LEARN TO WRITE JAVASCRIPT THE AWESOME WAY

JavaScript "Awesomeness Book"

GILAD E TSUR-MAYER

Javascript Awesomeness

Learn to write Javascript the awesome way

Gilad E. Tsur Mayer

Table of content

Stop Everything! I Have a Present For You!

OMG! OMG!!! OMG!!! Aren’t you excited? … who doesn’t *love* presents???

This book gets you to know Javascript, but personally I prefer videos over books – I mean, it’s easier if someone does all the work of reading for you, right? You can watch a bunch of videos and be an expert in no time.

So, I took this book, and blended it all into a bite-sized video course. if you will watch it, you will know every basic building block of Javascript in no time! Neat, huh?

And because you have purchased this book, I will give you 50% off this video course. Awesome, right?

Grab the videos here: http://basicjavascript.gilad.me/discount50/

Introduction

Hey Guys!

Welcome to the Javascript Awesomeness Course, where you will learn to write the basics of Javascript, the awesome way!

My name is Gilad, and I will lead you through this amazing course!

I began my career as a web developer, but soon transitioned to entrepreneurship, where I founded my very own startup company.

Currently, I work at the company I founded from scratch, doing what I love most - teaching you guys!

I designed this course for anyone seeking to learn basic of Javascript and begin a career as a rockstar web developer, as well as anybody who just loves to expand their knowledge.

By the end of the course, you will have a rock solid knowledge of all Javascript building blocks such as:

●Javascript Variables

●Javascript Functions

●Javascript If Statement

●Javascript Form Validation

●And many many more…

I will teach you the latest version of Javascript by the standards of the W3C Association. These standards are used by all the major companies in the world.

I will not only cover all these topics, but I will also give you an opportunity to practice them by giving you a pop quiz every now and then.

The ideal student for this course is anybody who wants to expand their knowledge of Javascript or get a leg up in the web developer world.

To take this course, you will have to know the basics of HTML and CSS, and also come open-minded to my silly jokes!

You are free to take a look at the course description, and I look forward to meeting you inside.

Chapter 1
Why Learn Javascript Anyway?

Did you ever sit in your comfy couch at your house and wondered;

Why Javascript?

What can it do for me?

Why all the buzz about it?

And, what is an elephant doing in my living room?

If it happens to you, pet your brand new elephant right away. And also, join me in being more knowledgeable about Javascript.

Right of the bat, Javasript is *really* cool! You can add logic to your website, make it more interactive, and also manipulate your HTML and CSS due to that logic!

But if you thought “cool” is the only feature of Javascript, then you are wrong, because Javascript is also relatively easy, and it’s the base of the most popular frameworks out there, such as angular.js, react.js, backbone.js, knockout, and even good old JQuery! Wherever you step, on the web development world, you’ll see the footprints of Javascript.

If you desire to be a web developer, then you just HAVE to be familiar with javascript. And that’s why you are here, right?

So I’m glad you are here. Let’s get started.

Chapter 2
What is Javascript?

Here’s a person, called Bob.

Bob has eyes, ears, nose, mouth…

He has a body, and also some clothes…

But until it has a brain that functions everything together, it’s only a bunch of organs connected to each other.

Just like our weird but beloved Bob, you can surely know HTML, and structure huge website.

You can also decorate and style your website, with your amazingly CSS skills.

But, if you lack in your Javascript skills, your webpages will look dull and old.

Javascript is the programming language of the web. It is our functioning brain that keeps everything in touch.

If you want your website to be more interactive, so your users could click on one element and be amazed when it actually does something, be sure you have your Javascript skills well put together.

So sit tight, and let us begin.

Chapter 3
Do I need to know anything before I learn Javascript?

Although it's a javascript course, you will need to know at least a little bit of HTML and CSS to understand the course content.

I will use several examples that uses HTML and CSS, so if you don't know any of those, please go back and strengthen those skills.

I do have HTML course and CSS course for those who needs it, and because I'm so awesomely generous with you guys, I will give you a 50% coupon for any of my other courses.

You can grab your coupon here: http://basicjavascript.gilad.me/discount50/

One more thing; for all my examples in this course I will use Notepad++, which is a completely free text editor.

(you can grab it here, for free! : http://notepad-plus-plus.org/download)

However, if you happen to be allergic to my suggestions, you can always try other software as well.

I won't judge you, I promise!

whispering **I will**!… ***whispering****

Chapter 4
How to set up your page

Alright Alright Alright!!! You are already sold about learning Javascript, you are supposed to be set with your Notepad++, and you want to start coding Javascript, right?

