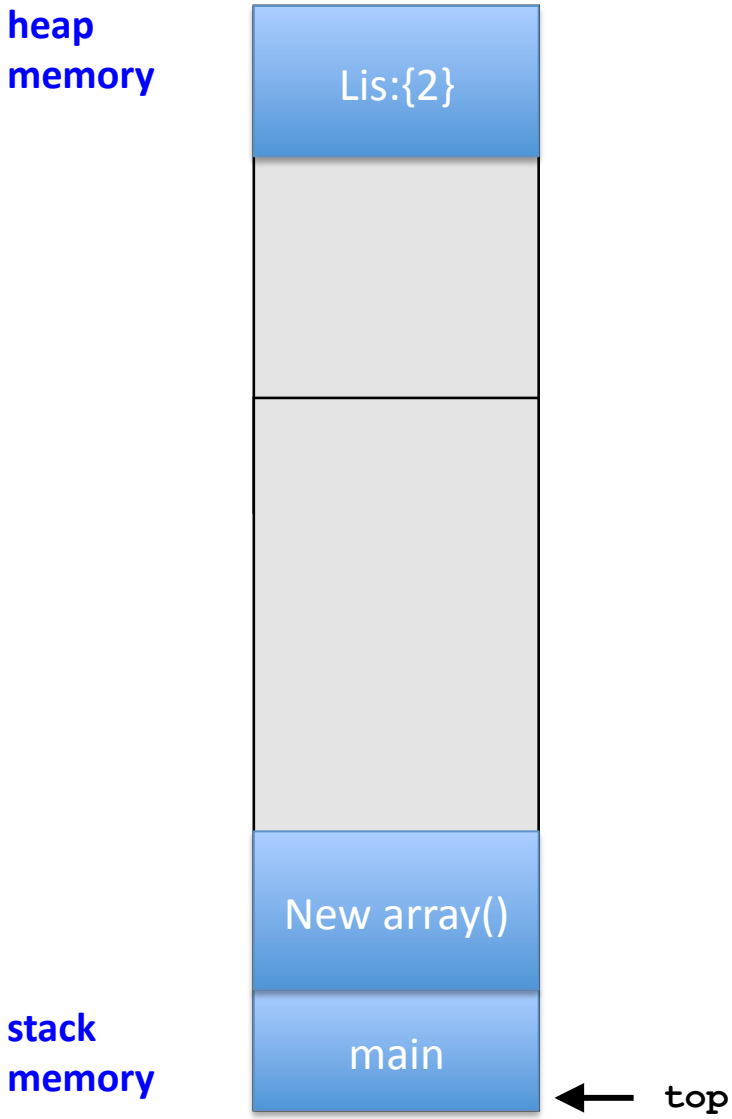
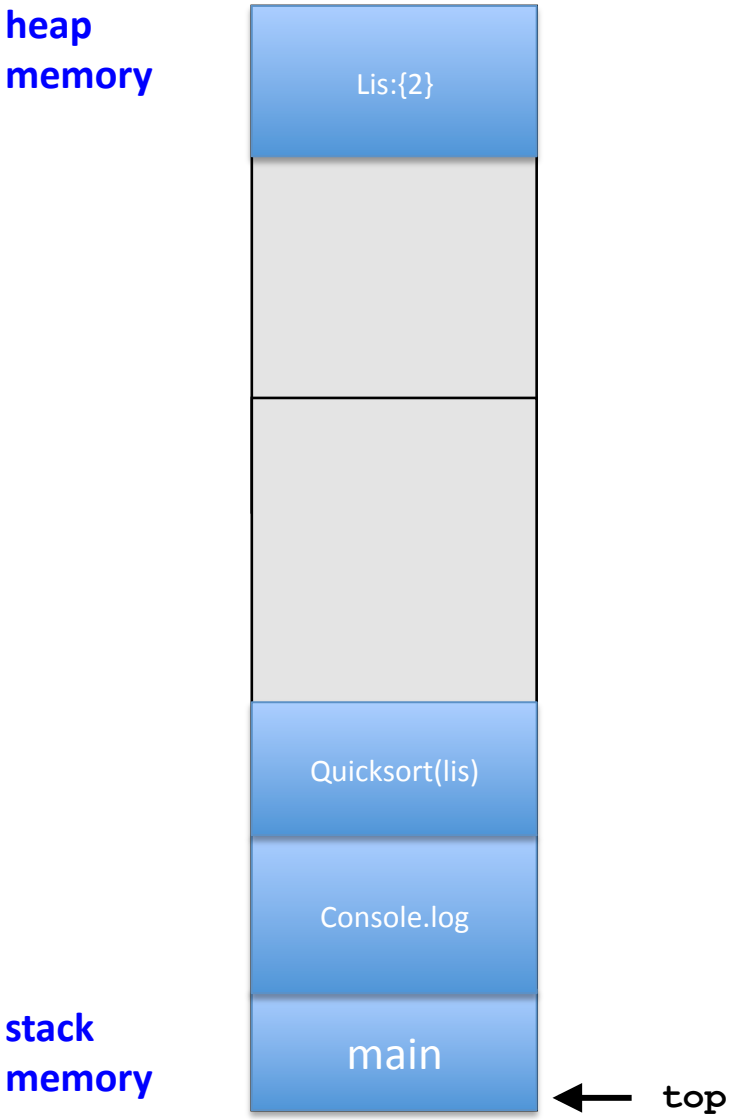
Quicksort

R1.2/R1.3



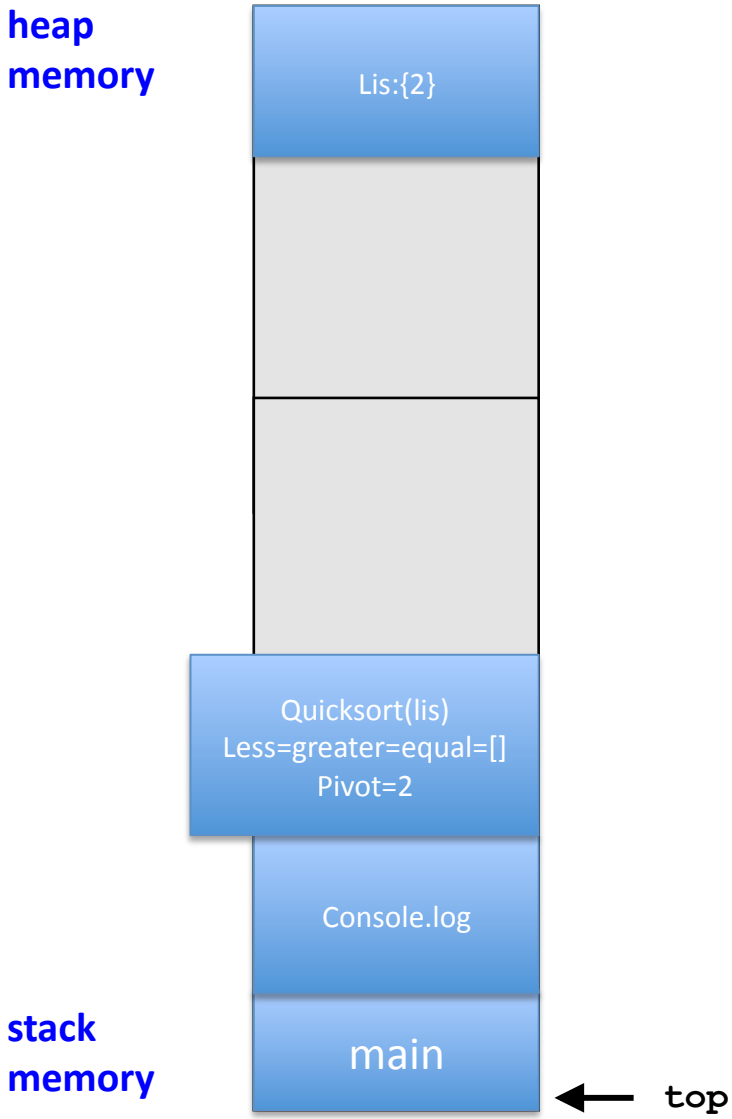

```
Function quicksort(array){
If(array.length<=0) return[]
   var less=[];
   var greater=[];
   var equal=[]
   var pivot=array[0];
 for(var i=0;i<array.length;i+
   +)
 if(pivot>array[i])
      less.push(array[i]);
 if(pivot==array[i])
      equal.push(array[i]);
 if(pivot<array[i])
     greater.push(array[i]);
}
Return quicksort(less).concat
   (equal,quicksort(greater));
}
Var lis=new array(2);

Console.log(quicksort(lis));
```



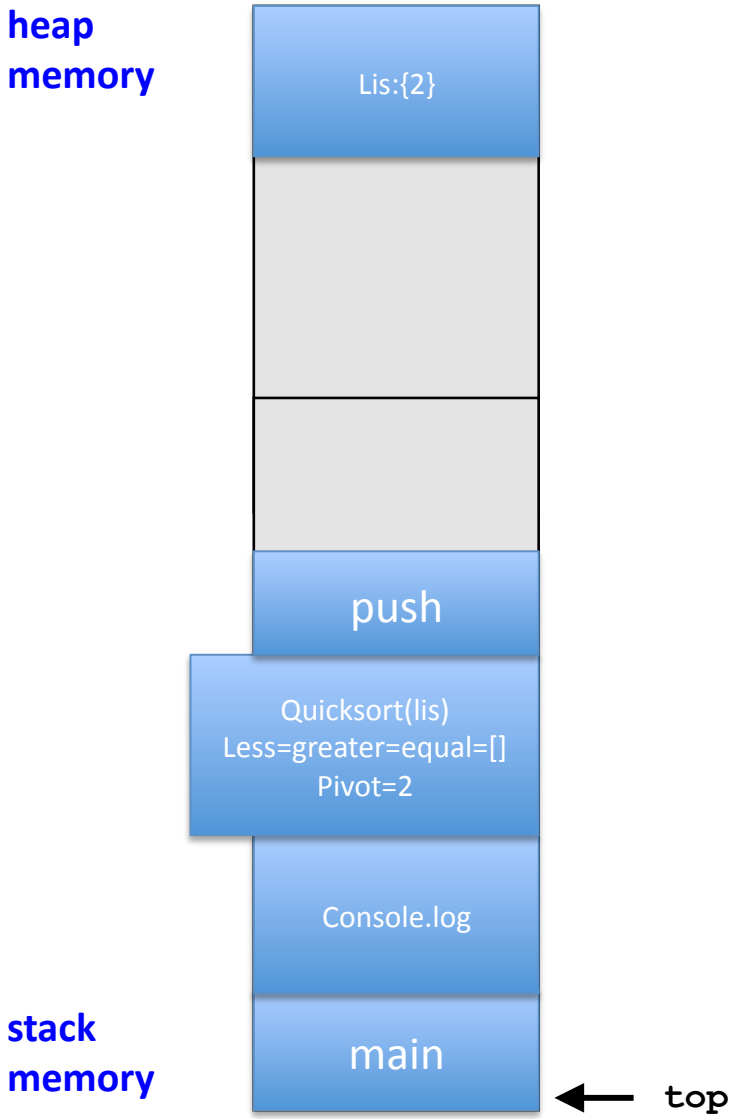

```
Function quicksort(array){
If(array.length<=0) return[]
   var less=[];
   var greater=[];
