Some commonly used definitions related to JavaScript

@saidali_ibn_zafar

27th May, 2023

Functional JS First Steps

JS

What is a Side Effect?

A side effect occurs in a program whenever we use an external code in our function—which, as a result, impacts the function's ability to perform its task.

What is an Impure Function?

An impure function is a function that contains one or more side effects.

What is an Pure Function?

A pure function is a function without any side effects.

```
1  let oldDigit = 5;
2
3  function addNumber(newValue) {
4    return oldDigit += newValue;
5  }
```

Side Effect

```
1  const myNames = ["Saidali", "Zikirillaev"];
2
3  function updateMyName(newName) {
4    myNames.push(newName);
5    return myNames;
6  }
```

Impure Function

```
1  function updateMyName(newName) {
2      const myNames = ["Saidali", "Zikirillaev"];
3      myNames[myNames.length] = newName;
4      return myNames;
5  }
```

Pure Function

What is a Currying inJS?

Currying in JavaScript is a process in functional programming in which you can transform a function with multiple arguments into a sequence of nesting functions.

```
// Normal function
function addition(x, y) {
  return x + y;
}
// Curried function
function addition(x) {
  return function(y) {
    return x + y;
  }
}
```

Currying always returns another function with only one argument until all of the arguments have been applied. So, we just keep calling the returned function until we've exhausted all the arguments and the final value gets returned.

What is a Partial Application?

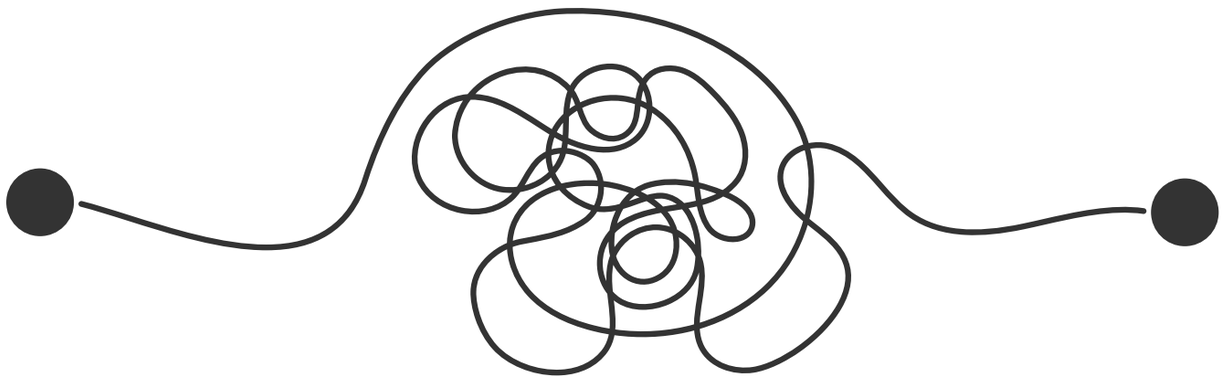
The process of applying a function to some of its arguments. The partially applied function gets returned for later use. In other words, a function that takes a function with multiple parameters and returns a function with fewer parameters.

```
function add(a, b, c) {
  return a + b + c;
}

const add5 = add.bind(null, 5); // Fixing the first argument to 5

console.log(add5(2, 3)); // Output: 10 (5 + 2 + 3)

```



What is a Function Composition?

an approach where the result of one function is passed on to the next function, which is passed to another until the final function is executed for the final result.

