6th Class

polyfills - 2

- filter
- reduce

→ Arrays

5 mins

- call
- Apply
- Bind

→ functions

⇒ oldArr → [ele1, ele2, ele3]

* callback(ele1) → true → ✗
  → false → ✓

* callback(ele2) → true → ✓
  → false → ✗

* callback(ele3) → true

* [ele2, ele3]

oldArr.filter(callback);

```
Array.prototype.filter = function (callbackfn) {
    const result = [];
    for(let i=0; i<this.length; i++){
        if (callback(this[i], i)) {
            result.push(this[i]);
        }
    return result;
}
```

⇒ reduce

↳ collection of something

↳ it reduces collection to single entity

[{}, {}, {}] ⇒ {}

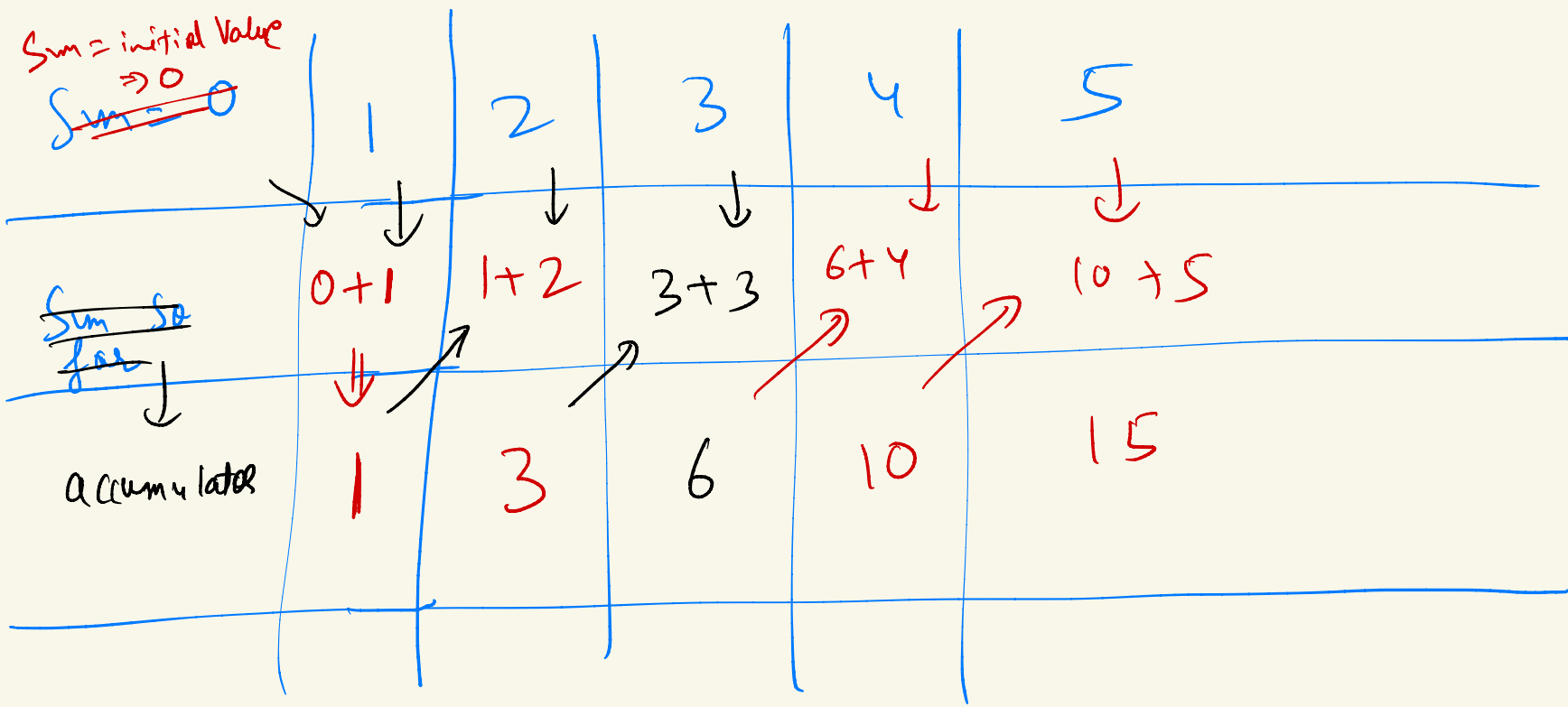
["strig", "string"] ⇒ "string"

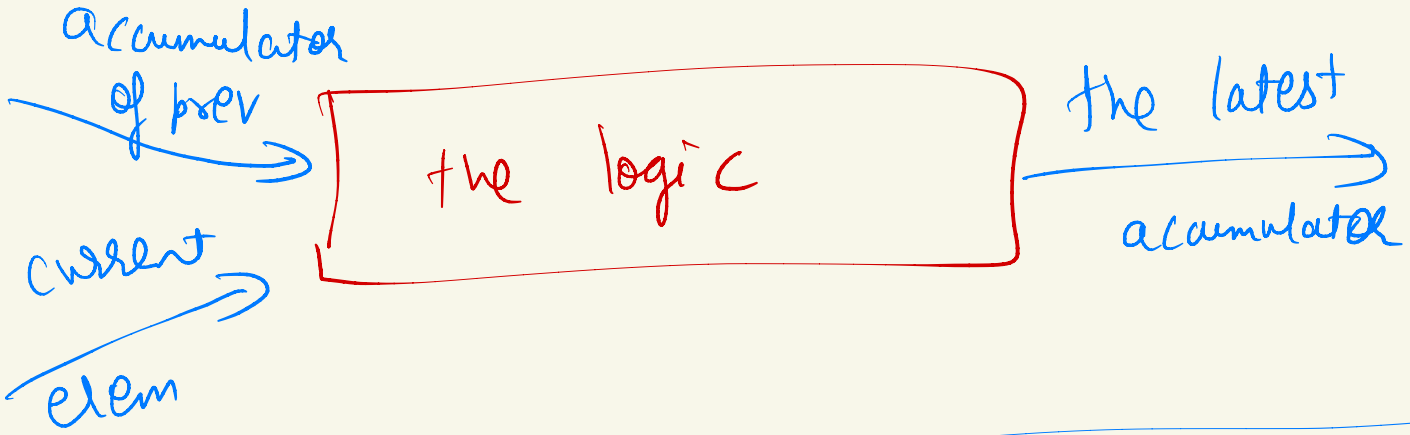
[1, 2, 3, 4, 5]

