# **Local Scope**

R2.2/2.3



```
//this code simply tests the effects of local
     scope
var counter=0;
var ans2=[];//global variable
function f(x) {
var ans=[];//local variable of f(x)
function add(x){
     ans[counter]=x;//accessing the local
     variable fo f(x)
     ans2[counter]=x;//accessing the global variable
     counter++;
     return ans;
return add(x);
console.log(f(3));//log the example
console.log(ans2);//log the global variable
```

stack memory Main Counter:0 Ans2:[]



Console.log

stack memory

Counter:0
Ans2:[]

```
//this code simply tests the effects of local
      scope
var counter=0;
var ans2=[];//global variable
function f(x) {
var ans=[];//local variable of f(x)
function add(x){
     ans[counter]=x;//accessing the local
      variable fo f(x)
     \begin{array}{ll} {\tt ans2[counter]=x;//accessing \ the \ global} \\ {\tt variable} \end{array}
      counter++;
      return ans;
return add(x);
console.log(f(3));//log the example
console.log(ans2);//log the global variable
```

F(3)

Console.log

stack memory Main Counter:0 Ans2:[]

```
var counter=0;
var ans2=[];//global variable
function f(x) {
var ans=[];//local variable of f(x)
function add(x){
     ans[counter]=x;//accessing the local
     variable fo f(x)
     ans2[counter]=x;//accessing the global variable
     counter++;
     return ans;
return add(x);
console.log(f(3));//log the example
console.log(ans2);//log the global variable
```

//this code simply tests the effects of local

scope



F(3) Ans:[]

Console.log

stack memory Main Counter:0 Ans2:[]

```
al variable
var ans2=[];/
function f(
var ans=[]\frac{9}{1000}/local variable of f(x)
function add(x){
     ans[counter]=x;//accessing the local
     variable fo f(x)
     ans2[counter]=x;//accessing the global
variable
     counter++;
     return ans;
return add(x);
console.log(f(3));//log the example
console.log(ans2);//log the global variable
```

//this code simply tests the effects of local

scope

var counter=0;

Add(x)

F(3) Ans:[]

Console.log

stack memory Main Counter:0 Ans2:[]

```
var counter=0;
var ans2=[];//global variable
function f(x) {
var ans=[];//local variable of f(x)
function add(x){
     ans[counter]=x;//accessing the local
     variable fo f(x)
     ans2[counter]=x;//accessing the global
variable
     counter++;
     return ans
return add(x)
console.log(f(3));//log the example
console.log(ans2);//log the global variable
```

//this code simply tests the effects of local

scope



stack

memory

Add(x) Ans:[3] Console.log Counter:0 Ans2:[]

```
//this code simply tests the effects of local
      scope
var counter=0;
var ans2=[];//global variable
function f(x) {
                           ole of f(x)
var ans=[];//local
function add(x){
     ans[counter]=x;//accessing the local
      variable for f(x)
     \begin{array}{ll} {\tt ans2[counter]=x;//accessing \ the \ global} \\ {\tt variable} \end{array}
      counter++;
      return ans;
return add(x);
console.log(f(3));//log the example
console.log(ans2);//log the global variable
```

Add(x)

F(3) Ans:[3]

Console.log

stack memory Main Counter:0 Ans2:[3]

```
//this code simply tests the effects of local
     scope
var counter=0;
var ans2=[];//global variable
function f(x) {
var ans=[];//local variable of f(x)
function add(x){
     ans[counter]=x;/
variable for f(/
                              ing the local
     ans2[counter] x;/X ccessing the global variable
     counter++;
     return ans;
return add(x);
console.log(f(3));//log the example
console.log(ans2);//log the global variable
```

Add(x)

F(3) Ans:[3]

Console.log

stack memory

```
Main
Counter:1
Ans2:[3]
```

```
//this code simply tests the effects of local
     scope
var counter=0;
var ans2=[];//global variable
function f(x) {
var ans=[];//local variable of f(x)
function add(x){
     ans[counter]=x;//accessing the local
     variable for f(x)
     ans2[counter]=x
variable
                          essing the global
     counter++;
     return ans;
return add(x);
console.log(f(3));//log the example
console.log(ans2);//log the global variable
```

Add(x)

F(3) Ans:[3

Console.log

stack memory Main Counter:1 Ans2:[3]

```
var ans2=[];//global variable
function f(x) {
var ans=[];//local variable of f(x)
function add(x){
      ans[counter]=x;//accessing the local
      variable for f(x)
      \begin{array}{ll} {\tt ans2[counter]=x;//accessing \ the \ global} \\ {\tt variable} \end{array}
      counter++;
      return ans
return add(x);
console.log(f(3));//log the example
console.log(ans2);//log the global variable
```

//this code simply tests the effects of local

scope

var counter=0;



F(3) Ans:[3]

Console.log

stack memory Main Counter:1 Ans2:[3]

```
← to
```

```
//this code simply tests the effects of local
     scope
var counter=0;
var ans2=[];//global variable
function f(x) {
var ans=[];//local variable of f(x)
function add(x){
     ans[counter]=x;//accessing the local
     variable for f(x)
     \begin{tabular}{ll} ans 2 [counter] = &x; // accessing the global variable \end{tabular}
     counter++;
     return ans;
return add(x);
console.log(f(3));//log the example
console.log(ans2);//log the global variable
```

Console.log

stack memory Main Counter:1 Ans2:[3]

**←** to

```
//this code simply tests the effects of local
     scope
var counter=0;
var ans2=[];//global variable
function f(x) {
var ans=[];//local variable of f(x)
function add(x){
     ans[counter]=x;//accessing the local
     variable for f(x)
     ans2[counter]=x;//accessing the global variable
     counter++;
     return ans;
return add(x);
console.\log(f(3));//log the example
console.log(ans2);//log the global variable
```

Console.log

stack memory Main Counter:1 Ans2:[3]

```
← to
```

```
//this code simply tests the effects of local
     scope
var counter=0;
var ans2=[];//global variable
function f(x) {
var ans=[];//local variable of f(x)
function add(x){
     ans[counter]=x;//accessing the local
     variable for f(x)
     ans2[counter]=x;//accessing the global variable
     counter++;
     return ans;
return add(x);
console.log(f(3));
                          he example
console.log(ans2);//log the global variable
```

stack memory Main Counter:1 Ans2:[3]

**←** to

```
//this code simply tests the effects of local
     scope
var counter=0;
var ans2=[];//global variable
function f(x) {
var ans=[];//local variable of f(x)
function add(x){
     ans[counter]=x;//accessing the local
     variable for f(x)
     ans2[counter]=x;//accessing the global variable
     counter++;
     return ans;
return add(x);
console.log(f(3));//log the example
console.log(ans2);//log the global variable
```

## Conclusion/R2.3

- No modifications are required for my example to work, it's a fairly simple one.
- I would suggest a better way to keep track of a variable through a "reference" notation, so that we only need to change the base case.