JavaScript

Amit Fegade
amit.fegade@capgemini.com

Capgemini

Table of Content

What you will learn

- Go from a total beginner to an advanced JavaScript Developer
- JavaScript and Programming fundamentals: Variables, Data types, Loops, Selection statements, Arrays, functions.
- Object-oriented JavaScript
- Asynchronous JavaScript: The event loop, promises, async/await, AJAX and APIs
- ES5 and ES6, Implementing various concepts like closures, callbacks, namespaces, and the module pattern to secure their applications, You will be make use JSON to transmit and store data.
- Prototypal inheritance, closures, currying, __proto__
- Organize and structure your code using JavaScript pattern like Modules.
- Implement exception handling, Manage Ajax requests.
- Implement regular expressions to validate data andManipulate Document Object Model (DOM) elements.
- You will be able to write JavaScript code using established coding standards and best practices, Create custom JavaScript object classes.

Pre-requisite for the course

- Should be familiar with HTML and CSS
- Basic Programming fundamentals

Software Requirements

- Google Chrome / Mozilla Firefox
- Visual Studio Code

Basic Topics (Day 1)

- Introduction to JavaScript
- Variables, operators, Data types
- Conditional statements and Loops
- Functions, Function expressions and arrows.
- “use strict” mode
- Objects, Symbol type, Constructor, operator new
- Object methods, this, Garbage Collection
- Methods of primitives, Arrays, Strings, Iterables
- JSON methods, var, let and const
- Object Property flags and descriptors
- Property getters and setters
- Classes, private & protected properties & methods, class Inheritance
- Instanceof, static properties and methods, Mixins

Advanced Topics (Day 2)

- Closures, Currying, call() and apply() functions
- Prototypal inheritance, Native prototypes
- F.prototype, prototype methods, __proto__
- Error handling try..catch, custom errors
- Introduction to callbacks, Promises
- Async/await, Microtasks and event loops
- Error handling with promises
- Promises chaining
- Generators, Async iterators
- Modules, export and import modules
- Dynamic imports

Reference Material

MDN Web Docs – JavaScript

https://developer.mozilla.org/en-US/docs/Web/JavaScript

Useful Resources

1) JavaScript video course by JavaBrains

https://www.youtube.com/playlist?list=PLqq-6Pq4lTTYFJxC9NLJ7dSTI5Z1WWB6K

https://www.youtube.com/playlist?list=PLqq-6Pq4lTTaflXUL0v3TSm86nodn0c_u

https://www.youtube.com/watch?v=O312eN5J2bc&list=PLqq-6Pq4lTTZ_LyvzfrndUOkIvOF4y-_c

2) Websites for JavaScript tutorials

a. https://javascript.info/

b. https://hackr.io/tutorials/learn-javascript

c. https://frontendmasters.com/books/javascript-enlightenment/#1

d. https://github.com/getify/You-Dont-Know-JS

e. https://www.theodinproject.com/courses/javascript

Useful Resources

JavaScript Books

1) Eloquent JavaScript: A Modern Introduction to Programming
2) JavaScript in Action
3) JavaScript: The Definitive Guide, by David Flanagan
4) Learn JavaScript VISUALLY, by Ivelin Demirov
5) JavaScript: The Good Parts

Assignments

Exercises: Basic

For all of these exercises, use the "more powerful practice" approach from the first lecture, where you load an HTML file that has a script tag pointing at a JavaScript file, test the function calls in Firebug or Chrome, and reload the HTML page each time you change the function definitions.

1. Make a function that returns "even" or "odd" depending on the number passed to it.

```
parity(1); --> "odd"
parity(2); --> "even"
```

2. The notes showed a simple version of max that took exactly two arguments. Update this to take exactly three arguments. Note that the builtin version of Math.max takes any number of arguments, which is much better, but we don't know how to do variable arguments yet. And, of course, no using Math.max on this exercise.

```
max(1, 2, 3); --> 3
max(1, 3, 2); --> 3
max(3, 2, 1); --> 3
```

3. Copy the flipCoin function from the last set of exercises. Now, make a function that, given a number, flips a coin that many times and returns the number of heads.

```
numHeads(10); --> 4
numHeads(10); --> 6
numHeads(10); --> 6
```

4. Make a function that takes a number of flips and returns the fraction that were heads.

```
headsRatio(10); --> 0.7
headsRatio(10); --> 0.4
headsRatio(10000); --> 0.5023
headsRatio(10000000); --> 0.4999948
```

5. Make a function that takes a number and a short string, and returns the string concatenated that number of times.

```
padChars(5, "x"); --> "xxxxx"
padChars(7, "-"); --> "-------"
```

6. Write a function that returns the number of times you have to roll a die to get a 6. (Minor hint: compare Math.random() to 5/6).

```
numRollsToGetSix(); --> 13
numRollsToGetSix(); --> 2
```

7. Update the HTML page so that each time you reload it, you randomly see either a "Have a GOOD day!" or "Have a BAD day!" message. If you know some CSS already, make the font big.