   var equal=[]
   var pivot=array[0];
 for(var i=0;i<array.length;i+
   +)
 if(pivot>array[i])
      less.push(array[i]);
 if(pivot==array[i])
      equal.push(array[i]);
 if(pivot<array[i])
     greater.push(array[i]);
}
Return quicksort(less).concat
   (equal,quicksort(greater));
}
Var lis=new array(2);

Console.log(quicksort(lis));
```



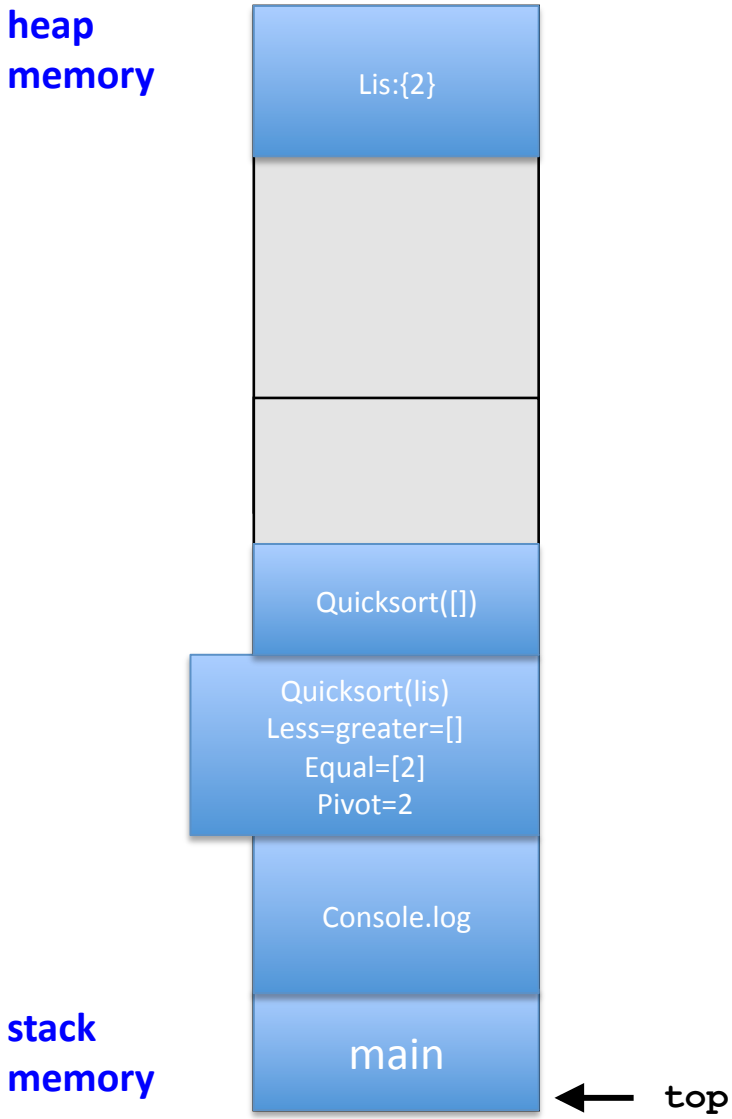

```
Function quicksort(array){
If(array.length<=0) return[]
   var less=[];
   var greater=[];
   var equal=[]
   var pivot=array[0];
 for(var i=0;i<array.length;i+
   +)
 if(pivot>array[i])
      less.push(array[i]);
 if(pivot==array[i])
      equal.push(array[i]);
 if(pivot<array[i])
     greater.push(array[i]);
}
