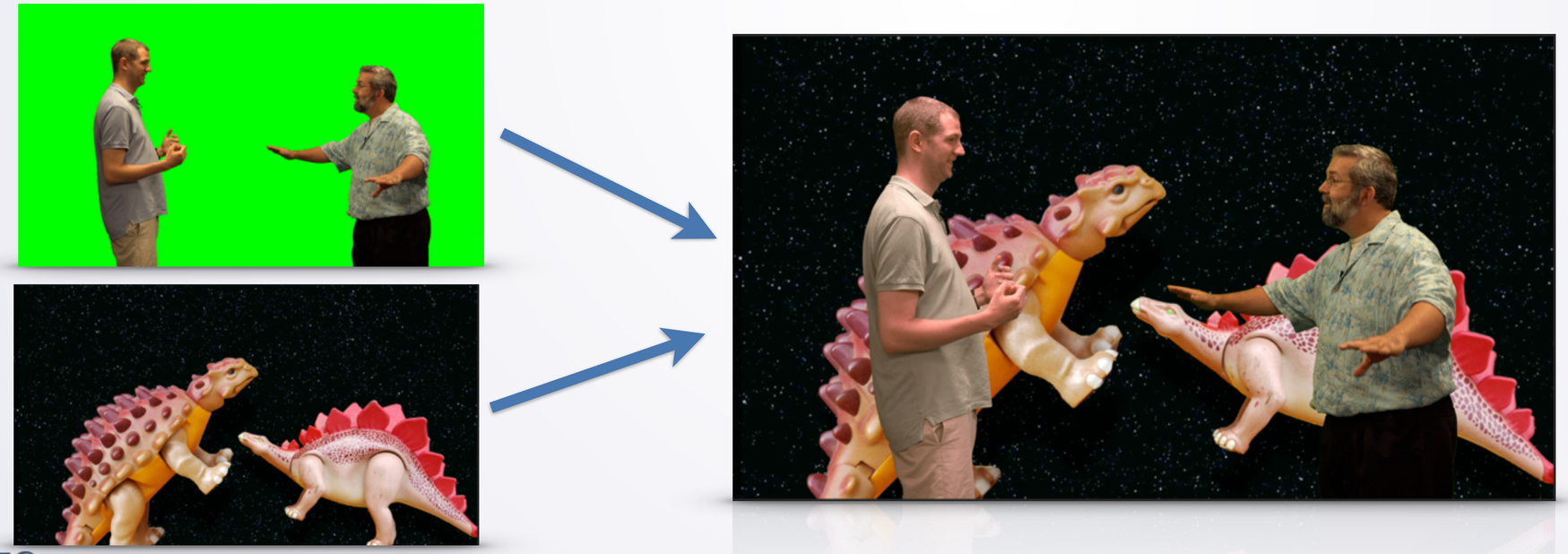
Functions

Creating and Using Functions

- Powerful addition to code vocabulary
  Create new “words” and use them!
  - Programs are easier to develop, test, and change
  - Re-run, re-use, and share code: code libraries