Before you do, you need to set up your page:

In order to do so, all you need to do, is to write inside of your head tag, a tag named script. Like that:

```
<!DOCTYPE>

<html>

        <head>

                <script type="text/javascript">

                      #… JAVASCRIPT CODE… #

                </script>

        </head>

        <body>

        </body>

</html>
```

Inside of the script tag, we will have an attribute called type/javascipt, that tells the browser we are having a javascript typed script.

Inside of that tag, you will write your Javascript code!

Chapter 5
Simple alert, write to document

OMG OMG OMG!!! I’m so excited!!! … just like a baby trying to walk its first steps, you are about to write your very first javascript code.

And it’s going to contain …*DrumsRoll*…

Printing a fluffy red goat!

Why printing a fluffy red goat, you ask?

Because there is nothing more suspicious than walking down the street and see a person printing goats just for the sake of printing goats. That is red… and… also fluffy.

And that’s the kind of things we want to be associate with, these days. Leather jackets and a fluffy red goat.
So… how are we gonna print our fluffy red goat into our screen?

All you need to do is to print document, follow by a dot, and then type the word write.

Right after that, you will have to open parentheses “(“ and close parentheses “)” . inside of these parentheses, we will have quotation marks.

And in the quotation marks, we will write down “I’m printing a fluffy red goat… ”. Because… that’s what we do.

And in the end, just like any language, if we want to end a sentence, you will write a dot, right?

But in javascript, we are writing a semicolon “;”.

```
document.write(“I’m printing fluffy red goat… ”);
```

We will save our file, and refresh our page and. Oh my!

We just printed our very own fluffy red goat!

Ah! Hack! That’s too good to be a singular event!

Let’s print *more* of these!

```
document.write(“I’m printing fluffy red goat… ”);

document.write(“I’m printing fluffy red goat… ”);

document.write(“I’m printing fluffy red goat… ”);

document.write(“I’m printing fluffy red goat… ”);
```

Yay…… We are having lots of fluffy goats….!

OK OK!!! … Now we want to warn everybody, before we are printing our goats, to watch out! Fluffy red goats are on the loose! Preferably with annoying pop up on the screen!

So we will write down alert, and then we will open parentheses, and inside our parentheses, we will have double quotation marks as well, and then we will write “watch out! Red fluffy goats are on their way!”

And we will close our statement with a semicolon, as usual.

```
alert(“watch out! Red Fluffy goats are on their way! ”);
```

Now save your document and refresh the page, and… oh! I am very impressed by your sort-of-a-road-sign thingy, warning everybody about what could be… the end of time!

By the way, did you see that we place our alert before the document.write, and refreshed the page, the alert showed first, but you didn’t see the any of the text on your document just yet.

That’s because javascript is operate as a queue. First in the first statement, then the second one… And so on!

Now you know all about writing and printing in your document, *and* all about annoying popups! As well as all about fluffy red goats.

In the next chapter I’ll show you how to write comments.

Chapter 6
Comments

Ssshhh!!! I have to tell you a secret.

Yesterday, I went to a fortune teller!

I sat on the chair, and then she approaches me and said;

“awww… My dearly beloved almighty, awesomely humble Gilad… I have to tell you something about your students…”

And you know me, I care a big deal about my young padawans, so I asked… “what?”

So she told me; ”In the near future, they will be working on a project… as web developers!”

And I was shocked! I said “WHAT??? Are you SURE? *My* students???”

So she said: “Wa… Wait, I didn’t get to the point yet!”. And I was like “oh… Yeah… well… Carry on“

“ …In the near future, they will be working on a project… As web developers!!! … And they will be working with ***other*** web developers as well!!! … “

“Other web developers??? Other??? Are you sure???”

“Indeed.”

So I ran quickly to my office and prepared you this lecture.

If you are about to work with *other* web developers, you will have to communicate with them properly, and there is no better option to do so, but with comments!

Comments are for your own sole purpose, and for your colleagues as well.

You see, Comments are not being rendered on the page, so you can write whatever you want, like “hey Jeff, don’t mention about yesterday, I was really hungry. And…. your computer was the only thing that tastes like chicken! P.S, this line of code is getting everything from the database.”.

So now you know why do you need comments, but how can we do that?

Simply write inside of your script tag two forward slash, and start typing!

Like that:

```
// Hey Jeff!… Sorry about your tasty chicken computer
```

Save it, Refresh it, and…. It’s a christmas miracle! Just like we’ve expected: nothing renders!

Now only me and my colleague Jeff would know about our little incident.