Return quicksort(less).concat
   (equal,quicksort(greater));
}
Var lis=new array(2);

Console.log(quicksort(lis));
```



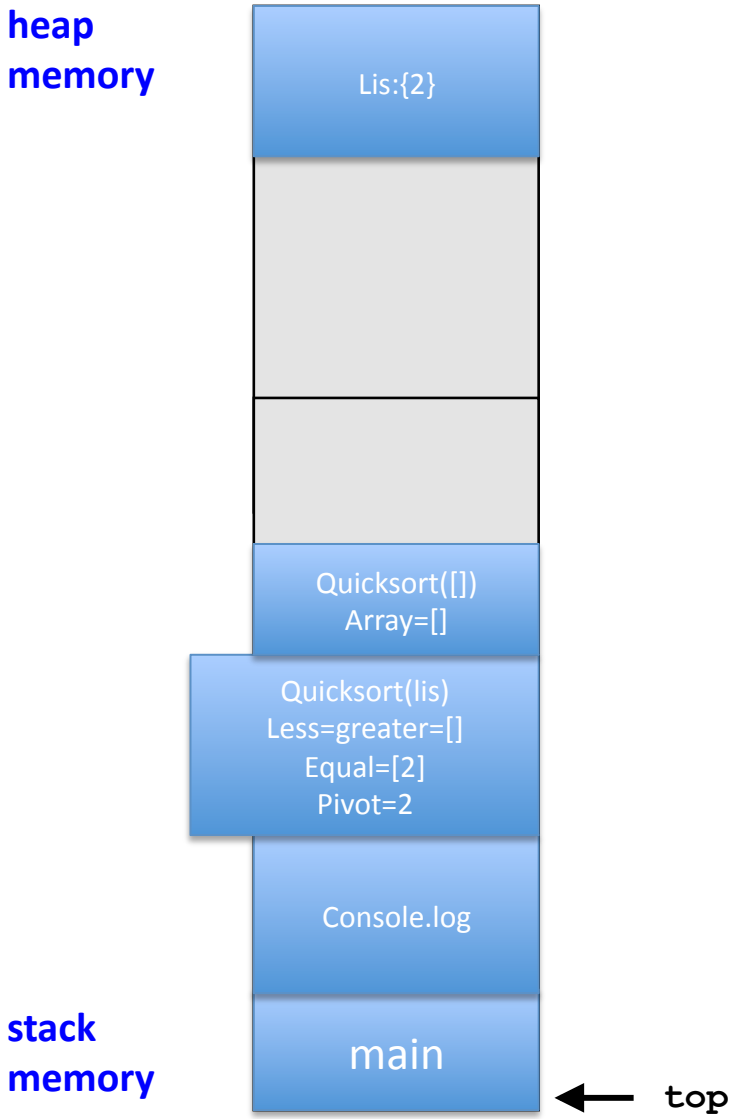

```
Function quicksort(array){
If(array.length<=0) return[]
   var less=[];
   var greater=[];
   var equal=[]
   var pivot=array[0];
 for(var i=0;i<array.length;i++
   +)
 if(pivot>array[i])
      less.push(array[i]);
 if(pivot==array[i])
      equal.push(array[i]);
 if(pivot<array[i])
     greater.push(array[i]);
}
Return quicksort(less).concat
   (equal,quicksort(greater));
}
Var lis=new array(2);

Console.log(quicksort(lis));
```



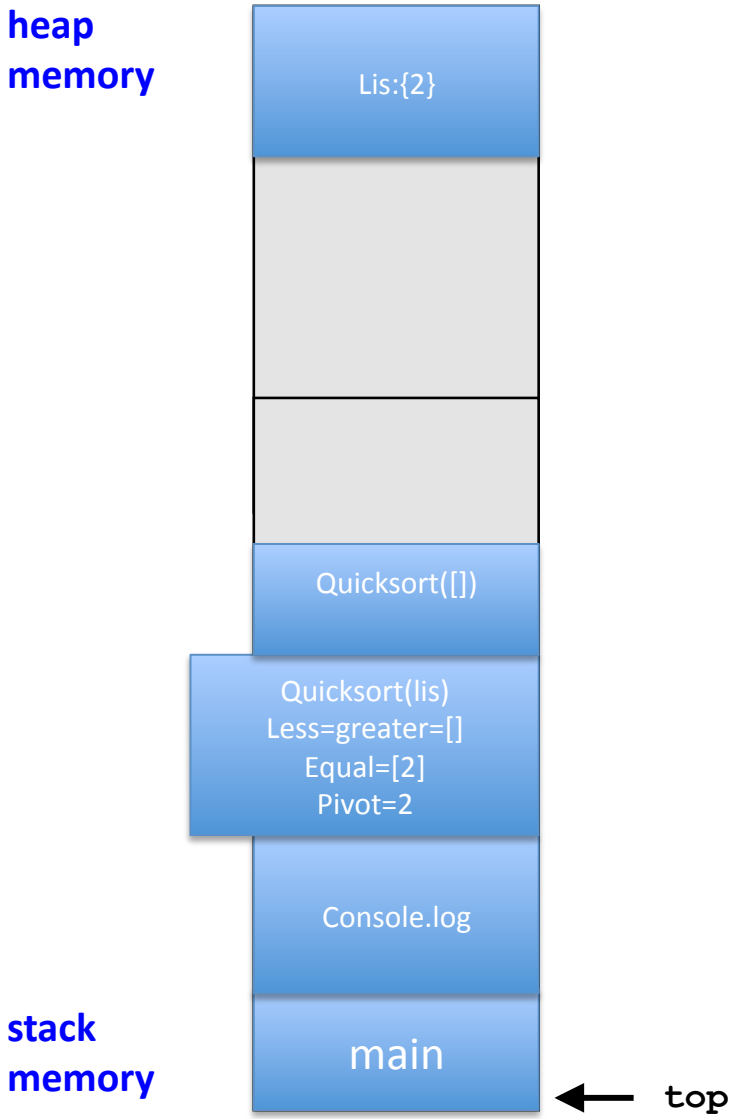

```
Function quicksort(array){
If(array.length<=0) return[]
   var less=[];
