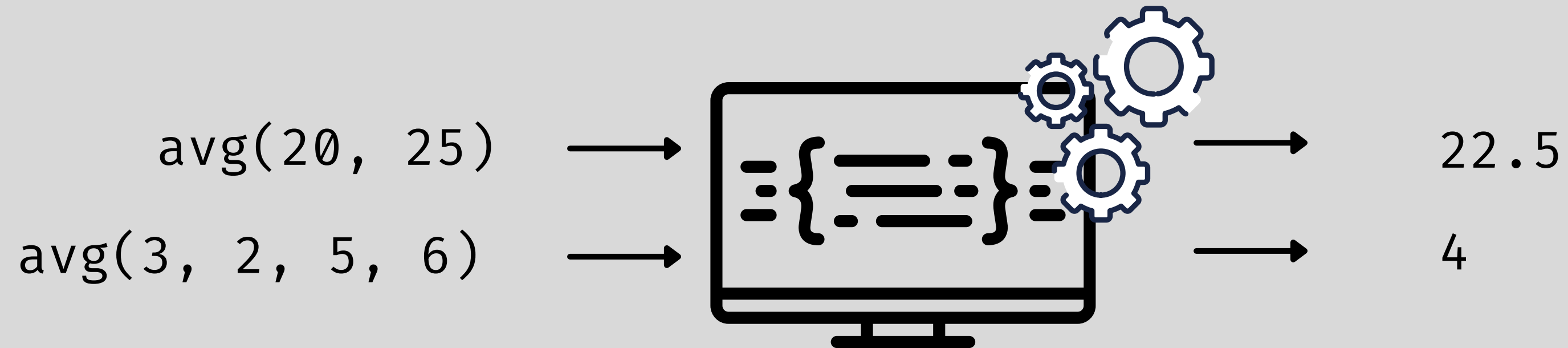
We can also give in inputs to the function and they will be used by the function. The inputs are the arguments.

ARGUMENTS



ARGUMENTS

ANOTHER EXAMPLE



ARGUMENTS

- NO ARGUMENTS



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

- ARGUMENTS



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

ARGUMENTS



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



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



"Hi, Arya!"



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



"Hi, Ned!"

ARGUMENTS

TWO 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

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

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.

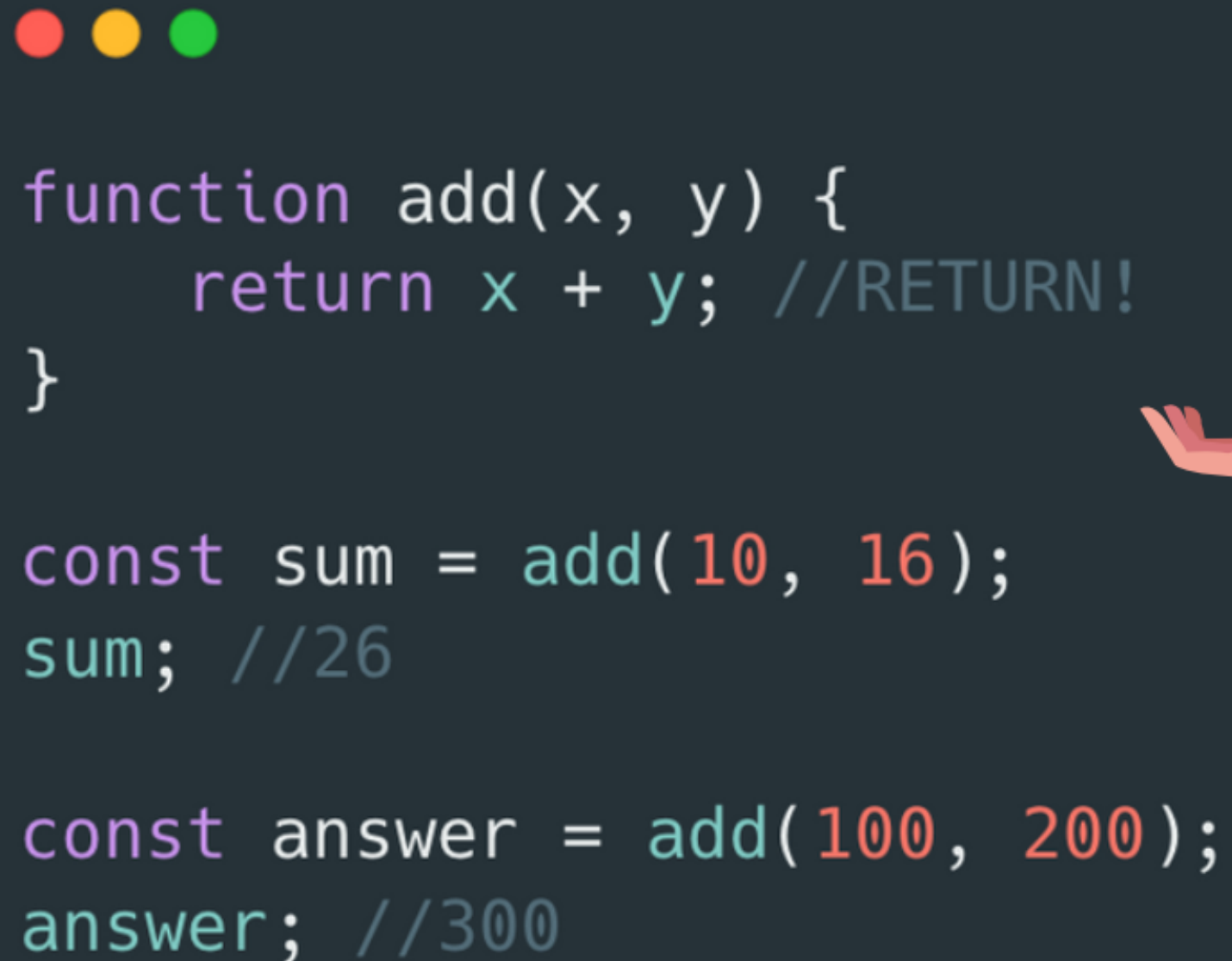


```
function add(x, y) {  
    console.log(x + y);  
}
```

```
const sum = add(10, 16);  
sum; //undefined
```

FIRST RETURN

Our functions print values out, but do NOT return anything.



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