EXAMPLE 41: NESTED FUNCTIONS CAN BE DECLARED (THEY NEED NOT ALWAYS BE FUNCTION LITERALS)

EXAMPLE 41: NESTED FUNCTIONS CAN BE DECLARED (THEY NEED NOT ALWAYS BE FUNCTION LITERALS)

LET US REDO THE EXAMPLE WE JUST DID, BUT THIS TIME, WE WILL MAKE BOTH THE NESTED FUNCTIONS DECLARED FUNCTIONS, RATHER THAN FUNCTION LITERALS.

LET US REDO THE EXAMPLE WE JUST DID, BUT THIS TIME, WE WILL MAKE BOTH THE NESTED FUNCTIONS DECLARED FUNCTIONS, RATHER THAN FUNCTION LITERALS.

WE JUST SAW HOW TO USE NESTED FUNCTIONS (IN FACT 2 LEVELS OF NESTED FUNCTIONS!)

A DECLARED FUNCTION IS SIMPLY A TRADITIONAL FUNCTION

```
function declaredFunction() {
  console.log("Declared function..I ex
}
```

LET US REDO THE EXAMPLE WE JUST DID, BUT THIS TIME, WE WILL MAKE BOTH THE NESTED FUNCTIONS DECLARED FUNCTIONS, RATHER THAN FUNCTION LITERALS.

RECALL THAT DECLARED FUNCTIONS ARE "HOISTED" TO THE TOP, I.E. THEY ARE ALWAYS AVAILABLE!

LET US REDO THE EXAMPLE WE JUST DID, BUT THIS TIME, WE WILL MAKE BOTH THE NESTED FUNCTIONS DECLARED FUNCTIONS, RATHER THAN FUNCTION LITERALS.

FUNCTION LITERALS ARE
VARIABLES THAT HOLD FUNCTIONS,

```
var someFunction = function() {
  console.log("Inside a function literal -
}
```

LET US REDO THE EXAMPLE WE JUST DID, BUT THIS TIME, WE WILL MAKE BOTH THE NESTED FUNCTIONS DECLARED FUNCTIONS, RATHER THAN FUNCTION LITERALS.

ALTHOUGH THE VAST MAJORITY OF NESTED FUNCTIONS ARE FUNCTION LITERALS, THIS NEED NOT BE THE CASE!

LET US REDO THE EXAMPLE WE JUST DID, BUT THIS TIME, WE WILL MAKE BOTH THE NESTED FUNCTIONS DECLARED FUNCTIONS, RATHER THAN FUNCTION LITERALS.

LET US REDO THE EXAMPLE

MAKE BOTH THE NESTED FUNCTIONS DECLARED FUNCTIONS, RATHER THAN FUNCTION LITERALS.

```
function Rectangle(l,b) {
  this.length = l;
  this.breadth = b;
}

window.onload = function(){
  var rectangle1 = new Rectangle(3,4);
  var rectangle2 = new Rectangle(4,5);
  var rectangle3 = new Rectangle(5,6);

  var rectArray = [rectangle1,rectangle2,rectangle3];
  //printStuffAboutRectangleArray(rectArray);
  var printStuffAboutRectangleArray = function(rectangleArray) {

    var getArea = function(rectangle){
      console.log("Inside a nested function that calculates the area
rectangle");
      return rectangle.length * rectangle.breadth;
    }
    for (var i = 0;i<rectangleArray.length;i++) {
       var r = rectangleArray[i];
       console.log(r.length + "," + r.breadth + "," + getArea(r));
    }
  }
  printStuffAboutRectangleArray(rectArray);
```

FUNCTION LITERALS

LET US REDO THE EXAMPLE MAKE BOTH THE NESTED FUNCTIONS DECLARED FUNCTIONS, RATHER THAN FUNCTION LITERALS.

FUNCTION LITERALS

```
var printStuffAboutRectangleArray = function(rectangleArray) {

    var getArea = function(rectangle){
      console.log("Inside a nested function that calculates the area o
rectangle");
      return rectangle.length * rectangle.breadth;
    }
    for (var i = 0;i<rectangleArray.length;i++) {
       var r = rectangleArray[i];
       console.log(r.length + "," + r.breadth + "," + getArea(r));
    }
}
```

LET US REDO THE EXAMPLE MAKE BOTH THE NESTED FUNCTIONS DECLARED FUNCTIONS, RATHER THAN FUNCTION LITERALS.

FUNCTION LITERALS

```
var printStuffAboutRectangleArray = function(rectangleArray) {

   var getArea = function(rectangle){
      console.log("Inside a nested function that calculates the area o
rectangle");
      return rectangle.length * rectangle.breadth;
   }
   for (var i = 0;i<rectangleArray.length;i++) {
      var r = rectangleArray[i];
      console.log(r.length + "," + r.breadth + "," + getArea(r));
   }
}
printStuffAboutRectangleArray(rectArray);
// the nested function getArea will not exist here!
```

LET US REDO THE EXAMPLE MAKE BOTH THE NESTED FUNCTIONS DECLARED FUNCTIONS, RATHER THAN FUNCTION LITERALS.