Exercises: JavaScript Objects

1. Make a Rectangle class that stores a width and a height. Make a few instances and print out the properties. Modify a few of the properties and print out the results again.
2. Add a getArea method. Use the prototype property.
3. Assuming that the Rectangle constructor takes a width and a height, why does the following output 20 instead of 200? (Hint: if you see an answer that seems too obvious to be what I am looking for, it probably *is* the answer I am looking for.)

```
Rectangle r = new Rectangle(4, 5);
r.hieght = 50;
r.getArea(); --> 20 // Not 200
```

4. Make a variable whose value is an object with firstName and lastName properties, but don't define a Person class first. Try looking up the first and last names. Try changing the last name. It seems very odd to Java programmers to make an object without first defining a class, but JavaScript programmers do this sort of thing all the time.
5. Try reading the middleName property from your variable above. Try assigning to the middleName property. Try reading the property again after you assign to it. Is this behavior a good thing or a bad thing?
6. Create a string that contains what looks like an object with firstName and lastName properties. Use "eval" to turn it into a real object, and test it the same way you did with the previous object that you created directly.
7. Do the same with JSON.parse. You have to follow strict JSON rules in this case.

Functional Programming: Basic Exercises

Even these supposedly basic exercises are a bit tricky if you have never seen functional programming.

1. Make a function called composedValue that takes two functions f1 and f2 and a value and returns f1(f2(value)), i.e., the first function called on the result of the second function called on the value.

```
function square(x) { return(x*x); }
function double(x) { return(x*2); }
composedValue(square, double, 5); --> 100 // I.e., square(double(5))
```

2. Make a function called compose that takes two functions f1 and f2 and returns a new function that, when called on a value, will return f1(f2(value)). Assume that f1 and f2 each take exactly one argument.

```
var f1 = compose(square, double);
f1(5); --> 100
f1(10); --> 400
var f2 = compose(double, square);
f2(5); --> 50
f2(10); --> 200
```

3. Make a function called "find" that takes an array and a test function, and returns the first element of the array that "passes" (returns non-false for) the test. Don't use map, filter, or reduce.

```
function isEven(num) { return(num%2 == 0); }
isEven(3) --> false
isEven(4) --> true
find([1, 3, 5, 4, 2], isEven); --> 4
```

4. Recent JavaScript versions added the "map" method of arrays, as we saw in the notes and used in the previous set of exercises. But, in earlier JavaScript versions, you had to write it yourself. Make a function called "map" that takes an array and a function, and returns a new array that is the result of calling the function on each element of the input array. Don't use map, filter, or reduce.

```
map([1, 2, 3, 4, 5], square); --> [1, 4, 9, 16, 25]
map([1, 4, 9, 16, 25], Math.sqrt); --> [1, 2, 3, 4, 5]
```

Hint: remember the push method of arrays.

Assignments

JavaScript

1. Write a web page with a javascript to write, "Welcome to my very special page" with a line break between my and very.
2. Write a web page with a javascript that writes a paragraph with an image inside of it (so that an image shows up on your page).
3. Write a web page that uses the prompt to get the user to input their first name and then their year of birth. Calculate how old they are and write it to the web page (assume everyone was born on the first day of January of this year)
4. Write a web page that uses a prompt to ask the user what color they'd like the background of their paragraphs to be. It should then write a paragraph with the background set to the appropriate color. (Where this could head – currently I'm designing an interface for students with low vision. While students who are completely blind don't care about font colors and background colors, some students with low vision prefer a black background with white, or even slightly gray text (especially if their visual difficulties are related to visual seizure-type disorders), whereas other low vision students prefer as much contrast as possible, but a white background with black text. This is just one relatively straightforward example of what this could be used for).
5. Simple psychoanalysis program: Write a web page that uses a prompt to ask the user, "How are you feeling?" Then, think of some potential answers. Using an if branching condition to get the computer to write out an appropriate response (e.g., if the answer was, "I'm feeling down", you might want something like,

Thank you for your Participation

kindly reach amit.fegade@capgemini.com for queries.