   var greater=[];
   var equal=[]
   var pivot=array[0];
 for(var i=0;i<array.length;i+
   +)
 if(pivot>array[i])
      less.push(array[i]);
 if(pivot==array[i])
      equal.push(array[i]);
 if(pivot<array[i])
     greater.push(array[i]);
}
Return quicksort(less).concat
   (equal,quicksort(greater));
}
Var lis=new array(2);

Console.log(quicksort(lis));
```



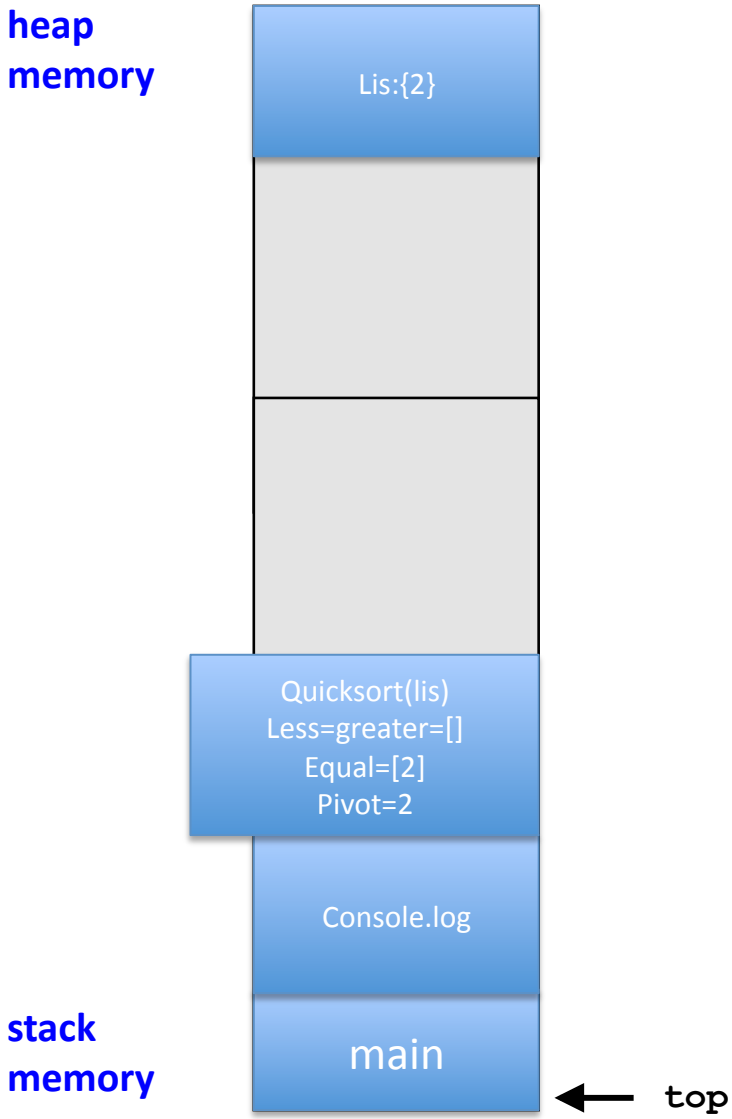

```
Function quicksort(array){
If(array.length<=0) return[]
   var less=[];
   var greater=[];
   var equal=[]
   var pivot=array[0];
 for(var i=0;i<array.length;i+
   +)
 if(pivot>array[i])
      less.push(array[i]);
 if(pivot==array[i])
      equal.push(array[i]);
 if(pivot<array[i])
     greater.push(array[i]);
