EXAMPLE 40: NESTED FUNCTIONS

IT HAPPENS ALL THE TIME.

A FEW LINES OF CODE INSIDE A FUNCTION THAT YOU COPY-PASTE OVER.

EVERYONE DOES IT - BUT ITS BAD PROGRAMMING PRACTICE.

EVERYONE DOES IT - BUT ITS BAD PROGRAMMING PRACTICE.

NESTED FUNCTIONS CAN MAKE A REAL DIFFERENCE HERE.

THIS IS ESPECIALLY TRUE IN JAVASCRIPT, WHERE CODE READABILITY IS A REAL PROBLEM.

NESTED FUNCTIONS

```
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 of a
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!
```

NESTED FUNCTIONS

```
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 of a
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!
```

```
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 of a
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!
```

WE HAVE A SIMPLE OBJECT CONSTRUCTOR

```
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 of a
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!
```

WE CREATE A FEW DIFFERENT OBJECTS FROM THIS OBJECT CONSTRUCTOR

NOW, LET'S SAY THERE IS SOME SIMPLE OPERATION WE'D LIKE TO PERFORM WITH ALL OF THESE.

NOW, LET'S SAY THERE IS SOME SIMPLE OPERATION
WE'D LIKE TO PERFORM WITH ALL OF THESE.

WE HAVE 3 OPTIONS

- ADD AN OBJECT PROPERTY (TO THE OBJECT) FOR THIS SIMPLE
- SET UP A NESTED FUNCTION, AND APPLY IT TO EACH OF THE OBJECTS
- JUST TYPE OUT THE CODE FOR THIS SIMPLE OPERATION 3 TIMES,
  ONCE PER OBJECT WE CREATED

NOW, LET'S SAY THERE IS SOME SIMPLE OPERATION WE'D LIKE TO PERFORM WITH ALL OF THESE.

WE HAVE 3 OPTIONS

- ADD AN OBJECT PROPERTY (TO THE OBJECT) FOR THIS SIMPLE ACTION OVERKILL - INVOLVES CHANGING THE OBJECT
- SET UP A NESTED FUNCTION, AND APPLY IT TO EACH OF THE OBJECTS
- JUST TYPE OUT THE CODE FOR THIS SIMPLE OPERATION 3 TIMES, ONCE PER OBJECT WE CREATED

NOW, LET'S SAY THERE IS SOME SIMPLE OPERATION WE'D LIKE TO PERFORM WITH ALL OF THESE.

WE HAVE 3 OPTIONS

- ADD AN OBJECT PROPERTY (TO THE OBJECT) FOR THIS SIMPLE ACTION
- SET UP A NESTED FUNCTION, AND APPLY IT TO EACH OF THE OBJECTS
- JUST TYPE OUT THE CODE FOR THIS SIMPLE OPERATION 3 TIMES, ONCE PER OBJECT WE CREATED

VERY HACKY - OVER TIME, COPY-PASTING CODE LEADS TO PROBLEMS.

NOW, LET'S SAY THERE IS SOME SIMPLE OPERATION WE'D LIKE TO PERFORM WITH ALL OF THESE.

WE HAVE 3 OPTIONS

- ADD AN OBJECT PROPERTY (TO THE OBJECT) FOR THIS SIMPLE ACTION
- SET UP A NESTED FUNCTION, AND APPLY IT TO EACH OF THE OBJECTS

AHA! PERFECT, THE BEST MIDDLE OF THE ROAD SOLUTION!

- JUST TYPE OUT THE CODE FOR THIS SIMPLE OPERATION 3 TIMES, ONCE PER OBJECT WE CREATED

- SET UP A NESTED FUNCTION, AND APPLY IT TO EACH OF THE OBJECTS

ACTUALLY, WE DO EVEN BETTER - WE BUILD A NESTED FUNCTION INSIDE A NESTED FUNCTION!!

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

window.onload = function(){
  var rectangle1 = new Rectangle(3,4);
  var rectangle2 = new Rectangle(5,7);
  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 of a
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!
```

ACTUALLY, WE DO EVEN BETTER - WE BUILD A NESTED FUNCTION INSIDE A NESTED FUNCTION!!

OUTER NESTED FUNCTION, APPLY IT TO AN ARRAY OF OBJECTS

```
var printStuffAboutRectangleArray = function(rectangleArray) {

  var getArea = function(rectangle){
    console.log("Inside a nested function that calculates the area of a 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));
  }
}
```

ACTUALLY, WE DO EVEN BETTER - WE BUILD A NESTED FUNCTION INSIDE A NESTED FUNCTION!!

OUTER NESTED FUNCTION, APPLY IT TO AN ARRAY OF OBJECTS

```
var printStuffAboutRectangleArray = function(rectangleArray) {

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

INNER NESTED FUNCTION, APPLY IT TO EACH OBJECT OF THE ARRAY!

ACTUALLY, WE DO EVEN BETTER - WE BUILD A NESTED FUNCTION INSIDE A NESTED FUNCTION!!

OUTER NESTED FUNCTION, APPLY IT TO AN ARRAY OF OBJECTS

INNER NESTED FUNCTION, APPLY IT TO EACH OBJECT

```
var printStuffAboutRectangleArray = function(rectangleArray) {

  var getArea = function(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));
  }
}
```

ACTUALLY, WE DO EVEN BETTER - WE BUILD A NESTED FUNCTION INSIDE A NESTED FUNCTION!!

