

Introduction to JavaScript Training

JavaScript Functions

Lesson 1, Activity 2: Global Functions

JavaScript has a number of global functions. We will examine some of them in this section.

Number(object)

The `Number()` function takes one argument: an object, which it attempts to convert to a number. If it cannot, it returns `NaN`, for "Not a Number."

Code Sample:

JavaScriptFunctions/Demos/Number.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Number() Function</title>
<link href="style.css" rel="stylesheet" type="text/css">
<script type="text/javascript">
  var strNum1 = "1";
  var strNum2 = "2";
  var strSum = strNum1 + strNum2; //returns 12
  alert(strSum);

  var intNum1 = Number(strNum1);
  var intNum2 = Number(strNum2);
  var intSum = intNum1 + intNum2; //returns 3
  alert(intSum);
</script>
</head>
<body>
  <p>Nothing to show here.</p>
</body>
</html>
```

Because `strNum1` and `strNum2` are both strings, the `+` operator

concatenates them, resulting in "12".

```
var strNum1 = "1";
var strNum2 = "2";
var strSum = strNum1 + strNum2; //returns 12
alert(strSum);
```

After the `Number()` function has been used to convert the strings to numbers, the `+` operator performs addition, resulting in 3.

```
var intNum1 = Number(strNum1);
var intNum2 = Number(strNum2);
var intSum = intNum1 + intNum2; //returns 3
alert(intSum);
```

String(object)

The `String()` function takes one argument: an object, which it converts to a string.

Code Sample:

JavaScriptFunctions/Demos/String.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>String() Function</title>
<link href="style.css" rel="stylesheet" type="text/css">
<script type="text/javascript">
  var intNum1 = 1;
  var intNum2 = 2;
  var intSum = intNum1 + intNum2; //returns 3
  alert(intSum);

  var strNum1 = String(intNum1);
  var strNum2 = String(intNum2);
  var strSum = strNum1 + strNum2; //returns 12
  alert(strSum);
```

```

</script>
</head>
<body>
  <p>Nothing to show here.</p>
</body>
</html>

```

Because `intNum1` and `intNum2` are both numbers, the `+` operator performs addition, resulting in 3:

```

var intNum1 = 1;
var intNum2 = 2;
var intSum = intNum1 + intNum2; //returns 3
alert(intSum);

```

After the `String()` function has been used to convert the numbers to string, the `+` operator performs concatenation, resulting in "12":

```

var strNum1 = String(intNum1);
var strNum2 = String(intNum2);
var strSum = strNum1 + strNum2; //returns 12
alert(strSum);

```

isNaN(object)

The `isNaN()` function takes one argument: an object. The function checks if the object is *not* a number (or cannot be converted to a number). It returns `true` if the object is not a number and `false` if it is a number.

Code Sample:

JavaScriptFunctions/Demos/isNaN.html

```

---- C O D E   O M I T T E D ----

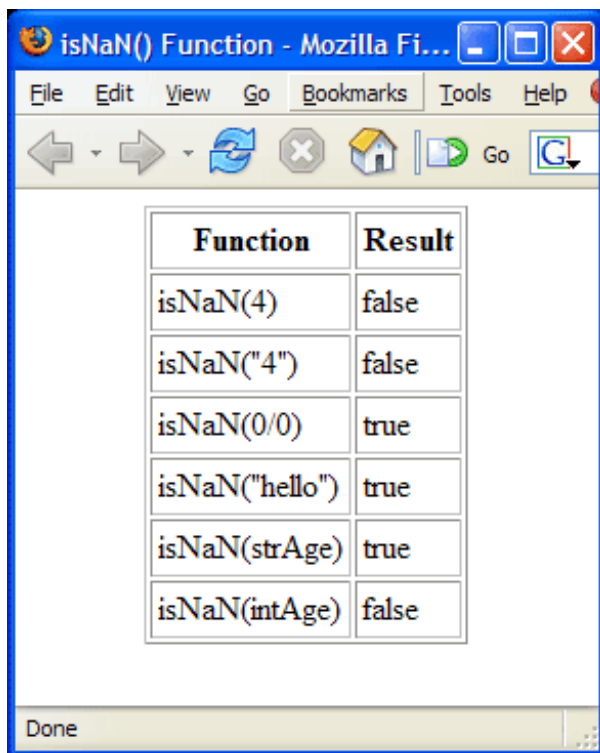
```

```

<table>
<tr>
  <th>Function</th><th>Result</th>
</tr>
<script type="text/javascript">
  document.write("<tr><td>isNaN(4)</td>");
  document.write("<td>" + isNaN(4) + "</td></tr>");
  document.write("<tr><td>isNaN(\"4\")</td>");
  document.write("<td>" + isNaN("4") + "</td></tr>");
  document.write("<tr><td>isNaN(0/0)</td>");
  document.write("<td>" + isNaN(0/0) + "</td></tr>");
  document.write("<tr><td>isNaN(\"hello\")</td>");
  document.write("<td>" + isNaN("hello") + "</td></tr>");
  var strAge = "twelve";
  document.write("<tr><td>isNaN(strAge)</td>");
  document.write("<td>" + isNaN(strAge) + "</td></tr>");
  var intAge = 12;
  document.write("<tr><td>isNaN(intAge)</td>");
  document.write("<td>" + isNaN(intAge) + "</td></tr>");
</script>
</table>
---- C O D E   O M I T T E D ----