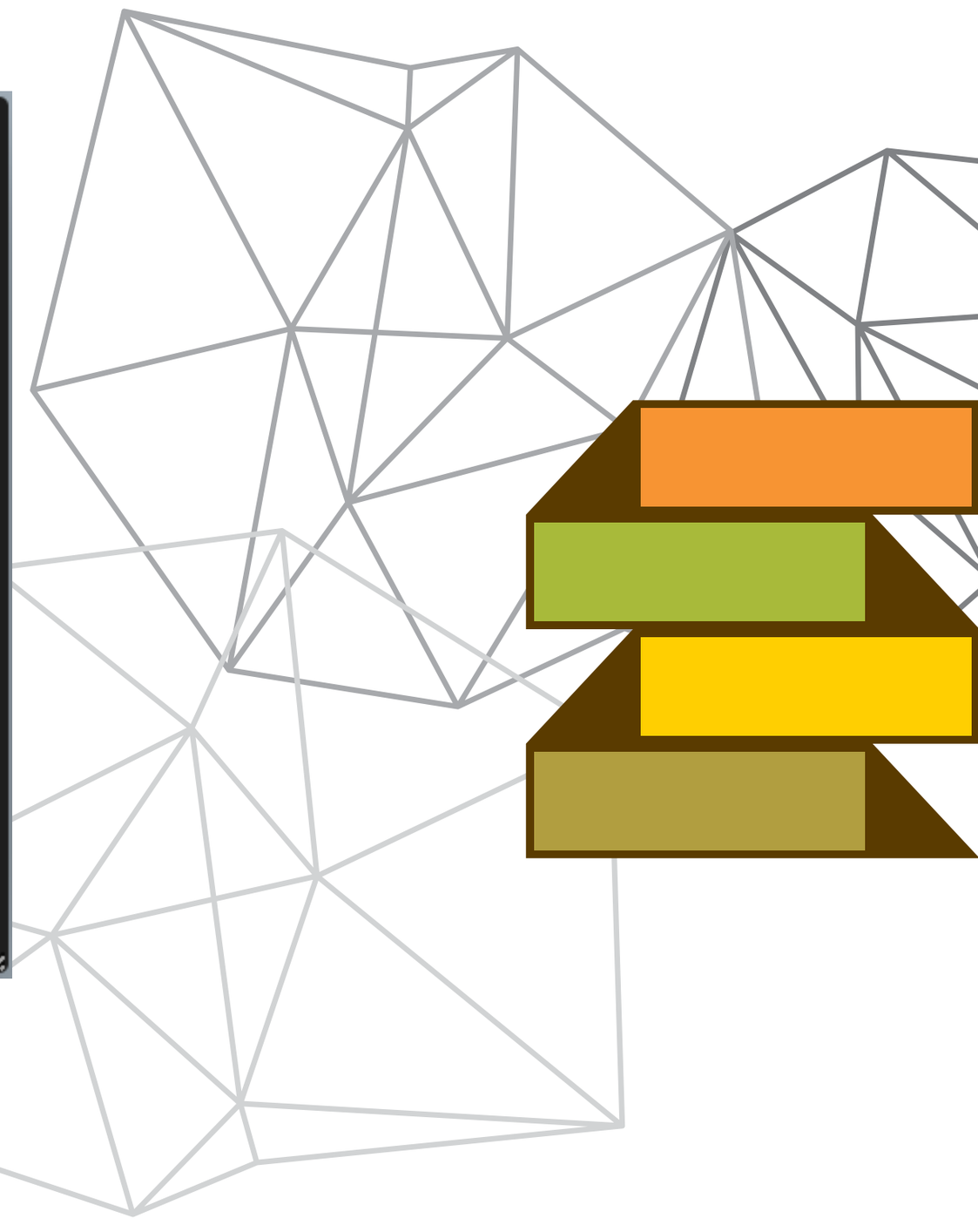
```
function add2(x) {
  return x + 2;
}

function multiplyBy3(x) {
  return x * 3;
}

function subtract1(x) {
  return x - 1;
}

// Function composition using higher-order functions
const finalResult = subtract1(multiplyBy3(add2(5)));
console.log(finalResult); // Output: 20

```



What is a Recursion?

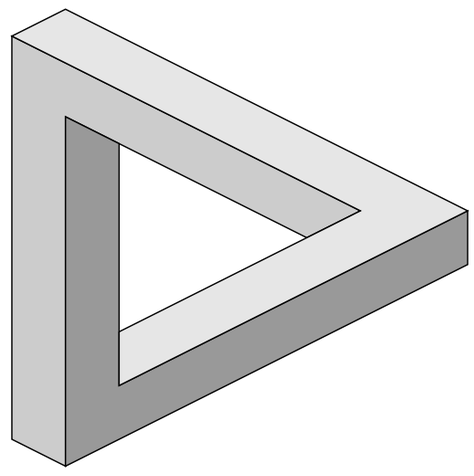
Recursion is a process of calling itself.

```
function sumUpTo(n) {
  // Base case: sum of 0 is 0
  if (n === 0) {
    return 0;
  }

  // Recursive call: sum of n is n plus the sum of (n - 1)
  return n + sumUpTo(n - 1);
}

// Calculate the sum of numbers up to 5
console.log(sumUpTo(5)); // Output: 15

```



What is a Iteration?

Iteration occurs when we want to execute code once for each item in a collection, usually elements in an array or properties of an object. Common looping methods are 'for' loops, 'while' loops and 'do... while' loops.

What is a Mutable and Immutable?

A **mutable** value is one that can be changed without creating an entirely new value.

An **immutable** value is one whose content cannot be changed without creating an entirely new value.

```
let numbers = [1, 2, 3, 4, 5];

function multiplyArrayByTwo(arr) {
  for (let i = 0; i < arr.length; i++) {
    arr[i] *= 2;
  }
}

multiplyArrayByTwo(numbers);
console.log(numbers); // Output: [2, 4, 6, 8, 10]

```

```
function addElementToArray(arr, element) {
  return [...arr, element];
}

let numbers = [1, 2, 3, 4, 5];
let newArray = addElementToArray(numbers, 6);

console.log(numbers);   // Output: [1, 2, 3, 4, 5]
console.log(newArray);  // Output: [1, 2, 3, 4, 5, 6]

```

saidali-ibn-zafar - Overview

Experienced front-end dev with a strong foundation in HTML, CSS, and JS. Excited to learn and build with ???. Passionate about creating user-friendly web app - saidali-ibn-zafar

GitHub