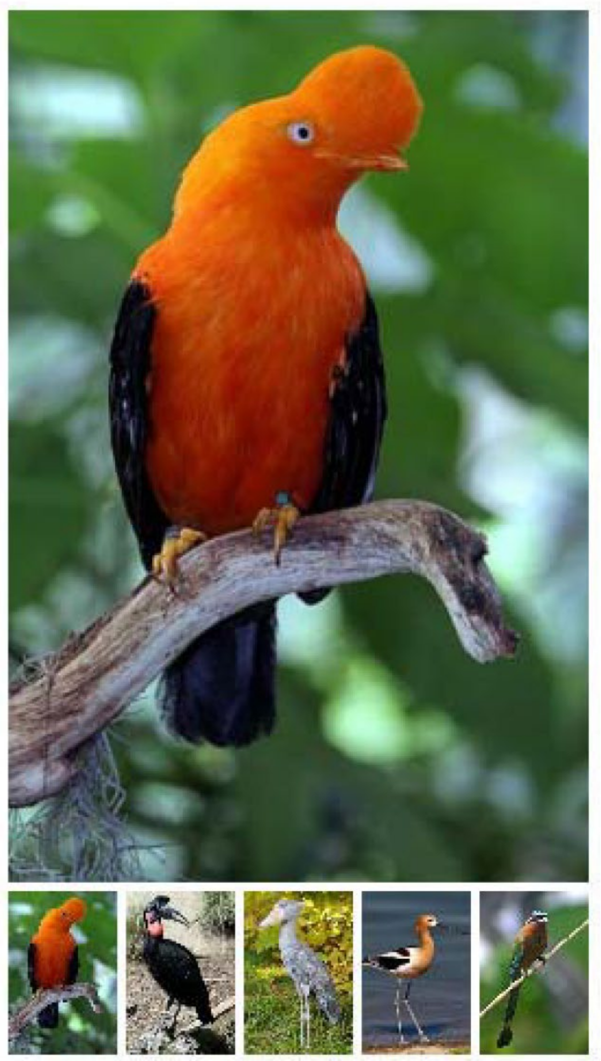
Building a Picture Carousel

Photo thumbnail galleries are a common website feature for showcasing pictures of products. Some galleries use simple CSS hover techniques to enlarge an image on mouseover. Some allow a user to click an image or arrows alongside the image to cycle through the images. The latter are sometimes referred to as carousels because the user can scroll through the images in a circular fashion.

In today's lesson, you'll create just such a photo gallery. In the process, you'll learn about using JavaScript to change the source of any image tag on the page. And you'll learn still more tools and techniques for JavaScript creativity, including global variables and string manipulation.

Creating the Thumbnail Gallery

Let’s start with a simple thumbnail gallery where you have a bunch of small pictures and one large one. Clicking a small picture makes it show as the larger picture. The basic appearance will still be a set of thumbnail images below a larger picture, like this:



Large image and thumbnails
(Click the image to see the gallery in action)

We're using clicking here as a starting point and vehicle to learn some new JavaScript tools and techniques. Later, we can add some buttons to cycle through the images in a carousel fashion. Put the supplied images in **pix3** folder.

1- Start with a html file and add a boilerplate
2- Then add the following code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Carousel</title>
    <link href="css/styles.css" rel="stylesheet">
  </head>
  <body>
    <!-- HTML to display the pictures -->
    <div class="carousel">
      <img src="pix3/birds01.png" alt="" id="bigpic" />
      <div class="thumbs">
        <img src="pix3/birds01.png" alt="" onclick="showbig(this.src)" />
        <img src="pix3/birds02.png" alt="" onclick="showbig(this.src)" />
        <img src="pix3/birds03.png" alt="" onclick="showbig(this.src)" />
        <img src="pix3/birds04.png" alt="" onclick="showbig(this.src)" />
        <img src="pix3/birds05.png" alt="" onclick="showbig(this.src)" />
      </div>
      <!-- End thumbs -->
    </div>
    <!-- End carousel -->
  </body>
</html>
```

And for the style.css add:

```
.carousel {
  text-align: center;
}
#bigpic {
  height: 400px;
}
.thumbs img {
  height: 75px;
}
```

Save your page. If you open the file, you must see the photos and thumbnails. Clicking a picture won't make it the large image, because we haven't yet written the JavaScript code for that.

Clicking a Thumbnail

If you look at the <img> tags in *carousel.htm,* you'll notice that each has an onclick event handler that looks like this:

```
onclick="showbig(this.src)"
```

that bit of code tells the browser, "When the user clicks this picture, call a function named *showbig*, and pass to it the value of the .src property of the image that's making the call." For example, take a look at this tag:

```
<img src="pix3/birds01.png" alt="" onclick="showbig(this.src)" />
```

The keyword *this* in JavaScript always means *this element (or object) that is calling the JavaScript code*. And that will always be the tag that contains the event handler code. In this case, it's the img tag. Like all img tags, this one uses an src= attribute to indicate the source (location and file name) of the image that the tag should show on the page. If you took a wild guess that *this.src* means "the value of the src= attribute of this tag," you'd be right. When you click an image that has that event handler, the onclick event will call a function named *showbig* and pass to it the value of the src= attribute of the tag that's doing the calling.

We don't have a showbig function yet to test it out, so let's add that now.

3- Add a js folder and a file named carousel.js
4- Link the js file to your html by adding the following code to your <head> </head> section

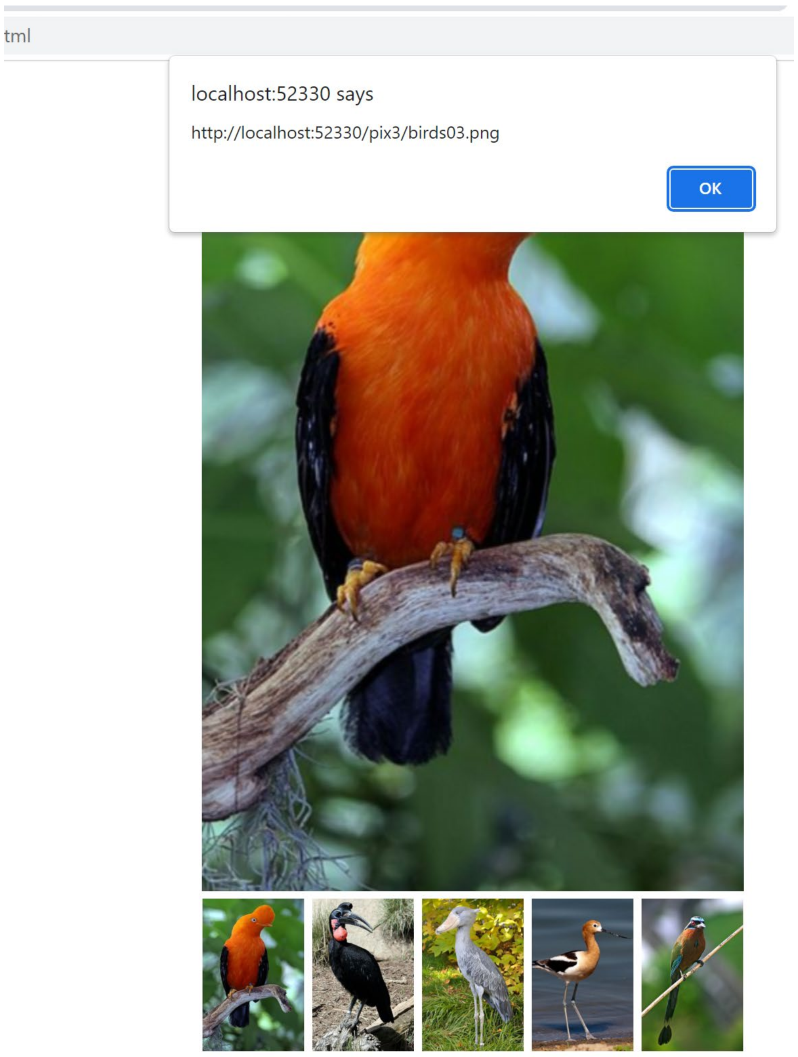
```
<script src="js/carousel.js"></script>
```

5- Add the following code to your carousel.js