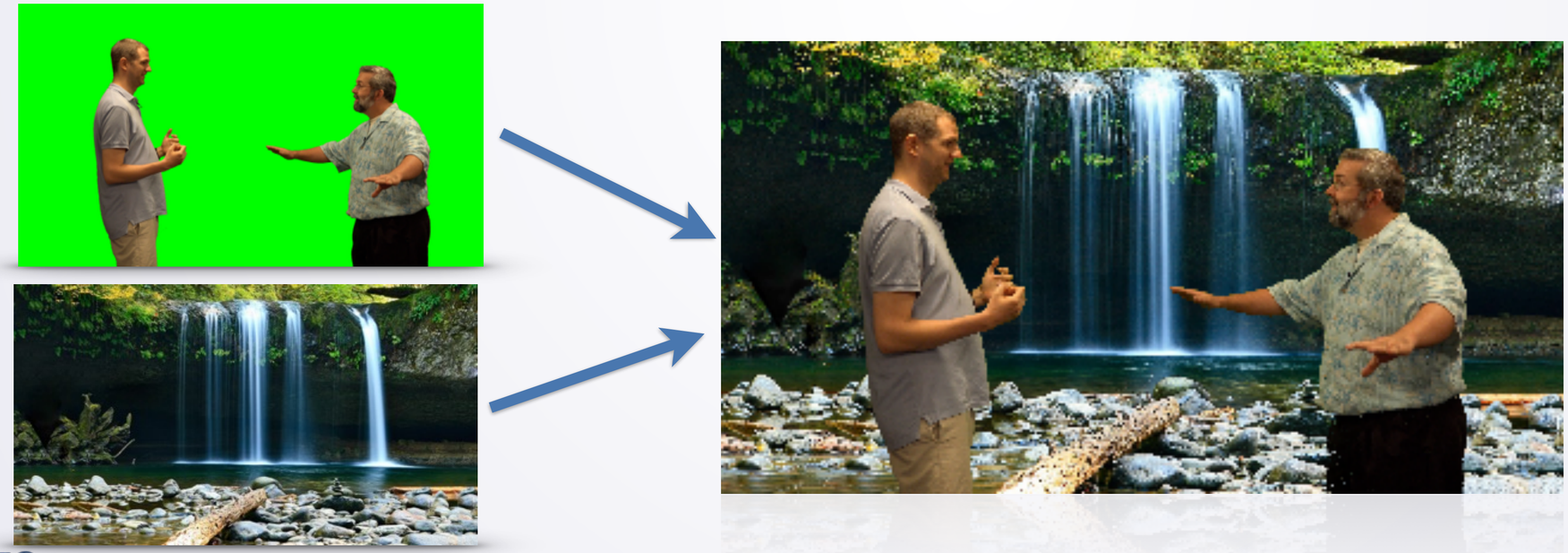
Creating and Using Functions

- Powerful addition to code vocabulary
  Create new "words" and use them!
  - Programs are easier to develop, test, and change
  - Re-run, re-use, and share code: code libraries

greenScreen(p1,p2)

Creating and Using Functions

- Powerful addition to code vocabulary
  Create new “words” and use them!
  - Programs are easier to develop, test, and change
  - Re-run, re-use, and share code: code libraries

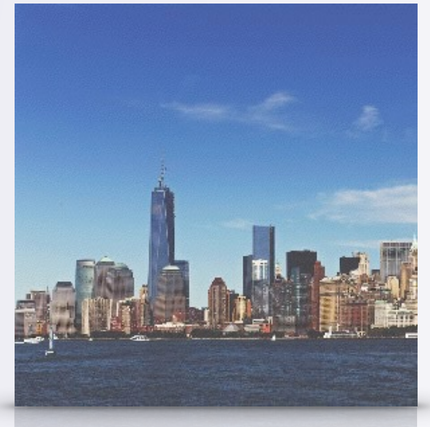


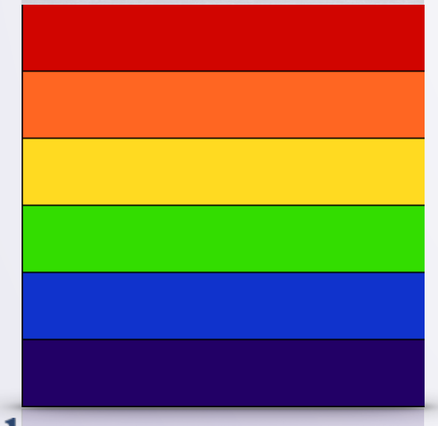
What Do You Know Already?

- You already have an extensive vocabulary!
  - What can we do with a Pixel? Pixel methods:
    - **`p.getRed(), p.getBlue(), p.getGreen(), p.setRed(), p.setBlue(), p.setGreen()`**
  - What are the SimpleImage methods?
    - **`new("foo.png"), new(w,h), im.getWidth(), im.getHeight(), im.values(), im.getPixel(x,y), im.setPixel(x,y,p)`**
- When you write new functions, you add to vocabulary!

Re-using Functions with Images

- On Facebook thousands of users did this
  - Write code once, use a million times!

