JavaScript Primer

To learn more JavaScript, please check out these links:

1. https://www.learn-js.org/
2. https://www.w3schools.com/js/
3. https://learnjavascript.online/app.html
4. https://javascript.info/

While we don't have enough space here to cover JavaScript completely, here is a quick overview for those familiar with other programming languages to get started. To experiment with JavaScript, you can use the JavaScript console supplied with your favorite browser or startup node.js to run the Read-Eval-Print-Loop (REPL) on your machine.

Variables

Variables are declared as follows:

```
var myFirstVariable;
```

Declarations can be separated by commas on the same line:

```
var myFirstVariable, mySecondVariable;
```

They can be assigned with some initial values using the equals operators.

```
var myFirstVariable = 0;
```

Types

JavaScript supports various types as follows:

```
var anInt = 0;
var aFloat = 2.5;
var aBoolean = true;
var aString = "This is a string";
var anArray = [1,2,3];
var anObjectLiteral = {
    someField : "Some value"
};
var aFunction=function() { return; }; // function
```

```
var re = /ab+c/;          // regular expression
```

You can instantiate an object using the new operator like this:

```
var aDate = new Date();
```

Comments

Comments can use the double slash single line or the C-style multiline comment format:

```
// this is a single line comment
/*
   This is a multi-line comment
*/
```

Basic Operators

The built-in numerical operators include the usual: + (addition), – (subtraction), / (division), * (multiplication) and % (modulo).

```
var a = 6;
a = a + 1; // a is 7
b = b % 2; // b is 3
a++; // a is 8
a–; //a is 7
a += 3; // a is 10
```

Strings can be concatenated as shown:

```
var aString = “The value of a is ” + a;
var greeting = “What a very “;
greeting += “nice day”;
```

The typeof operator returns the type of a variable.

```
typeof aString; // returns “string”
```

To determine the class of an object, for example, to distinguish between arrays and other objects, the Object.toString method can be used as follows:

```
Object.prototype.toString.call(new Date()); // “[object Date]”
Object.prototype.toString.call([1,2,3]); // “[object Array]”
```

```
Object.prototype.toString.call({someField:12}); //"[object Object]"
Object.prototype.toString.call(function(){return;}); //"[object Function]"
```

Conditionals

You can use the comparison operators, < (less than) > (greater than) <= (less than or equal) >= (greater than or equal) == (equals) != (not equals) in if statements as follows:

```
var a = 1;
var b = null;
if (a > 0) {
  b = "a is greater than zero";
} else {
  b = "a is zero or less";
}
```

To shorten this, one could use the conditional operator as follows:

```
var b = (a > 0) ? "a is greater than zero" : "a is zero or less";
```

Loops

Javascript supports for and while loops:

```
for (var i = 0; i < anArray.length; i++) {
    console.log(myArray[i]);
}
while (x < 10) {
    console.log(x++);
}
```

Functions

Functions are created like this:

```
function addOne (x) {
    return x + 1;
}
```

To call a function, you can call them as follows:

```
var y = addOne(5);          // y = 6
```

Or, if the function is a method of an object, for example, a Date object:

```
var d = new Date();
var year = d.getMonth();    // returns 0 for January, 1 for
February,
```

Callbacks

Node.js is an event-based framework with only one thread of execution. To allow more than one task to execute while performing I/O or waiting for a timer to expire, callbacks are used extensively. For example, to log 'done' to the console after 2 seconds, you can use an anonymous function callback as follows:

```
setTimeout(function() { console.log('done'); }, 2000);
```

You can use functions to process an array:

```
var friends = ["mike","roberto", "rodger", "ted", "daniel"];
friends.forEach(function (eachName, index){
  console.log(index + 1 + ". " + eachName); // 1. mike  2.
roberto  3. rodger  4. ted  5. daniel
});
```

Exception Handling

Exception handling can be used to catch errors in your code. To do so, wrap your code in a try/catch block as shown.

```
try {
    // try to do something
} catch (e) {
    // handle errors
} finally {
    // this block is always executed regardless of whether there
was an exception
}
```

That's the end of this JavaScript overview. Those new to JavaScript should now have enough knowledge to get going. If you feel you don't understand the JavaScript, it's worthwhile taking a look at one of the many JavaScript primers mentioned at the start of this primer.