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 OUTERSCOPE

"REFERENCING ENVIRONMENT"

THE QUESTION IS - WHAT IS THE REFERENCING ENVIRONMENT?

THE QUESTION IS - WHAT IS THE REFERENCING ENVIRONMENT?

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

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

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

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

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.

FUNCTION PARAMETER

```
var areaFunction = printStuffAboutCircleArray([],3.141
areaFunction(circle1);
```

CLOSURE VARIABLE

NESTED FUNCTION

```
Inside the nested function getArea, PI = 3.14159
```