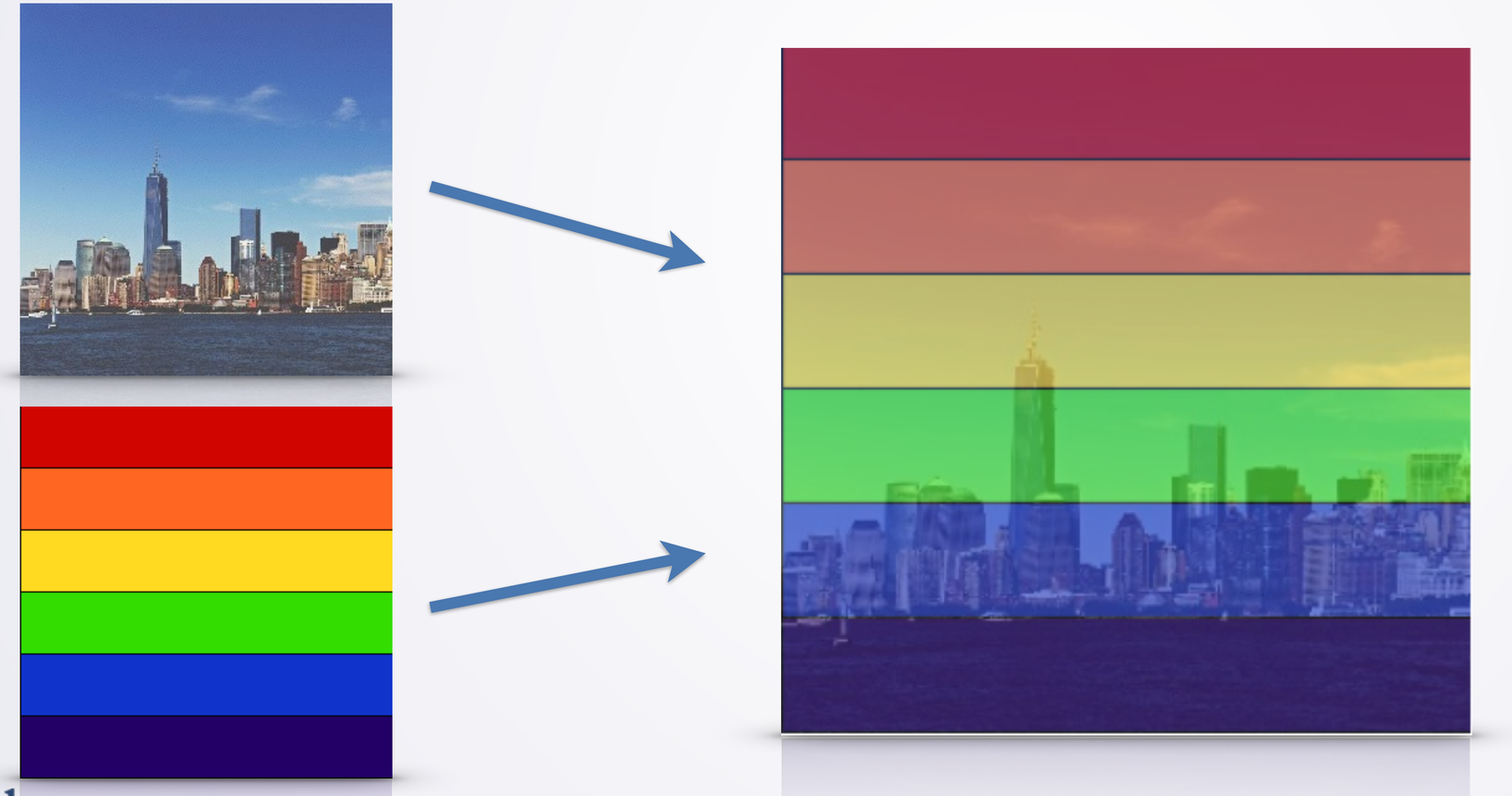




rainbowColor(p1,p2)

Re-using Functions with Images

- On Facebook thousands of users did this
  - Write code once, use a million times!



Coding and Using Steganography

- You will be able to find hidden meaning in the universe!
  - "Hide" one image inside another

You are learning about a new, digital world --- a world in which you can create things like web pages and programs and then share them with your friends or everyone.