OUTER NESTED FUNCTION, APPLY IT TO AN ARRAY OF OBJECTS

INNER NESTED FUNCTION, APPLY IT TO EACH OBJECT

```
var printStuffAboutRectangleArray = function(rectangleArray) {

  var getArea = function(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));
  }
}
```

ACTUALLY, WE DO EVEN BETTER - WE BUILD A NESTED FUNCTION INSIDE A NESTED FUNCTION!!

OUTER NESTED FUNCTION, APPLY IT TO AN ARRAY OF OBJECTS

INNER NESTED FUNCTION, APPLY IT TO EACH OBJECT

```
var printStuffAboutRectangleArray = function(rectangleArray) {

  var getArea = function(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));
  }
}
```

ACTUALLY, WE DO EVEN BETTER - WE BUILD A NESTED FUNCTION INSIDE A NESTED FUNCTION!!

OUTER NESTED FUNCTION, APPLY IT TO AN ARRAY OF OBJECTS

INNER NESTED FUNCTION, APPLY IT TO EACH OBJECT

```
var printStuffAboutRectangleArray = function(rectangleArray) {

  var getArea = function(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));
  }
}
```

ACTUALLY, WE DO EVEN BETTER - WE BUILD A NESTED FUNCTION INSIDE A NESTED FUNCTION!!

LET'S MAKE SURE WE GOT THE BIG PICTURE RIGHT!

INNER NESTED FUNCTION, APPLY IT TO EACH OBJECT OF THE ARRAY!

```
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 of a
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!
```

WE HAVE A SIMPLE OBJECT CONSTRUCTOR

```
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 of a
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!
```

WE CREATE A FEW DIFFERENT OBJECTS FROM THIS OBJECT CONSTRUCTOR

THERE IS SOME SIMPLE OPERATION WE'D LIKE TO PERFORM WITH ALL OF THESE.

WE CREATED AN ARRAY OF THESE OBJECTS..

```
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 of a
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!
```

WE CREATED AN ARRAY OF THESE OBJECTS..

AND PASSED IT TO A NESTED FUNCTION (THE OUTER NESTED FUNCTION)

```
var printStuffAboutRectangleArray = function(rectangleArray) {

  var getArea = function(rectangle){
    console.log("Inside a nested function that calculates the area of a 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));
  }
}
```

WE CREATED AN ARRAY OF THESE OBJECTS..

AND PASSED IT TO A NESTED FUNCTION (THE OUTER NESTED FUNCTION)

```
var printStuffAboutRectangleArray = function(rectangleArray) {

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

AND CREATED AN INNER NESTED FUNCTION THAT OPERATES ON 1 ELEMENT OF THIS ARRAY!

```
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 of a
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!
  //getArea(rectangle1);
};
```

1. CREATE ARRAY

2. CREATE (OUTER) NESTED

3. CALL NESTED FUNCTION WITH OUR ARRAY

A FEW POINTS WORTH NOTING ABOUT THESE NESTED FUNCTIONS

- THESE ARE FUNCTION LITERALS AND SO ARE ONLY AVAILABLE AFTER THE FUNCTION EXPRESSION IS EVALUATED
- THE INNER NESTED FUNCTION IS LOCAL TO THE OUTER NESTED FUNCTION AND CAN'T BE USED OUTSIDE IT

```
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("Calculating the area of a
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));
    }
  }
```

- THESE ARE FUNCTION LITERALS AND SO ARE ONLY AVAILABLE AFTER THE FUNCTION EXPRESSION IS EVALUATED

THE FUNCTION EXPRESSION IS EVALUATED HERE

```
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 of a
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));
    }
  }
```

- THESE ARE FUNCTION LITERALS AND SO ARE ONLY AVAILABLE AFTER THE FUNCTION EXPRESSION IS EVALUATED

ATTEMPTING TO USE IT HERE WOULD YIELD AN ERROR

A FEW POINTS WORTH NOTING ABOUT THESE NESTED FUNCTIONS

- THESE ARE FUNCTION LITERALS AND SO ARE ONLY AVAILABLE AFTER THE FUNCTION EXPRESSION IS EVALUATED
- THE INNER NESTED FUNCTION IS LOCAL TO THE OUTER NESTED FUNCTION AND CAN'T BE USED OUTSIDE IT

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

OUTER NESTED FUNCTION

INNER NESTED FUNCTION

```
var printStuffAboutRectangleArray = function(rectangleArray) {

    var getArea = function(rectangle){
      console.log("Inside a nested function that calculates the area of a
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!
  //getArea(rectangle1);
};
```

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

```
var rectangle = new Rectangle(5,6);
//printStuffAboutRectangleArray(rectArray);
var printStuffAboutRectangleArray = function(rectangleArray) {

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

  // the nested function getArea will not
exist here!
  //getArea(rectangle);
};
```

Uncaught ReferenceError: getArea is not defined

ATTEMPTING TO USE THE INNER NESTED FUNCTION HERE WOULD YIELD AN ERROR

NESTED FUNCTIONS ARE KEY TO UNDERSTANDING AND USING CLOSURES

THIS MAKES THEM ONE OF THE MOST IMPORTANT CONCEPTS IN JAVASCRIPT.