```
//Show clicked image in the large img tag
function showbig(pic) {
    alert(pic);
}
```

Notice that the name of the function is **showbig**. As always, **showbig** is just a name. After the function name, we have (**pic**). The **pic** part is also just a variable name that stores whatever value gets passed by the calling tag (**this.src**).

6- Now run the file and see the output.
7- Click on any of the thumbnail and you will see an alert will pop up.


tml

localhost:52330 says

http://localhost:52330/pix3/birds03.png

We just showed the alert box to test the code and see what is passed to the function. In fact, we want it to change the source of the large image to whatever source information is being passed in from the calling tag so that the source (src) of the large image matches the source of the thumbnail image the user clicked. Fortunately for us, that's easy to do because the JavaScript .src property is a Read/Write property. So you can use it both to see what source is currently assigned to an image and to change the source of an existing image.

In the **showbig()** function, we need to do two things. First, we have to tell the browser which image on the page needs to have its source changed. That will be the larger image, which is displayed with this tag in the page:

```
        <img src="pix3/birds01.png" alt="" id="bigpic" />
```

We are going to change the src of this element after clicking on one of the thumbnails, as in the following steps:

8- Delete the **alert** statement and replace it with:

```
        document.getElementById("bigpic").src = pic;
```

Now, open the file again and test the program; click on any thumbnail and if your code is correct, you must see the large image changes to the image of thumbnail you clicked on.

JavaScript Global Variables

Our working example uses five thumbnail images. You could write the code so it's easy to adapt to pages where there might be more or fewer images.

One way is to store in a variable any value that might change from one application to the next. People sometimes refer to such variables as global variables because they're handled a little differently from most variables. To understand how and why requires a little general knowledge about the scope (or lifespan) of JavaScript variables.

Most variables in JavaScript have a very short lifespan. They're created inside some function and are local variables. They're local to the function in which they're created—they exist only while that function is running. As soon as the function finishes, all variables created within that function are erased from memory.

The scope of a variable defines what other code on the page can access the contents of a variable. The scope is similar to the lifespan of a variable because any given variable is visible only to the function in which it's created. In other words, if you define a variable inside a function, other functions on the page can't "see" the variable or access the contents of that variable.

Variables and the values assigned to them are stored in memory (RAM). Having variables exist only while needed is a very efficient way to use memory. However, there may be times when you want to create variables that are accessible to two or more functions in a page. Such variables are often called global variables because they're visible to, and accessible to, all the JavaScript code on the page.

Let's create a global variable named **maximages** to store the number of images in the gallery. Our **carousel.htm** page has five images in its gallery. So in this page, we'll start it off with a value of five. Follow these steps:

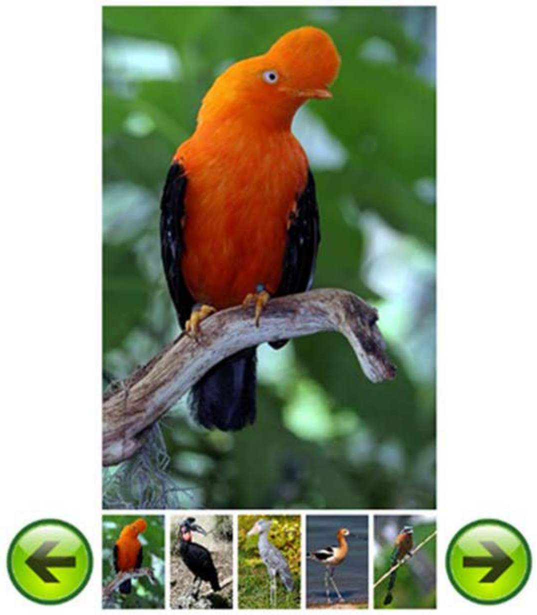
10- On the top of JS file, add the following:

```
var maximages = 5;
```

The rest of the code will be written to assume that **maximages** always reflects the number of images in the gallery. So if you used this code in a thumbnail gallery with, say, 10 images, you'd just have to change **maximages**=5 to **maximages**=10.

Naming the Image Files

As a nice enhancement, we can add buttons to the left and right of the thumbnails so users can click to cycle through the images. Here's an example:



There are a few different ways we could write the code to accomplish this. One of the easiest it to simply name the image files so that we need only change one number in the file name to change the image. That is why in our example images are named **birdsXX** so all image sources including the folder starts with **pix3/birds** . This requires some text manipulation as you will see.

11- For now add just on the top of you JS code:

```
//Folder name and start of file name of each image file.
var startpath = "pix3/birds"
//Filename extension of each image.
var extension = ".png"
```

Adding the Carousel Buttons

12- Add the arrows to the html file just before and after your images:

```
      <div class="thumbs">
        <img src="pix3/LeftArrow.png" alt="Previous" onclick="calcslide(-1)" >
        <img src="pix3/birds01.png" alt="" onclick="showbig(this.src)" />
        <img src="pix3/birds02.png" alt="" onclick="showbig(this.src)" />
        :
        <img src="pix3/RightArrow.png" alt="Next" onclick="calcslide(1)">
      </div>
```

Now test your program.

String Manipulation

As mentioned before, we intentionally used files named **birdXX.png** (where the XX is two-digit number) for our images. This approach allows our code to switch from one image to the next using a method known as string manipulation.

For example, if the user is currently viewing bird01.png and clicks the right arrow, all we have to do is add 1 to the number inside the filename and show bird02.png. And if the user is viewing bird02.png and clicks the left arrow to go to the previous image, we just subtract 1 from the number in the filename and choose bird01.png.

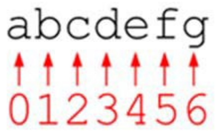
We can see the number hidden inside each filename like bird01.png, bird02.png, bird03.png, and so forth. And we can easily add or subtract 1 from the number hidden inside any filename. We can implement this by coding.

Finding One String in Another

JavaScript has an **indexOf** method, which can be used for finding the starting point of a small string contained within a larger string. The syntax is:

largestring.indexOf(smallstring)

In practice, you'd replace **largestring** with some string of text or the name of a variable that contains a string of text. The **smallstring** is also a string enclosed in quotation marks or the name of a variable that contains a string. The value returned by the **indexOf** method is -1 if **smallstring** doesn't exist in the larger string at all. Otherwise, it's a positive number indicating the location of the smaller string within the larger string. The location is zero-based, which means JavaScript always starts counting with zero rather than 1, like this:



That zero-based counting is counterintuitive and takes some time to get used to. Here are some examples of values returned by the indexOf method, which illustrate how it works:

Example	Returns
"abcdefg".indexOf("a")	0
"abcdefg".indexOf("b")	1
"abcdefg".indexOf("cd")	2
"abcdefg".indexOf("z")	-1
"abcdefg".indexOf("moose")	-1
"bird01.png".indexOf(".png")	6

Getting a Portion of a String

JavaScript also includes some methods for extracting a portion of a string. The smaller string that you extract is sometimes called a substring. One of the methods you can use to extract a substring uses this syntax:

string.substr(startposition,length)

In that syntax, string is the larger string from which you're be extracting (or the name of a variable that contains the string). In your code, replace **startposition** with a number indicating the starting position of the substring (where 0 is the first character). Replace **length** with the number of characters to extract beyond that. You can omit the **.length** argument. If you do, the value returned is a substring starting at startposition extending to the end of the string. Here are some examples:

Example	Returns
"abcdefg".substr(0,1)	a
"abcdefg".substr(0, 3)	abc
"abcdefg".substr(2, 4)	cdef
"abcdefg".substr(6, 1)	g
"abcdefg".substr(4)	efg
"bird01.png".substr(4,2)	01
"bird01.png".substr(6)	.png

Another handy string manipulation method served up by JavaScript is the slice() method. This one is often used to slice off the first or last characters from a string using this syntax:

string.slice(start)

As with all string methods, the string part can be any string, or the name of a variable that contains a string. The start part is a number (or the name of a variable that contains a number) indicating where to start the slice. As always, the first character in the string is at number 0, not number 1. If you specify a negative number for slice, it slices from the end of the string toward the start of the string however many characters you specify. Here are some examples:

Example	Returns
"abcdefg".slice(1)	bcdefg
"abcdefg".slice(4)	efg
"abcdefg".slice(6)	g
"abcdefg".slice(-1)	g
"abcdefg".slice(-2)	fg
"bird09.png".slice(-4)	.png

Converting Strings to Numbers

A filename like “bird01.png” is just a string, and JavaScript can't see the number inside. If you wanted to do some math on that number hidden in there, you'd first have to extract the numeric digits as a substring and convert it to a number.

JavaScript offers two methods for converting strings to numbers:

- **parseInt(string)**: Returns the integer portion of string as a number (if possible).
- **parseFloat(string)**: Returns the floating point portion of a string as a number (if possible).

The difference between an integer and a floating point is that an integer is always a whole number with no decimal point, such as 1 or -1 or 0 or 100 or 1000 or 18 or whatever. A floating point number can have a decimal portion, such as 5.5 or 1.23 or 0.5 or 100.11 or 1234.56.

The string can only be converted to a number if the string starts with a numeric digit (0-9), a hyphen (minus sign), or a dot (decimal point) and it doesn't contain any other non number characters. If the string to convert can't be converted to a number, then the function returns NaN, which means "not a number." Here are some examples:

Example	Returns
parseInt("01")	1
parseInt("0.9")	0
parseInt("-1")	-1
parseInt("123.45")	123
parseInt("abcdefg")	NaN
parseFloat("01")	1
parseFloat("0.9")	0.9
parseFloat("-1")	-1
parseFloat("123.45")	123.45
parseFloat("abcdefg")	NaN

Converting Numbers to String

Sometimes you may need to convert a number back to a string. And the way that's usually done in JavaScript is to simply concatenate, with an = sign, the number to some string. It can even be an empty string—for example, if x is a variable that contains the number 10 and you execute this line of code:

var y = "" + 10

. . .the variable y ends up containing "10" as a string rather than a number value.

The string can be anything. It can even be a numeric digit in quotes. For example, if x contains 1 and you execute this:

var x = 1

var z = "0" + x

then z ends up containing the string "01". If you then concatenate that to a couple other strings, like this:

var path="pix3/birds" + z + ".png"

you end up with an even longer string that contains **pix3/birds01.png**.

So, now let's combine all these techniques to complete our program.

Recall that the left arrow image and right arrow image use **onclick="calcslide(-1)"** and **onclick="calcslide(1)"**, respectively, as event handlers . So we'll need a JavaScript function named **calcslide** that can accept the incoming value in the parentheses (-1 or 1).

Inside that function, we need to calculate the next image to show by adding that positive or negative 1 to the number of the image that's currently showing.

13- Add the following code (meanwhile check the comments and make sure you understand each statement) below **var extension = '.png'**

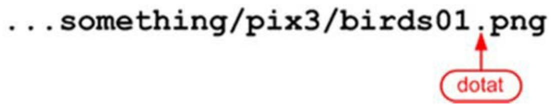
```
var extension = '.png';
//Show clicked image in the large img tag