Duke
UNIVERSITY

Avoid Repeated/Duplicated Code

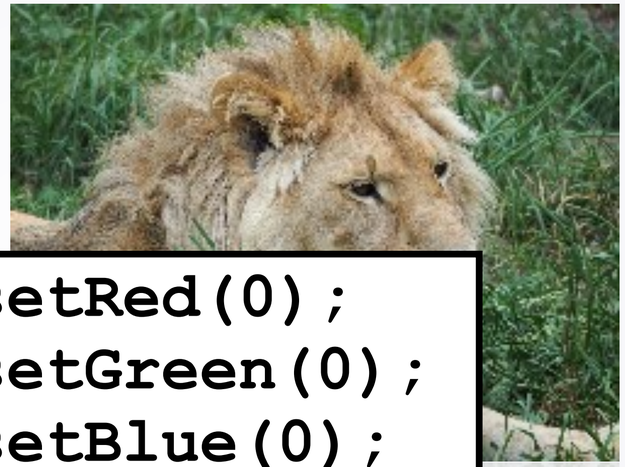
- Easier to make code correct and change it

```
var image = new SimpleImage("lion.jpg");
for (var pixel of image.values()) {
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < 10){
        pixel.setRed(0);
        pixel.setGreen(0);
        pixel.setBlue(0);
    }
    if (y < 10){
        pixel.setRed(0);
        pixel.setGreen(0);
        pixel.setBlue(0);
    }
    if (x >= image.getWidth()-10){
        pixel.setRed(0);
        pixel.setGreen(0);
        pixel.setBlue(0);
    }
    if (y >= image.getHeight()-10){
        pixel.setRed(0);
        pixel.setGreen(0);
        pixel.setBlue(0);
    }
}
```

Avoid Repeated/Duplicated Code

- Easier to make code correct and change it

```
var image = new SimpleImage("lion.jpg");
for (var pixel of image.values()) {
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < 10){
        pixel.setRed(0);
        pixel.setGreen(0);
        pixel.setBlue(0);
    }
    if (y < 10){
        pixel.setRed(0);
        pixel.setGreen(0);
        pixel.setBlue(0);
    }
    if (x >= image.getWidth()-10){
        pixel.setRed(0);
        pixel.setGreen(0);
        pixel.setBlue(0);
    }
    if (y >= image.getHeight()-10){
        pixel.setRed(0);
        pixel.setGreen(0);
        pixel.setBlue(0);
    }
}
```



```
pixel.setRed(0);
pixel.setGreen(0);
pixel.setBlue(0);
```

How Functions Work: Write and Call

- Design, write, test, and call: how functions work

```
function setBlack(pixel){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return pixel;
}
var image = new SimpleImage("lion.jpg");
for (var pixel of image.values()) {
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < 10){
        pixel = setBlack(pixel);
    }
    if (y < 10){
        pixel = setBlack(pixel);
    }
    if (x >= image.getWidth()-10){
        pixel = setBlack(pixel);
    }
    if (y >= image.getHeight()-10){
        pixel = setBlack(pixel);
    }
}
```

How Functions Work: Write and Call

- Design, write, test, and call: how functions work
  - Let's look at how the function **`setBlack`** is called

```
function setBlack(pixel){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return pixel;
}
var image = new SimpleImage("lion.j
for (var pixel of image.values()) {
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < 10){
        pixel = setBlack(pixel);
    }
    if (y < 10){
        pixel = setBlack(pixel);
    }
    if (x >= image.getWidth()-10){
        pixel = setBlack(pixel);
    }
    if (y >= image.getHeight()-10){
        pixel = setBlack(pixel);
    }
}
```

```
function setBlack(pixel){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return pixel;
}
```

Duke UNIVERSITY

Anatomy of a Function

- Understanding functions and function calls
  - Writing: name, parameter list, return value

```
function setBlack(px){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return px;
}
```

```
if (x < 10){
    pixel = setBlack(pixel);
}
```

Anatomy of a Function

- Understanding functions and function calls
  - Writing: name, parameter list, return value

```
function setBlack(px){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return px;
}
```

```
if (x < 10){
    pixel = setBlack(pixel);
}
```

