

**EXAMPLE 44:** THE REFERENCING ENVIRONMENT  
CAN BE BUILT FROM FUNCTION PARAMETERS

## EXAMPLE 44: THE REFERENCING ENVIRONMENT CAN BE BUILT FROM FUNCTION PARAMETERS

CLOSURE = SAY WE HAVE A NESTED FUNCTION +  
VARIABLES LOCAL TO THE  
OUTER SCOPE  
"REFERENCING ENVIRONMENT"


THE QUESTION IS - WHAT IS THE  
REFERENCING ENVIRONMENT?

# THE QUESTION IS - WHAT IS THE REFERENCING ENVIRONMENT?

CLOSURE = A diagram illustrating the components of a closure. It consists of a dashed green rectangular box. Inside the box, the text "VARIABLES LOCAL TO THE" is in red, and "SCOPE" is in black. Above the box, the text "SAY WE HAVE A NESTED" is in red, followed by a red plus sign. Below the box, the word "REFERENCING" is written in green.

IN THE EXAMPLES WE SAW SO FAR, THE REFERENCING ENVIRONMENT WAS ALWAYS LOCAL VARIABLES INSIDE A FUNCTION THAT RETURNED THE NESTED FUNCTION

IN THE EXAMPLES WE SAW SO FAR, THE REFERENCING ENVIRONMENT WAS ALWAYS **LOCAL VARIABLES** INSIDE A FUNCTION THAT RETURNED THE NESTED

CLOSURE = A diagram illustrating the components of a closure. It shows a dashed green rectangular box containing the text "VARIABLES LOCAL TO THE" in red and "SCOPE" in black. Above the box, the text "SAY WE HAVE A NESTED" is in red, followed by a red plus sign. Below the box, the word "REFERENCING" is written in green.

BUT THE REFERENCING ENVIRONMENT CAN ALSO BE BUILT FROM **FUNCTION PARAMETERS**

BUT THE REFERENCING ENVIRONMENT CAN  
ALSO BE BUILT FROM FUNCTION PARAMETERS

CLOSURE =

SAY WE HAVE A NESTED +  
VARIABLES LOCAL TO THE  
OUTER SCOPE  
"REFERENCING"

WHAT DOES THAT MEAN? IT MEANS THE FUNCTION  
PARAMETERS BECOME VISIBLE TO OUR NESTED FUNCTION

CLOSURE =

SAY WE HAVE A NESTED +  
VARIABLES LOCAL TO THE  
OUTER SCOPE  
"REFERENCING"

```
function printStuffAboutCircleArray(circleArray,PI) {  
  
    function getArea(circle){  
        console.log("Inside the nested function getArea, P  
" + PI);  
        return PI * circle.radius * circle.radius;  
    };  
    for (var i = 0;i<circleArray.length;i++) {  
        var c = circleArray[i];  
        console.log(c.radius + ", " + getArea(c));  
    }  
    return getArea;  
}
```

```
function printStuffAboutCircleArray(circleArray,PI) {
```

```
    function getArea(circle){
```

```
        console.log("Inside the nested function getArea, PI =  
" + PI);  
        return PI * circle.radius * circle.radius;  
    };
```

```
    for (var i = 0;i<circleArray.length;i++) {  
        var c = circleArray[i];  
        console.log(c.radius + ", " + getArea(c));
```

```
    }
```

```
    return getArea;  
}
```

**CLOSURE =**

SAY WE HAVE A **NESTED** +

**VARIABLES LOCAL TO THE  
OUTER SCOPE**

**"REFERENCING"**

CLOSURE =

SAY WE HAVE A NESTED +  
VARIABLES LOCAL TO THE  
OUTER SCOPE  
"REFERENCING"

```
function printStuffAboutCircleArray(circleArray,PI) {  
    function getArea(circle){  
        console.log("Inside the nested function getArea, PI =  
" + PI);  
        return PI * circle.radius * circle.radius;  
    };  
    for (var i = 0;i<circleArray.length;i++) {  
        var c = circleArray[i];  
        console.log(c.radius + "," + getArea(c));  
    }  
    return getArea;  
}
```



CLOSURE =

SAY WE HAVE A NESTED  
VARIABLES LOCAL TO THE  
OUTER SCOPE  
"REFERENCING"

```
function printStuffAboutCircleArray(circleArray,PI) {
```

```
    function getArea(circle){  
        console.log("Inside the nested function getArea, PI =  
+ PI);  
        return PI * circle.radius * circle.radius;  
    };  
    for (var i = 0;i<circleArray.length;i++) {  
        var c = circleArray[i];  
        console.log(c.radius + ", " + getArea(c));  
    }  
    return getArea;  
}
```

**CLOSURE =** SAY WE HAVE A **NESTED** **+**  
**VARIABLES LOCAL TO THE**  
**OUTER SCOPE**  
**"REFERENCING"**

```
function printStuffAboutCircleArray(circleArray,PI) {  
  
    function getArea(circle){  
        console.log("Inside the nested function getArea, PI =  
" + PI);  
        return PI * circle.radius * circle.radius;  
    }  
    THE OUTER FUNCTION RETURNS THE INNER FUNCTION, AND THAT  
INNER FUNCTION CAN NOW BE SAVED TO A VARIABLE, AND CALLED.  
    for (var i = 0; i < circleArray.length; i++) {  
        var c = circleArray[i];  
        console.log(c.radius + ", " + getArea(c));  
    }  
    return getArea;  
}
```

CLOSURE =

SAY WE HAVE A NESTED  
+  
VARIABLES LOCAL TO THE  
OUTER SCOPE  
"REFERENCING"

THE OUTER FUNCTION RETURNS THE INNER FUNCTION, AND THAT  
INNER FUNCTION CAN NOW BE SAVED TO A VARIABLE, AND CALLED.

var areaFunction = printStuffAboutCircleArray([], 3.14159)  
areaFunction(circle1)  
NESTED FUNCTION

FUNCTION PARAMETER

CLOSURE VARIABLE

Inside the nested function getArea, PI = 3.14159