But what about if I want to leave Jeff multi-line comment about what I ate for dinner… the list of things I’m allergic to… and why I always laugh like a crazy person when it rains?

So in that case we would have one forward slash, followed by star. We write whatever we want to write, and then close it with star, and… another forward slash.

```
/* Hey Jeff,
Thanks for the present for christmas,
however I feel like We are not in that phase
of our relationship to sending each other
Parking tickets.

P.S. : This code is not working.

P.S.S : we are out of milk. So get one. thanks! */
```

Hopefully, you would not work with Jeff, because he is a handful…

But if you would or wouldn’t, at least now you know how to write Javascript comments.

Chapter 7
Variables

Do you remember back in school when your teacher said something along the lines of…

X = 98

And then she said something like:

“Danny just eat X apples”,

You all knew… You all knew she's talking about Danny, that just swallowed 98 apples, right?

That’s exactly how variables works on Javascript.

Javascript Variables are like little placeholders, they store your data, and then if you want to use that data later, you just call the variable name, and use it!

So before you will use your variable, you need to declare a name for it.

So to declare variable, the lazy javascript people just shorthand the word variable into just “var”… so you should always write “var” first, and then , you can come up with every name for your variable, any name you can think of… I’ll write my name for example, “gilad”

Then, you should write an equal sign, and then what you want to assign your variable to.

Let’s say we want to print the number 9.

And we will close it with a semicolon like always.

```
var gilad = 9;
```

Right now you already know how to print stuff to your page, remember our fluffy red goat?

```
document.write(“I’m printing fluffy red goat… ”);
```

So erase the text inside, and just write your variable’s name (no quotation mark needed!).

```
document.write(gilad);
```

Save it, and refresh it.

Holly goats! We’ve got a 9 on our screen!

Let’s explain what just happened:

Your browser thinking for himself, “oh alright, awesomely-cool-web-developer! You want me to set a variable, and then you call the variable ‘gilad’. So i’ll set a place in my room just for you… And I will call that place ‘gilad’ from now on…“

Then we wrote the equal sign… so that means you set it to whatever on the right side of the equal sign.

Which is 9.

So the computer says “ok, I’ll now store 9, in my place called ‘gilad’. And I promise you, I will use it whatever you will tell me to use it… “

Then when we called the ‘gilad’ variable inside our ‘document.write’ , the computer took the value, and print it inside of the document.

Oh, but one more thing… Do you remember I told you about the name of variable, and that you can name it whatever you like… Well… you can.

But you have 2 rules you *have* to keep, ok?

Rule number one:

Don’t ever write anything funky like that: #5^$%^^$. Your browser will go bananas,

and wouldn’t know what you mean by that name! You can only call your variables by letters,

numbers, and underscores.

Rule number two:

Javascript is case sensitive language, so don’t call your variable jeffery , all letter case, and

then afterwards call it JeFferY like a crazy person. For the browser, It’s like you will call

Some girl “Hillary!” and then switch to “Hey Brittney! Sup Brit… I mean… Doris!… Doris!!!

Jenni… Bath!… Elizab… Doris!!! Doris…”

Yeah… so don’t do that… Other than that. You’re safe!

Chapter 8
Variable types

Do you remember when you were a baby, you had this weird game of shapes, and you were supposed to sort those shapes and put every shape into its place?

Just like that game, Javascript has different types of variables. And they cannot fit everywhere…

Let me explain it to you by first writing it down in our text editor:

I will write var , *name*, equals to 9:

```
var name = 9;

document.write(name);
```

That’s what we’ve done in the last lecture, right?

We have a variable named *name*, and it’s equal to the number 9.

But, my name is not nine, right? (I swear it’s doesn’t!)

Let’s try to erase the number nine, and place my name instead.

```
var name = gilad;

document.write(name);
```

Save it, refresh it, and let’s see what we have here…

ah! Nothing shows! What happened?

The difference between 9 and a gilad, is that “gilad” is a text (or as Javascript like to call it: a “string”). And 9 is just a number.

The thing is, javascript could easily think that gilad may be just another variable, and because it could be a variable, it's trying to look for it in the *entire* page, and when it couldn’t find it, it just says… “Oh Well, alright… I did my best, I cannot find it anywhere… So I’ll just do nothing and relax… ” (giving up early, huh?)

To say explicitly to Javascript that we have a string here, we need to wrap our text inside of quotation mark.

Let’s do that:

```
var name = “gilad”;

document.write(name);
```

Save it, refresh it, and… yeah baby! We’ve still got it!

So now we know that we have two different types of variables.