//Calculates which picture to show next based on x
//which is either 1 or -1.
function calcslide(x) {
  //Get file name of image that's showing.
  var currentimage = document.getElementById('bigpic').src;
  var path = 'pix3/birds' + z + '.png';
  var z = '0' + x;
  var y = '' + 10;
  //Locate file name extension in current image source string.
  var dotat = currentimage.indexOf(extension);
  //Grab two digits to the left of that file name extension.
  var stringnumber = currentimage.substr(dotat - 2, 2);
  //Convert stringnumber string to number and add x.
  var nextnum = parseInt(stringnumber) + x;
  //If nextnum is less than 1, wrap around to maximages.
  if (nextnum < 1) {
    nextnum = maximages;
  }
  //If nextnum is greater than maximages, wrap around to 1.
  if (nextnum > maximages) {
    nextnum = 1;
  }
  //Create two-digit string from number (leading zero if less than 10).
  var twodigitnum = ('0' + nextnum).slice(-2);
  //Create new file name from two-digit number string.
  var showimg = startpath + twodigitnum + extension;
  showbig(showimg);
}
```

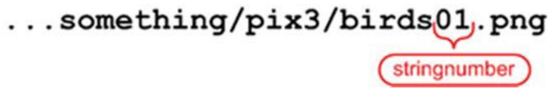
Now program is complete.

save the program and test it.

var dotat = currentimage.indexOf(extension);



var stringnumber = currentimage.substr(dotat - 2, 2);


...something/pix3/birds01.png
stringnumber