Anatomy of a Function

- Understanding functions and function calls
  - Writing: name, parameter list, return value

```
function setBlack(px){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return px;
}
```

```
if (x < 10){
    pixel = setBlack(pixel);
}
```

Anatomy of a Function

- Understanding functions and function calls
  - Writing: name, parameter list, return value
  - Calling: name, arguments, use return value

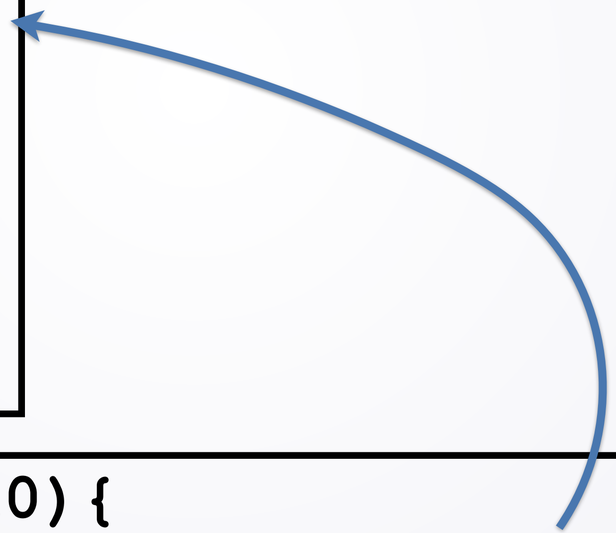
```
function setBlack(px){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return px;
}
```

```
if (x < 10){
    pixel = setBlack(pixel);
}
```

Anatomy of a Function

- Understanding functions and function calls
  - Writing: name, parameter list, return value
  - Calling: name, arguments, use return value

```
function setBlack(px){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return px;
}
```



```
if (x < 10){
    pixel = setBlack(pixel);
}
```

Anatomy of a Function

- Understanding functions and function calls
  - Writing: name, parameter list, return value
  - Calling: name, arguments, use return value

```
function setBlack(px){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return px;
}
```

```
if (x < 10){
    pixel = setB( px );
}
```

Modifying Working Code

- Code worked, confidence in modifying it!
  - Change name, make somewhat general

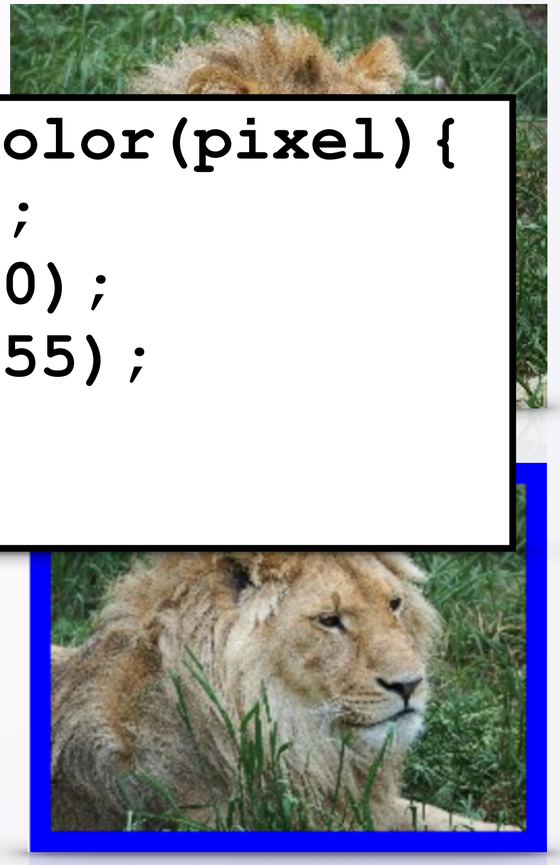
```
function setBorderColor(pixel){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return pixel;
}
var image = new SimpleImage("lion.jpg");
for (var pixel of image.values()) {
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < 10 || y < 10 ||
        x >= image.getWidth()-10 ||
        y >= image.getHeight()-10){
        pixel = setBorderColor(pixel);
    }
}
```