Numbers, and strings.

And the difference between them is that little quotation mark.

We have a few more types of variables in javascript, and we will cover them in later lectures.

Chapter 9
Using different types of var

Did you know that Javascript can make your homework?

If I knew that, I would study Javascript at age 5!

Here’s an example:

```
document.write(3 + 3);
```

That could come handy in first grade, don’t you think?

Hmm… let’s try something…

I’ll declare two vars , the first one I will name num1 and set it to 4, and then second one, I will name num2 , and set it to the number 2.

Now in the document.write , I will replace the 3+ 3 into num1 + num2 , and run it.

```
var num1 = 4;

var num2 = 2;

document.write(num1 + num2);
```

Alright… we’ve got 6. Awesome! We’ve learned something new! We can actually add variables together! Imagine the possibilities!

But I wonder what would happened if I would change … say… 4, into “ 4 ” , you know, as a string…

Alright, let’s put our mad scientist goggles and try it out:

```
var num1 = “4”;

var num2 = 2;

document.write(num1 + num2);
```

Oh, that’s weird… 42?

I mean, I know the answer to life, the universe and everything is 42, but … I didn’t ask that…

Let’s see what happened:

First, we typed num1 , equals to 4. But since we have our quotation mark around 4, it would be considered as “string” and not as number. So our browser merely printed 4 on the screen. And then, we added 2. But since our 4 is only a string, our browser didn’t thought it’s a math problem, and just added it to the screen, as is…

So we could easily replace 4 to “I love you “

And if we will refresh the page, we can see that your browser loves you 2.

Browser!!! Silly you….

Chapter 10
Math operators

In math, we have 4 basic operations that you can make.

Addition, subtraction, multiplication, and division.

In the last lecture, We’ve already seen that we can add numbers in javascript.

```
var num1 = 4;
var num2 = 2;
document.write(num1 + num2);
```

But apart from basic addition, we’ve got another math operation in Javascript.

Let’s take a take a look at the rest of them.

We have got subtraction:

```
var num1 = 4;
var num2 = 2;
document.write(num1 - num2);
```

You simply make the minus sign.

If you want to multiply numbers, don’t write the x letter or something like that, simply write star instead:

```
var num1 = 4;
var num2 = 2;
document.write(num1 * num2);
```

That’s Multiplication.

We can divide numbers as well.

All you need to do, is to type a forward slash.

```
var num1 = 4;
var num2 = 2;
document.write(num1 / num2);
```

So that’s division.

Now I’ve got a bonus for ya!

Let’s say you want to increment a number by one.

So… of course… you can type something like that… :

```
var num1 = 4;
document.write(num1 + 1);
```

But! If you are a lazy person like myself, you can shorthand and do that: !

```
var num1 = 4;
document.write(num1++);
```

Or something like that if we want a number to be decreased:

```
var num1 = 4;
document.write(num1--);
```

The plus plus (++) sign (and also the minus minus), will increment or… decrement? (Can we say decrement?!) your number by one. It’s short and dandy, and we will also see that a lot in our future, when we will encounter more complex stuff on Javascript!

Chapter 11
Functions

Here’s a True story:

When I first came out from my parents’ house, I had a… well… a… little tiny… Really… mini-ti… tiny-problem.

Well, you see? I *hate* cleaning, and when I first came out from my parents’ house, nobody was cleaning after me (ahh!!! The horror!). So after a day or two, I had tons of dishes in my sink, my clean cloths blend perfectly with my dirty clothes on my bed, my floor was completely a mess, and my two cats didn’t really help (meow!).

So the first thing I’ve done, is to get me one those robots who clean the floor after you.

A perfect solution for the lazy man in our era.

And I was very glad! It was a repetitive task, that I could have done myself for many times, that some sort of… robot: is doing it for me!

So if we go to our world of Javascript, you may already think that there is a possibility to code the SAME code over and over again…

And if you are lazy like me… well… you will need to figure out how to make yourself a new robot!

To this robot, the Javascript-people called “*Function*”s.

Instead of writing the same code over and over again, your *Function* will basically store a segment of your code, and when you would like to execute it, you can call it, and the function will run.

Like I always say, Let’s see some example, and you will understand it better:

Let’s say we want to have a function that will congratulate us everytime we enter our website.

So, in order to write a function, all you need to write is….

“function”!

So go ahead, and write function, and then your name of the function, and just like your variables, you can name it whatever name you want, but it’s a good practice to go with a meaningful of what your function needs to do. So if we want to congratulate your users, let’s call it “Congrats”.

