

12 August 2023 20:05

```
function add(value){
  if(typeof value=='undefined') return 0;
  function inner(nextvalue){
    if(typeof nextvalue=="undefined"){
      return value;
    }
    value+=nextvalue;
    return inner;
  }
  return inner;
}
```

Handwritten notes and corrections:

- ① circled around the first parameter `value`.
- ② circled around the `return 0;` line.
- ③ circled around the `nextvalue` parameter in the `inner` function.
- Arrows pointing to the `return value;` line in the `inner` function and the `return inner;` line in the `add` function.
- A large arrow pointing from the `return inner;` line to a circled note: `Value = 15`.
- Handwritten calculations on the right:  $10 + 5 = 15$  and  $15 + 1 = 16$ .

inner(2)(3)(4)(5)(6)( ) ;  
inner(3)(4)(5)(6)( )  
inner(4)(5)(6)( )  
inner(5)(6)( )  
inner(6)( )  
inner( )

10 20 un-definer  
↓ ↓ ↓  
GEC

21

Handwritten diagram illustrating the execution of a JavaScript function call. The diagram shows the function definition, the call, and the state of the call stack and environment.

**Function Definition:**

```
function update(a, b, c) {
  c = a + b;
}
let a = 10, b = 20, c;
update(a, b, c);
console.log(c);
```

**Call Stack:**

- update: f {
  - a = 10
  - b = 20
  - c = undefined

**Environment:**

- a = 10
- b = 20
- c = undefined

**Execution Flow:**

- The function `update` is called with arguments `10`, `20`, and `undefined`.
- The function body executes `c = a + b`, where `a` is `10` and `b` is `20`.
- The result of the function call is `30`.

update ↓

mc	ce