Modifying Working Code

- Code worked, confidence in modifying it!
  - Change name, make somewhat general

```
function setBorderColor(pixel){
    pixel.setRed(0);
    pixel.setGre
    pixel.setBlu
    return pixel
}
var image = new
for (var pixel o
    var x = pixe
    var y = pixe
    if (x < 10 || y < 10 ||
        x >= image.getWidth()-10 ||
        y >= image.getHeight()-10){
        pixel = setBorderColor(pixel);
    }
}
```

```
function setBorderColor(pixel){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(255);
    return pixel;
}
```



Code Easier to Read, Verify, and Debug

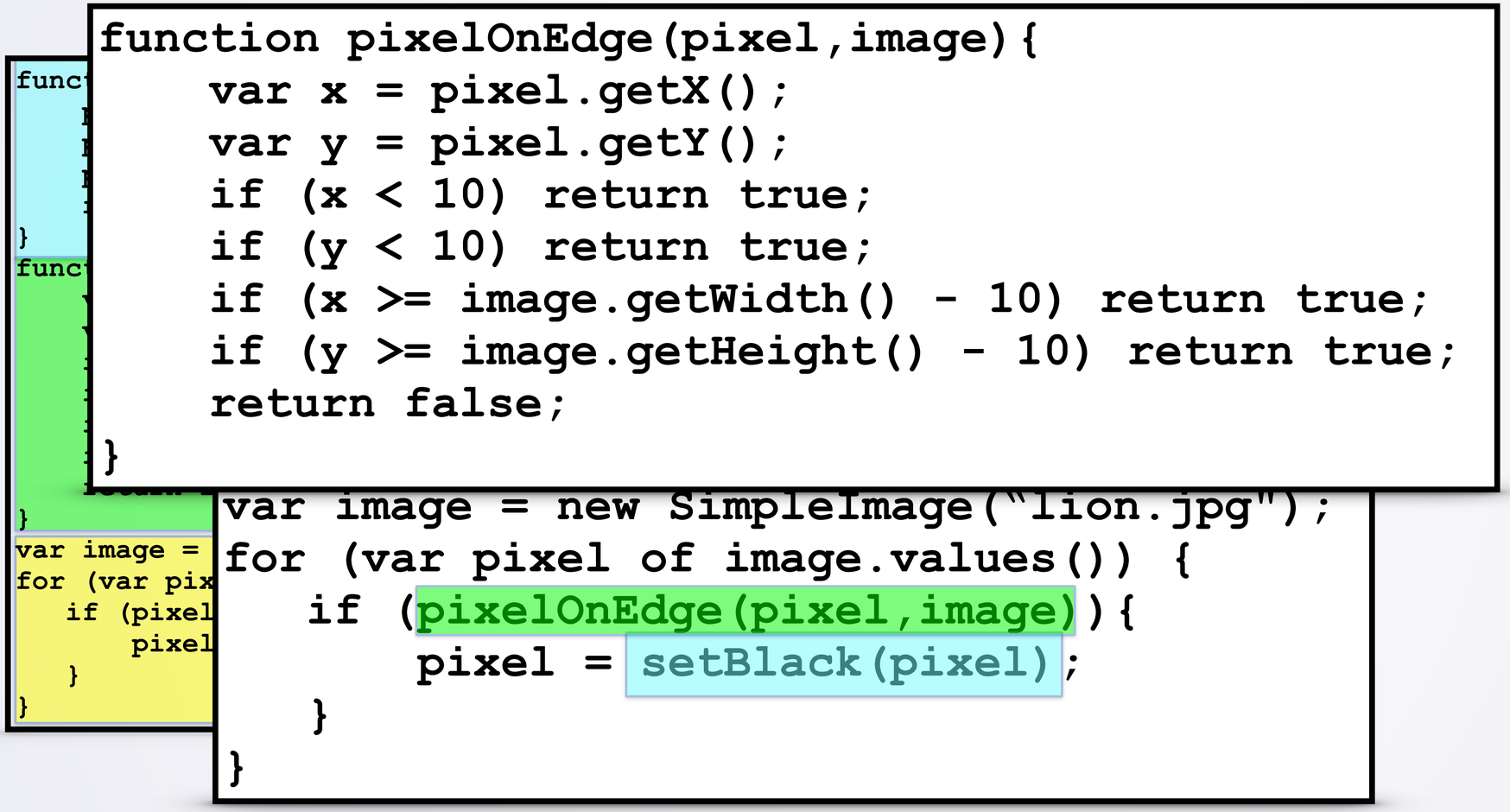
```
function setBlack(pixel){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return pixel;
}
function pixelOnEdge(pixel,image){
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < 10) return true;
    if (y < 10) return true;
    if (x >= image.getWidth() - 10) return true;
    if (y >= image.getHeight() - 10) return true;
    return false;
}
var image = new SimpleImage("lion.jpg");
for (var pixel of image.values()) {
   if (pixelOnEdge(pixel,image)){
       pixel = setBlack(pixel);
   }
}
```