```

The output will look like this:



Function	Result
isNaN(4)	false
isNaN("4")	false
isNaN(0/0)	true
isNaN("hello")	true
isNaN(strAge)	true
isNaN(intAge)	false

parseFloat() and parseInt()

The `parseFloat()` function takes one argument: a string. If the string begins with a number, the function reads through the string until it finds the end of the number, hacks off the remainder of the string, and returns the result. If the string does not begin with a number, the function returns `NaN`.

The `parseInt()` function also takes one argument: a string. If the string begins with an integer, the function reads through the string until it finds the end of the integer, hacks off the remainder of the string, and returns the result. If the string does not begin with an integer, the function returns `NaN`.

Code Sample:

JavaScriptFunctions/Demos/ParsingNumbers.html

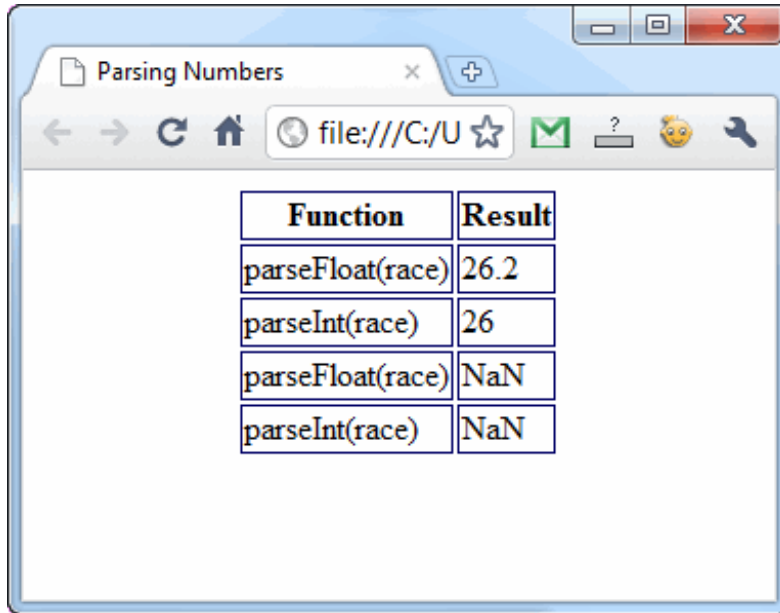
```

---- C O D E   O M I T T E D ----

<table>
<tr>
  <th>Function</th><th>Result</th>
</tr>
<script type="text/javascript">
  var race = "26.2 miles";
  document.write("<tr><td>parseFloat(race)</td>");
  document.write("<td>" + parseFloat(race) + "</td></tr>");
  document.write("<tr><td>parseInt(race)</td>");
  document.write("<td>" + parseInt(race) + "</td></tr>");
  race = "Marathon";
  document.write("<tr><td>parseFloat(race)</td>");
  document.write("<td>" + parseFloat(race) + "</td></tr>");
  document.write("<tr><td>parseInt(race)</td>");
  document.write("<td>" + parseInt(race) + "</td></tr>");
</script>
</table>
---- C O D E   O M I T T E D ----

```

The output will look like this:



A screenshot of a web browser window titled "Parsing Numbers". The address bar shows a file path: `file:///C:/U`. The main content area displays a table with two columns: "Function" and "Result". The table contains four rows of data.