}
Return quicksort(less).concat
   (equal,quicksort(greater));
}
Var lis=new array(2);

Console.log(quicksort(lis));
```



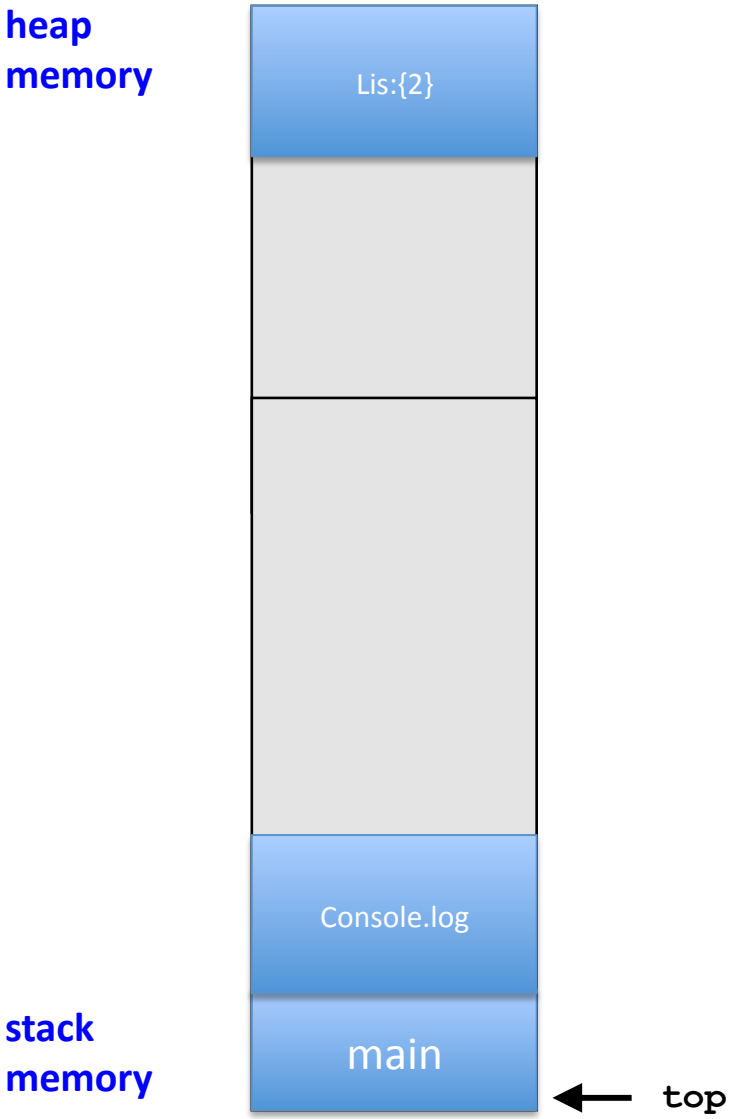

```
Function quicksort(array){
If(array.length<=0) return[]
   var less=[];
   var greater=[];
   var equal=[]
   var pivot=array[0];
 for(var i=0;i<array.length;i+
   +)
 if(pivot>array[i])
      less.push(array[i]);
 if(pivot==array[i])
      equal.push(array[i]);
 if(pivot<array[i])
     greater.push(array[i]);
}
Return quicksort(less).concat
   (equal,quicksort(greater));
}
Var lis=new array(2);

Console.log(quicksort(lis));
```



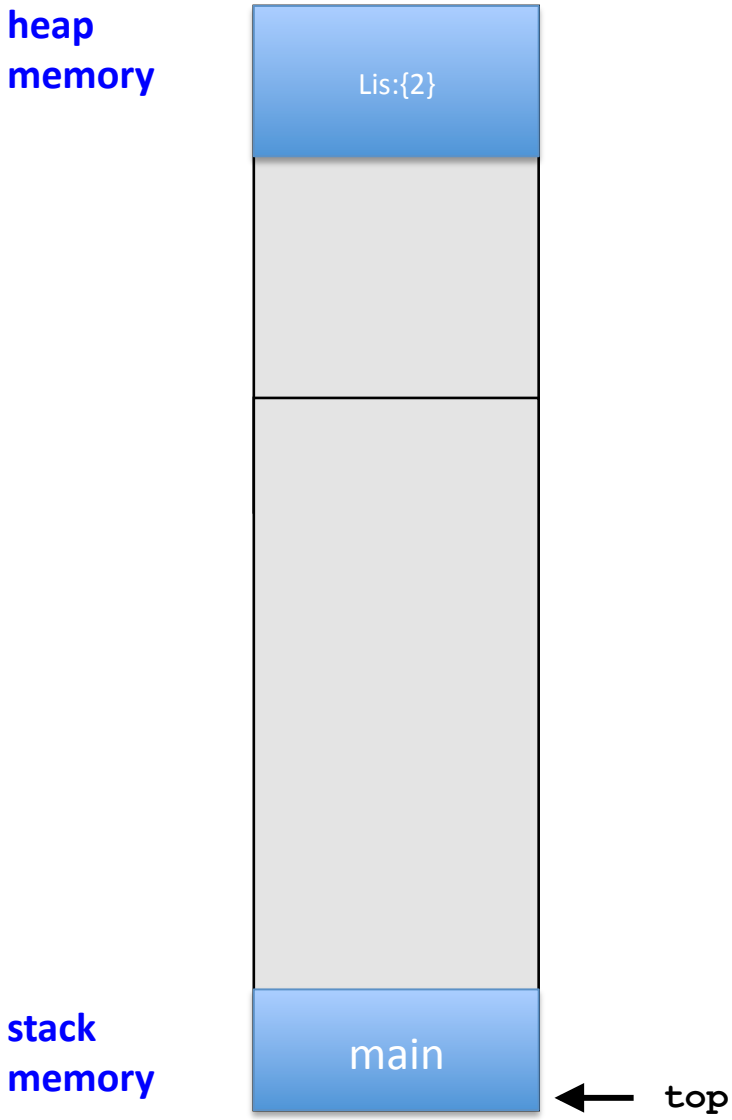

```
Function quicksort(array){
If(array.length<=0) return[]