DECLARED FUNCTIONS

```
printStuffAboutRectangleArray(rectangleArray) {

    getArea(rectangle){
      console.log("Inside a nested function that calculates the area o
rectangle");
      return rectangle.length * rectangle.breadth;
    }
    for (var i = 0;i<rectangleArray.length;i++) {
       var r = rectangleArray[i];
       console.log(r.length + "," + r.breadth + "," + getArea(r));
    }
  }
printStuffAboutRectangleArray(rectArray);
// the nested function getArea will not exist here!
```

A FEW POINTS WORTH NOTING ABOUT THESE DECLARED FUNCTIONS

- THESE ARE DECLARED FUNCTIONS AND SO ARE AVAILABLE EVERYWHERE IN CODE
- THE INNER NESTED FUNCTION IS STILL LOCAL TO THE OUTER NESTED FUNCTION AND CAN'T BE USED OUTSIDE IT

- THESE ARE DECLARED FUNCTIONS AND SO ARE AVAILABLE EVERYWHERE IN CODE

```
window.onload = function(){
  var rectangle1 = new Rectangle(3,4);
  var rectangle2 = new Rectangle(4,5);
  var rectangle3 = new Rectangle(5,6);

  var rectArray = [rectangle1,rectangle2,rectangle3];

  printStuffAboutRectangleArray(rectArray);
  //getArea(rectangle1);

  function printStuffAboutRectangleArray(rectangleArray) {

    function getArea(rectangle){
      console.log("Inside a nested function that calculates the
rectangle");
      return rectangle.length * rectangle.breadth;
    }
    for (var i = 0;i<rectangleArray.length;i++) {
       var r = rectangleArray[i];
       console.log(r.length + "," + r.breadth + "," + getArea(r
    }
  }

  printStuffAboutRectangleArray(rectArray);
  //getArea(rectangle1);
```

- THESE ARE DECLARED FUNCTIONS AND SO ARE AVAILABLE EVERYWHERE IN CODE

FUNCTION CAN BE CALLED EVEN BEFORE ITS DECLARATION, THANKS TO HOISTING!

```
window.onload = function(){
  var rectangle1 = new Rectangle(3,4);
  var rectangle2 = new Rectangle(4,5);
  var rectangle3 = new Rectangle(5,6);

  var rectArray = [rectangle1, rectangle2, rectangle3];

  printStuffAboutRectangleArray(rectArray);
  //getArea(rectangle1);

  function printStuffAboutRectangleArray(rectangleArray) {

    function getArea(rectangle){
      console.log("Inside a nested function that calculates the
rectangle");
      return rectangle.length * rectangle.breadth;
    }
    for (var i = 0;i<rectangleArray.length;i++) {
      var r = rectangleArray[i];
      console.log(r.length + "," + r.breadth + "," + getArea(r
    }
  }

  printStuffAboutRectangleArray(rectArray);
  //getArea(rectangle1);
```

A FEW POINTS WORTH NOTING ABOUT THESE DECLARED FUNCTIONS

- THESE ARE DECLARED FUNCTIONS AND SO ARE AVAILABLE EVERYWHERE IN CODE
- THE INNER NESTED FUNCTION IS STILL LOCAL TO THE OUTER NESTED FUNCTION AND CAN'T BE USED OUTSIDE IT

- THE INNER NESTED FUNCTION IS STILL LOCAL TO THE OUTER NESTED FUNCTION AND CAN'T BE USED OUTSIDE IT

```
window.onload = function(){
  var rectangle1 = new Rectangle(3,4);
  var rectangle2 = new Rectangle(4,5);
  var rectangle3 = new Rectangle(5,6);

  var rectArray = [rectangle1,rectangle2,rectangle3];

  printStuffAboutRectangleArray(rectArray);
  getArea(rectangle1);

  function printStuffAboutRectangleArray(rectangleArray) {

    function getArea(rectangle){
      console.log("Inside a nested function that calculates the
rectangle");
      return rectangle.length * rectangle.breadth;
    }
    for (var i = 0;i<rectangleArray.length;i++) {
       var r = rectangleArray[i];
       console.log(r.length + "," + r.breadth + "," + getArea(r
    }
  }

  printStuffAboutRectangleArray(rectArray);
  //getArea(rectangle1);
```

- THE INNER NESTED FUNCTION IS STILL LOCAL TO THE OUTER NESTED FUNCTION AND CAN'T BE USED OUTSIDE IT

```
window.onload = function(){
    var rectangle1 = new Rectangle(3,4);
    var rectangle2 = new Rectangle(4,5);
    var rectangle3 = new Rectangle(5,6);

    var rectArray = [rectangle1,rectangle2,rectangle3];

    printStuffAboutRectangleArray(rectArray);
    getArea(rectangle1);

    function printStuffAboutRectangleArray(rectangleArray) {

      function getArea(rectangle){
        console.log("Inside a nested function that calculates the
rectangle");
        return rectangle.length * rectangle.breadth;
      }

        console.log(r.length + "," + r.breadth + "," + getArea(r
      }
    }

    printStuffAboutRectangleArray(rectArray);
    //getArea(rectangle1);
```

THIS FUNCTION STILL CAN NOT BE ACCESSED OUTSIDE THE OUTER NESTED FUNCTION

```
✖ ▶ Uncaught ReferenceError: getArea is not defined
```