function outer(num) {

let myNum = num \* 2;

return () => { console.log(`

original parameter: ${num}

myNum: ${++myNum}`);

}

}

let firstCall=outer(1);

let secondCall=outer(100);

firstCall();

secondCall();

Memory snapshots

let firstCall=outer(1);

|  |  |
| --- | --- |
| heap | stack |
| function outer(num 1) {  let myNum =2;  return () => {}  } | firstCall |

let secondCall=outer(100);

|  |  |
| --- | --- |
| heap | stack |
| function outer(num 100) {  let myNum =200;  return () => {}  }  function outer(num 1) {  let myNum =2;  return () => {}  } | firstCall  secondCall |

firstCall();

secondCall();

|  |  |
| --- | --- |
| heap | stack |
| function outer(num 100) {  let myNum =201;  return () => {}  }  function outer(num 1) {  let myNum =3;  return () => {}  } | firstCall  secondCall |

let firstCall=null;

|  |  |
| --- | --- |
| heap | stack |
| Function without a pointer  function outer(num 100) {  let myNum =201;  return () => {}  }  function outer(num 1) {  let myNum =3;  return () => {}  } | firstCall  null  secondCall |

After GC action:

|  |  |
| --- | --- |
| heap | stack |
| function outer(num 100) {  let myNum =201;  return () => {}  } | firstCall  null  secondCall |