Function	Result
<code>parseFloat(race)</code>	26.2
<code>parseInt(race)</code>	26
<code>parseFloat(race)</code>	NaN
<code>parseInt(race)</code>	NaN

These "global" functions we have discussed above are actually methods of the `window` object, but as `window` is assumed if no object is referenced, we don't need to explicitly write `window.parseFloat()` or `window.isNaN()`. Also, some of these functions such as `Number()` and `String()` are really function constructors for creating new `String` and `Number` objects. For now, just remember that you can use these functions to ensure you are working with a `String` or `Number`.

Lesson 1, Activity 4: Working with Global Functions

Duration: 10 to 15 minutes.

In this exercise, you will practice working with JavaScript's global functions.

1. Open [JavaScriptFunctions/Exercises/BuiltinFunctions.html](#) for editing.
2. Modify the file so that it outputs the sum of the two numbers entered by the user.

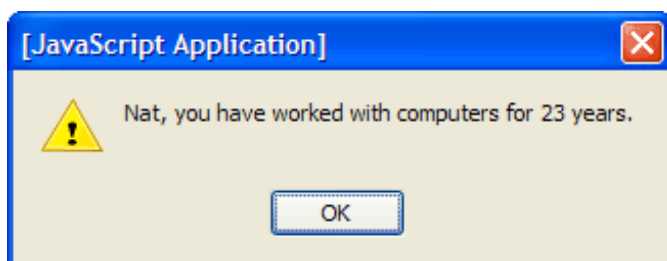
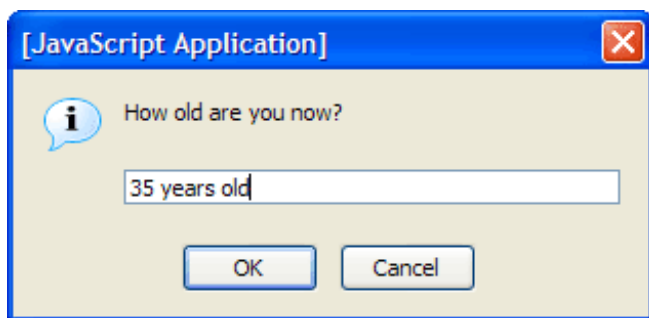
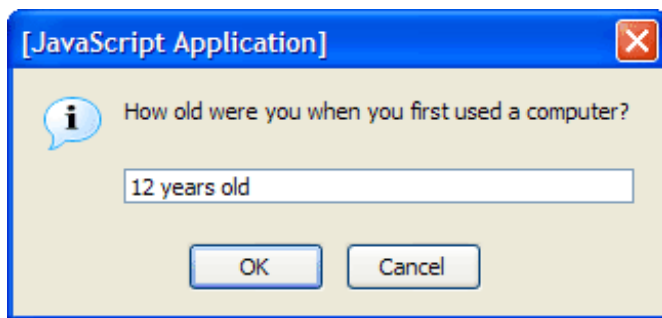
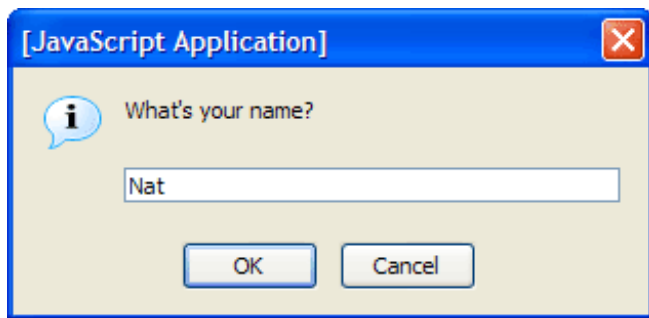
Code Sample:

[JavaScriptFunctions/Exercises/BuiltinFunctions.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>JavaScript Built-in Functions</title>
<link href="style.css" rel="stylesheet" type="text/css">
<script type="text/javascript">
  var userNum1, userNum2, numsAdded;
  userNum1 = window.prompt("Choose a number.", "");
  alert("You chose " + userNum1);
  userNum2 = window.prompt("Choose another number.", "");
  alert("You chose " + userNum2);
  numsAdded = userNum1 + userNum2;
</script>
</head>
<body>
<p>
  <script type="text/javascript">
    document.write(userNum1 + " + " + userNum2 + " = ");
    document.write(numsAdded + "<br/>");
  </script>
</p>
</body>
</html>
```

Challenge

Create a new HTML file that prompts the user for his name, the age at which he first worked on a computer, and his current age. After gathering this information, pop up an alert that tells the user how many years he's been working on a computer. The images below show the steps:



Notice that the program is able to deal with numbers followed by

strings (e.g, "12 years old").

Solution:

JavaScriptFunctions/Solutions/BuiltinFunctions.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>JavaScript Built-in Functions</title>
<link href="style.css" rel="stylesheet" type="text/css">
<script type="text/javascript">
  var userNum1, userNum1, numsAdded;
  userNum1 = window.prompt("Choose a number.", "");
  alert("You chose " + userNum1);
  userNum2 = window.prompt("Choose another number.", "");
  alert("You chose " + userNum2);
  numsAdded = Number(userNum1) + Number(userNum2);
</script>
</head>
<body>
<p>
  <script type="text/javascript">
    document.write(userNum1 + " + " + userNum2 + " = ");
    document.write(numsAdded + "<br/>");
  </script>
</p>
</body>
</html>
```