   var less=[];
   var greater=[];
   var equal=[]
   var pivot=array[0];
 for(var i=0;i<array.length;i+
   +)
 if(pivot>array[i])
      less.push(array[i]);
 if(pivot==array[i])
      equal.push(array[i]);
 if(pivot<array[i])
     greater.push(array[i]);
}
Return quicksort(less).concat
   (equal,quicksort(greater));
}
Var lis=new array(2);

Console.log(quicksort(lis));
```



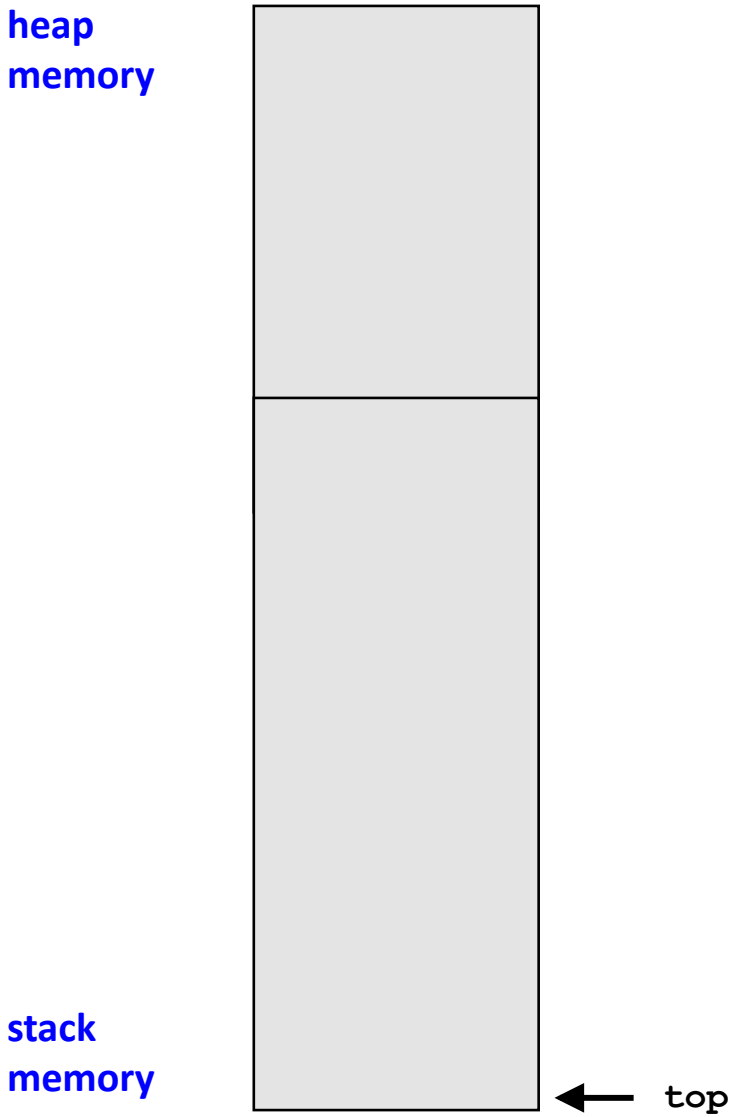

```
Function quicksort(array){
If(array.length<=0) return[]
   var less=[];
   var greater=[];
   var equal=[]
   var pivot=array[0];
 for(var i=0;i<array.length;i+
   +)
 if(pivot>array[i])
      less.push(array[i]);
 if(pivot==array[i])
      equal.push(array[i]);
 if(pivot<array[i])
