Advanced JS - 2

03 June 2022 20:07

Higher order function:-

→ HOF are functions that returns other functions as arguments, or return functions as result.

→ Arrays;-

- map
- filter
- reduce
- Sort

[1, 2, 3, 4]

[1, 4, 9, 16]

```
arr.sort((a,b) => b-a)
arr.map(  )
```

} transform array

```
function square(x)
{
    return (x*x);
}
```

```
arr = [1, 2, 3, 4]
arr.filter(function (item) {
    return item > 3;
});
```

```
const numbers = [1, 2, 3, 4, 5, 6]
```

find sum of all!

```
function reducer(acc, curr)
```

```
{
    return acc + curr;
}
console.log(numbers.reduce(reducer));
```

```
sum = sum + numbers[i]
```

(*) Composability :-

→ Composability is a process of combining multiple simple functions to build a more complicated one.

```
const add = (a, b) => a + b;
const mult = (a, b) => a * b;
add(2, mult(3, 5));
// 17
```

---

2 + 3 * 5 + 4 * 7 + 5 * 3

add(2, mul(3,5), mul(4,7), mul(5,3))

add(2, 15, 28, 15)

---

2 + 3 * 5 | 7

add(2 divide(mul(3 5) 7)

```
divide(a,b)
{
    a/b
```

add(2, ...)

-> inner fn => exec first
↓
outer fn

Ordering

Currying

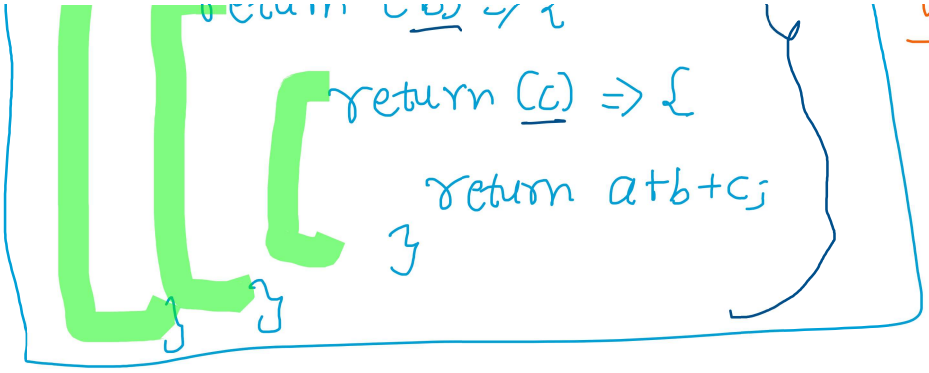
↳ Currying is a technique of evaluating function with multiple arguments, into a sequence of functions with single arguments.

↳ When a function, instead of taking all arguments at one time, takes the first one & return a new fn that takes a second one & retns a new fn which take third one ---- so on ----

```
const add = (a, b, c) => {
    return a+b+c
}
```

```
function func(a) {
    return (b) => {
```

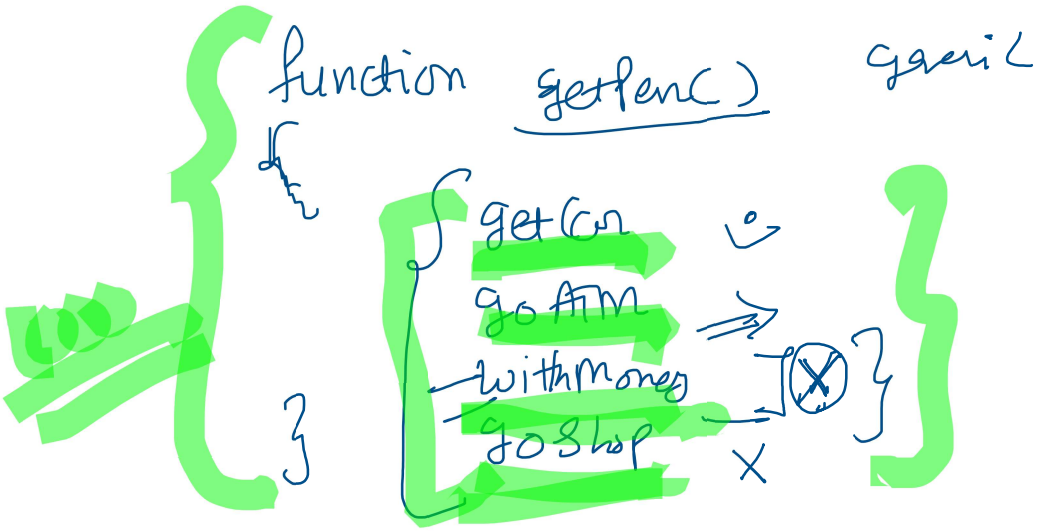
Syntax:



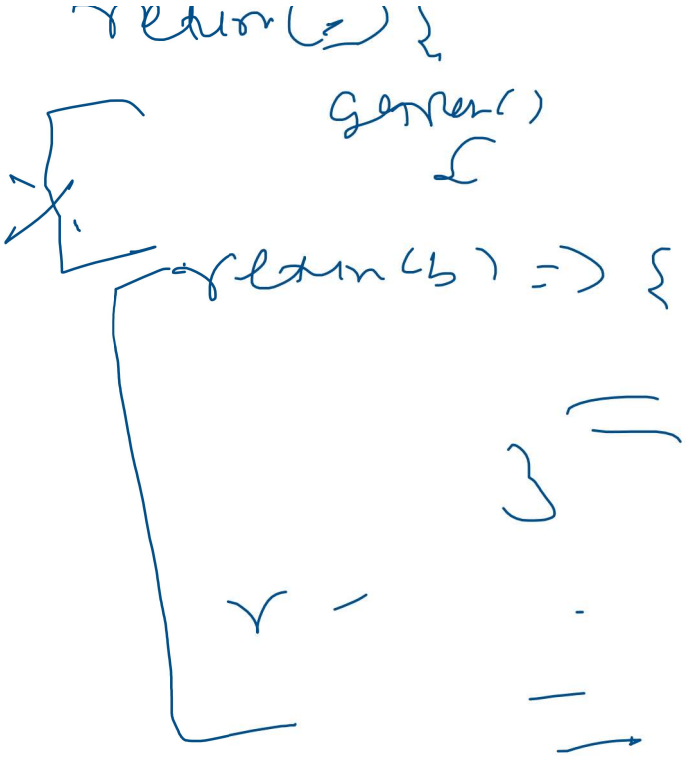
{ 2+3+undefined }

```
const add = (a,b,c) =>{
    return a+b+c;
}

console.log(add(2,3,4))
debugger;
//######Currying######//
const curryAdd = (sum) => {
  return (a) => {
    return (b) => {
      return (c) =>{
        return sum(a,b,c);
      }
    }
  }
}

const addition = curryAdd(add)
console.log(addition(1)(2)(3));
```

add(2,3)



function getPen() {



```
const add = (a,b,c) =>{
    return a+b+c;
}
console.log(add(2,3,4))

//######Currying######//
const curryAdd = (sum) => {
    debugger;
  return (a) => {
      console.log("inside a",a)
    return (b) => {
        console.log("inside b",a,b)
      return (c) =>{
          console.log("inside c",a,b,c)
        return sum(a,b,c);
      }
    }
  }
}

const addition = curryAdd(add);
console.log(addition(1)(2)(3));
```

④ Anonymous function :-

→ Anonymous function in javascript are those functions without any identifier or na to refer it.

⑤ Arrow fn

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

```
const res = (a,b) => a+b;
```

* function - A first class citizen;

1. We can store fn in variables
2. Pass fn as an arguments
3. return fn as an result

```
const add = (a,b) => { return a+b; }

const pass = (add) => {
    return add(1,2);
}

const func = (a,b) {
    return () => {
        console.log(a+b+2);
    }
}
```

// IIFE

Immediately invoked function expression!.

after define -> call

(function() {