Challenge Solution:

JavaScriptFunctions/Solutions/BuiltInFunctions-challenge.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>JavaScript Built-in Functions</title>
<link href="style.css" rel="stylesheet" type="text/css">
<script type="text/javascript">
  var userName = prompt("What's your name?", "");
  var age1 = prompt("How old were you when you first used a computer?", "");
  var age2 = prompt("How old are you now?", "");
  var diff = parseFloat(age2) - parseFloat(age1);
```

```
    alert(userName + ", you have used computers for " + diff + " years.");  
</script>  
</head>  
<body>  
    <p>Nothing to show here.</p>  
</body>  
</html>
```

Lesson 1, Activity 5: User-defined Functions

Writing functions makes it possible to reuse code for common tasks. Functions can also be used to hide complex code. For example, an experienced developer can write a function for performing a complicated task. Other developers do not need to know how that function works; they only need to know how to call it.

Function Syntax

JavaScript functions generally appear in the head of the page or in external JavaScript files. A function is written using the `function` keyword followed by the name of the function.

Syntax

```
function doSomething() {  
    //function statements go here  
}
```

As you can see, the body of the function is contained within curly brackets (`{ }`). The following example demonstrates the use of simple functions:

Code Sample:

JavaScriptFunctions/Demos/SimpleFunctions.html

```
<!DOCTYPE HTML>  
<html>  
<head>  
<meta charset="UTF-8">  
<title>JavaScript Simple Functions</title>  
<link href="style.css" rel="stylesheet" type="text/css">  
<script type="text/javascript">  
    function changeBgRed() {
```

```

    document.bgColor = "red";
}

function changeBgWhite() {
    document.bgColor = "white";
}
</script>
</head>
<body>
<p>
    <span onclick="changeBgRed();">Red</span> |
    <span onclick="changeBgWhite();">White</span>
</p>
</body>
</html>

```

Passing Values to Functions

The functions above aren't very useful because they always do the same thing. Every time we wanted to add another color, we would have to write another function. Also, if we want to modify the behavior, we will have to do it in each function. The example below shows how to create a single function to handle changing the background color.

Code Sample:

JavaScriptFunctions/Demos/PassingValues.html

```

<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Passing Values</title>
<link href="style.css" rel="stylesheet" type="text/css">
<script type="text/javascript">
    function changeBg(color) {
        document.bgColor = color;
    }
</script>
</head>
<body>
<p>
    <span onclick="changeBg('red');">Red</span> |
    <span onclick="changeBg('white');">White</span>

```

```

</p>
</body>
</html>

```

As you can see, when calling the `changeBg()` function, we pass a value (e.g, 'red'), which is assigned to the `color` variable. We can then refer to the `color` variable throughout the function. Variables created in this way are called function arguments or parameters. A function can have any number of arguments, separated by commas.

A Note on Variable Scope

Variables created through function arguments or declared within a function with `var` are local to the function, meaning that they cannot be accessed outside of the function.

Variables declared with `var` outside of a function and variables that are used without being declared are global, meaning that they can be used anywhere on the page.

Returning Values from Functions

The `return` keyword is used to return values from functions as the following example illustrates:

Code Sample:

[JavaScriptFunctions/Demos/ReturnValue.html](#)

```

---- C O D E   O M I T T E D ----

<script type="text/javascript">
function setBgColor(){
    document.bgColor = prompt("Set Background Color:", "");
}

```