     greater.push(array[i]);
}
Return quicksort(less).concat
   (equal,quicksort(greater));
}
Var lis=new array(2);

Console.log(quicksort(lis));
```




```
Function quicksort(array){
If(array.length<=0) return[]
   var less=[];
   var greater=[];
   var equal=[]
   var pivot=array[0];
 for(var i=0;i<array.length;i+
   +)
 if(pivot>array[i])
      less.push(array[i]);
 if(pivot==array[i])
      equal.push(array[i]);
 if(pivot<array[i])
     greater.push(array[i]);
}
Return quicksort(less).concat
   (equal,quicksort(greater));
}
Var lis=new array(2);

Console.log(quicksort(lis));
```




```
Function quicksort(array){
If(array.length<=0) return[]
   var less=[];
   var greater=[];
   var equal=[]
   var pivot=array[0];
 for(var i=0;i<array.length;i+
   +)
 if(pivot>array[i])
      less.push(array[i]);
 if(pivot==array[i])
      equal.push(array[i]);
 if(pivot<array[i])
     greater.push(array[i]);
}
Return quicksort(less).concat
   (equal,quicksort(greater));
}
Var lis=new array(2);

Console.log(quicksort(lis));
```

Conclusion R1.3

- I did not need any modifications to the current stack and heap model to show how recursion works with javascript functions.
- Modifications to simplify the model would be greatly appreciated however, as this example used a lot of slide for such a simple case, The function works for much larger cases but the provided model is a bit too detailed for anything much larger than this.