* Sum = 0

```
for (let i=0; i< size; i++) {
    Sum = Sum + arr[i];
}
return Sum;
```

accumulator

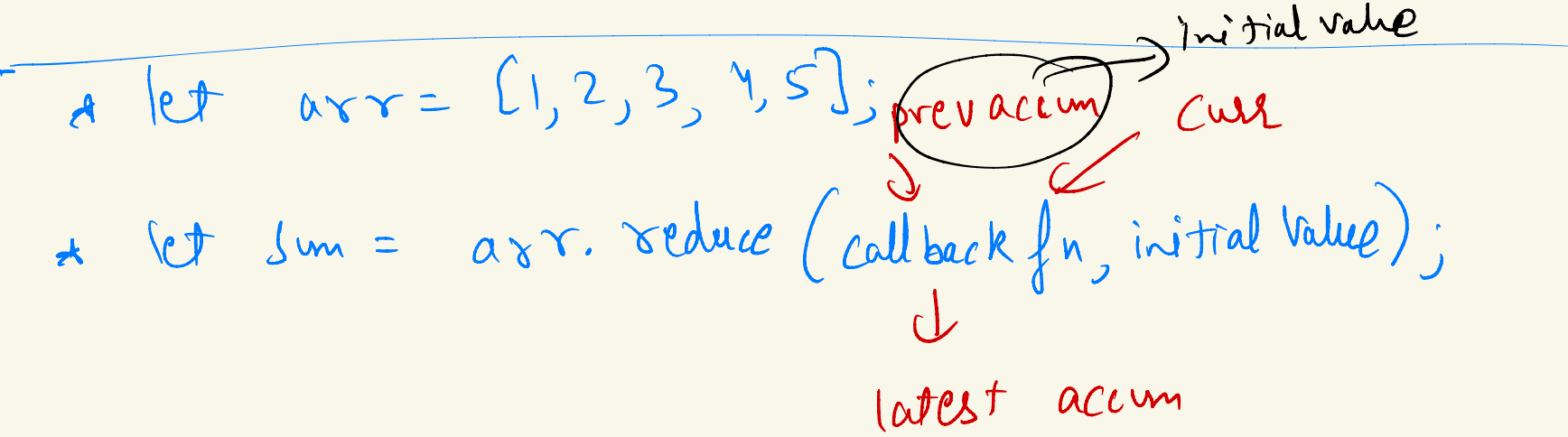




the latest accumulator

reduce → dividing → sum into

initial value & accumulator



* let arr = [1, 2, 3, 4, 5];

* let sum = arr.reduce(

```
let arr = [1, 2, 3]

* arr.reduce((accumulator, item) => {
      // whatever logic I write here
      // it will update my accumulator
      return accumulator + item;
  }, 0);
```

undefined

function (callback, ~~initialValue~~) {

let acc = initialValue → undefined

for (let i = 0; i < lenght; i++) {

acc = this[0] ⇒ for

acc = callback(acc, item);

}

}

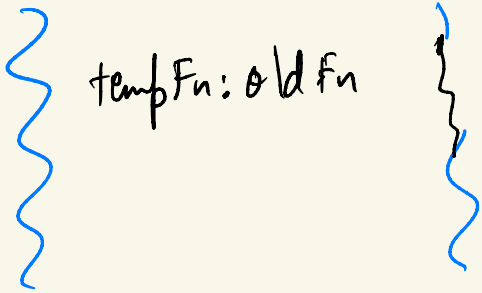
[1, 2, 3]

acc ⇒ 1

initival Value ⇒ undefined

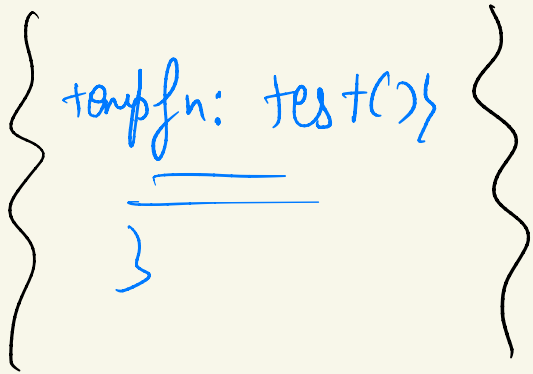
* call , apply

* oldFn.call(obj, arg1, arg2, arg3);



* obj

* obj.tempFn = oldFn
* obj.tempFn()
* delete obj.tempFn



function test() {

}

obj

* obj.tempfn = this;

* obj.tempfn();

test.call()

↓

any this will be having access fn defi