```
function getBgColor() {  
    return document.bgColor;  
}  
</script>  
</head>  
<body>  
<form>  
    <input type="button" value="Set Background Color"  
        onclick="setBgColor();">  
    <input type="button" value="Get Background Color"  
        onclick="alert(getBgColor());">  
</form>  
</body>  
</html>
```

When the user clicks on the "Get Background Color" button, an alert pops up with a value returned from the `getBgColor()` function. This is a very simple example. Generally, functions that return values are a bit more involved. We'll see many more functions that return values throughout the course.

Lesson 1, Activity 7: Writing a JavaScript Function

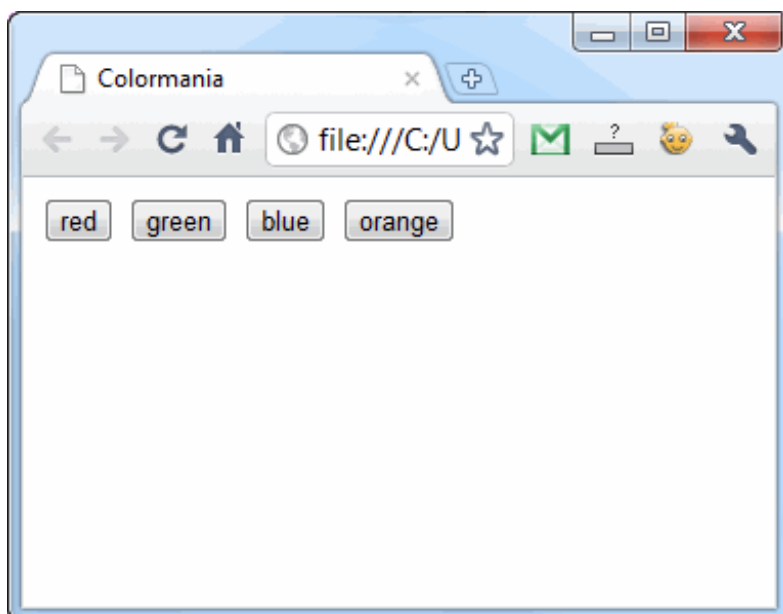
Duration: 15 to 25 minutes.

In this exercise, you will modify a page called [ColorMania.html](#), which will contain a form with four buttons. Each button will show the name of a color (e.g, red) and, when clicked, call a function that changes the background color. The buttons you will create will be of type `button`. For example,

```
<input type="button" value="red" onclick="functionCall();">
```

1. Open [JavaScriptFunctions/Exercises/ColorMania.html](#) for editing.
2. Write code to prompt the user for her name.
3. Write a function called `changeBg()` that changes the background color and then pops up an alert telling the user, by name, what the new background color is.
4. In the form, add four buttons that, when clicked, call the `changeBg()` function and pass it a color value.

The resulting page should look like this:



Code Sample:

JavaScriptFunctions/Exercises/ColorMania.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Colormania</title>
<script type="text/javascript">
  //PROMPT USER FOR NAME

  /*
  Write a function called changeBg() that changes the background
  color and then pops up an alert telling the user, by name, what
  the new background color is.
  */
</script>
</head>
<body>
<form>
  <!--ADD BUTTONS HERE-->
</form>
</body>
</html>
```

Challenge

Add another button called "custom" that, when clicked, prompts the user for a color, then changes the background color to the user-entered color and alerts the user to the change.

Solution:

JavaScriptFunctions/Solutions/ColorMania.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Colormania</title>
<script type="text/javascript">
```

```

var userName = prompt("Your name?", "");

function changeBg(color){
    document.bgColor = color;
    alert(userName + ", the background color is " + color + ".");
}
</script>
</head>
<body>
<form>
    <input type="button" value="red" onclick="changeBg('red');">
    <input type="button" value="green" onclick="changeBg('green');">
    <input type="button" value="blue" onclick="changeBg('blue');">
    <input type="button" value="orange" onclick="changeBg('orange');">
</form>
</body>
</html>

```

Challenge Solution:

JavaScriptFunctions/Solutions/ColorMania-challenge.html

```

<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Colormania</title>
<script type="text/javascript">
    var userName = prompt("Your name?", "");

    function changeBg(color){
        document.bgColor = color;
        alert(userName + ", the background color is " + color + ".");
    }

    function customBg(){
        var userColor = prompt("Choose a color:", "");
        changeBg(userColor);
    }
</script>
</head>
<body>
<form>
    <input type="button" value="red" onclick="changeBg('red');">
    <input type="button" value="green" onclick="changeBg('green');">
    <input type="button" value="blue" onclick="changeBg('blue');">
    <input type="button" value="orange" onclick="changeBg('orange');">
    <input type="button" value="custom" onclick="customBg();">
</form>

```

```
</body>  
</html>
```