Code Easier to Read, Verify, and Debug

```
function setBlack(pixel){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return pixel;
}
function pixelOnEdge(pixel,image){
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < 10) return true;
    if (y < 10) return true;
    if (x >= image.getWidth() - 10) return true;
    if (y >= image.getHeight() - 10) return true;
    return f
}
var image =
for (var pix
   if (pixel
       pixel
   }
}
```

```
var image = new SimpleImage("lion.jpg");
for (var pixel of image.values()) {
    if (pixelOnEdge(pixel,image)){
        pixel = setBlack(pixel);
    }
}
```

Code Easier to Read, Verify, and Debug



```
function pixelOnEdge(pixel,image){
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < 10) return true;
    if (y < 10) return true;
    if (x >= image.getWidth() - 10) return true;
    if (y >= image.getHeight() - 10) return true;
    return false;
}
```

```
var image = new SimpleImage("lion.jpg");
for (var pixel of image.values()) {
    if (pixelOnEdge(pixel,image)){
        pixel = setBlack(pixel);
    }
}
```

Adding a Parameter to a Function

```
function setBlack(pixel){
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(0);
    return pixel;
}
function pixelOnEdge(pixel,image){
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < 10) return true;
    if (y < 10) return true;
    if (x >= image.getWidth() - 10) return true;
    if (y >= image.getHeight() - 10) return true;
    return false;
}
var image = new SimpleImage("lion.jpg");
for (var pixel of image.values()) {
   if (pixelOnEdge(pixel,image)){
       pixel = setBlack(pixel);
   }
}
```

Adding a Parameter to a Function

```
function pixelOnEdge(pixel,image){
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < 10) return true;
    if (y < 10) return true;
    if (x >= image.getWidth() - 10) return true;
    if (y >= image.getHeight() - 10) return true;
    return false;
}
```

```
function pixelOnEdge(pixel,image,borderWidth){
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < borderWidth) return true;
    if (y < borderWidth) return true;
    if (x >= image.getWidth() - borderWidth) return true;
    if (y >= image.getHeight() - borderWidth) return true;
    return false;
}
```

Adding a Parameter to a Function

```
function pixelOnEdge(pixel,image){
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < 10) return true;
    if (y < 10) return true;
    if (x >= image.getWidth() - 10) return true;
    if (y >= image.getHeight() - 10) return true;
    return false;
}
```

```
function pixelOnEdge(pixel,image,borderWidth){
    var x = pixel.getX();
    var y = pixel.getY();
    if (x < borderWidth) return true;
    if (y < borderWidth) return true;
    if (x >= image.getWidth() - borderWidth) return true;
    if (y >= image.getHeight() - borderWidth) return true;
    return false;
}
```

Calling `pixelOnEdge` with Parameters

- Why function call changes
  - One more parameter

```
var image = new SimpleImage("lion.jpg");
for (var pixel of image.values()) {
   if (pixelOnEdge(pixel,image,3)){
       pixel = setBlack(pixel);
   }
}
```

```
var image = new SimpleImage("lion.jpg");
for (var pixel of image.values()) {
   if (pixelOnEdge(pixel,image,20)){
       pixel = setBlack(pixel);
   }
}
```

Summary of Functions in JavaScript

- Functions allow us to create new names for blocks of code making it easier to write more complex programs
- Functions have names, parameters, and return values
- Function calls supply arguments, use the value returned to replace call
- Have fun with functions (that's a joke)