And then after the name, we need to open parenthesis, and close parenthesis (by the way, those parenthesis’ are actually have a meaning, but for now, just leave it without anything inside).

Then open curly bracket and close curly brackets. And inside the curly brackets we will type our code that we want to be execute when we call the function.

```
function Congrats(){
        //….javascript "congrats" code …
}
```

We want our code to congratulate our users, so let's write something on alert… Let's see…

"Welcome to the 'hamster-mobile' website. Where you can find a cheap vehicle to your beloved hamster… "

Well, I think we know what our next start-up would be, right? I hope we don't have any competition… hmm…

I have to check this one…

Oh… man… I cannot believe it!

Apparently there *IS* a hamster-mobile… ohhh… my dreams just shuttered like windows 8…

How can I recover from this one?

Anyway…

Let's also add a bit of code that tell us our congratulation alert has just been started. And also one text that our congrats just ended.

Let's run our thing!

Nothing shows! I wonder why?

That's because we set our function, but we didn't call it, right?

So let's call it. Just type anywhere outside of your function, your name of your function, and then open parenthesis and close parenthesis. And then end with semicolon:

```
function Congrats(){
        document.write("starting congratulations… ");
        alert("Welcome to the hamster-mobile website.
                Where you can find a cheap vehicle to your beloved
                hamster… ");
        document.write("ending congratulations… ");
}

Congrats();
```

Save it, refresh your page, and… BAM! We've got our weirdly congratulations up and running!

Now every time you want to use this thing, we wouldn’t need to write our document write again, and our alert… And another time the document.write… We just need to call our Congrats function again, and that’s it!

```
Congrats();

Congrats();
```

Chapter 12
Functions Parameters

Do you know that game show call “Let’s make a deal”, that do you have 3 curtains, and you suppose to guess which curtain has the brand new shiny car and which one has like a living adult llama?

I always wondered if they actually have to take the llama afterwards… And feed it, cleaning after it, be best friend with it… Even if they are allergic to that llama, but they don’t feel safe enough because that Llama is a vicious Llama…

Anyway…

Today we will have the Javascript version of the show “Let’s make a deal”!

First, we will have 3 inputs, typed button. Each and every one of them, will represent a curtain.

Curtain 1 , curtain 2 and curtain 3.

```
<input type=”button” value=”Curtain 1” />

<input type=”button” value=”Curtain 2” />

<input type=”button” value=”Curtain 3” />
```

Now let’s make room to our function:

Create a new function called ShowCurtain , but instead of leaving the parenthesis orphaned, type prize. (don’t worry, we will explain everything in a moment, just let me finish up this function…)

Now we will write inside of the function an alert, with “you won” message, and also, the same prize that we have inside of our parenthesis.

```
function ShowCurtain(prize){

        alert(“You won” + prize);

}
```

Alright, let’s explain what is that prize thingy…

Whatever we have in our parenthesis, called “*parameter*”, and our *parameter* is actually acting like a dynamic variable…

That means, when we will call our function, we will set the parameter in our parentheses just like we set every variable… And then, our parameter will live and be used as long as the function is running.

Alright, so let’s try something new, ey?

Instead of calling our function the last time, we can call it when our button is being clicked.

So add to each button an attribute called “ onclick ”, and set it to your function, and remember, our function asks for a prize parameter, so set it to whatever prize you want to get. And because we already use a double quotation mark for our HTML attribute, use single quotation mark instead to wrap our string… That should work.

```
<input type=”button” value=”Curtain 1” onclick=”’a shiny brand new tooth pick!’” />

<input type=”button” value=”Curtain 2” onclick=”’a colorful rubber band’” />

<input type=”button” value=”Curtain 3” onclick=”’a living Llama!’” />
```

Alright, save it, refresh your page, and let’s try to see what we’ve got here:

Let’s open our curtain number 1… Its… its… yes!!! A shiny brand new tooth pick! very handy… very handy…

…Now curtain number 2… The crowd roar with enthusiasm… it’s… oh my! A colorful rubber band!!! Cannot wait to use it!

Now… curtain number 3… what it would be? oh my!!! I cannot believe it… my all-time favorite! A living llama!!! who would feed this animal!?

I cannot believe it… What a wonderful day…

Javascript function parameters??? and a living Llama?! What more can an instructor ask for?!

Chapter 13
Multiple Parameters

Sometimes, you can have functions, that can hold parameters, but you will need to add another parameter as well.

In our last example, we had one parameter that asked the prize money, and we had a message that tells the user what prize that he won, but what if we want to add another information, like, what curtain number the user just opened?

If we would like to achieve this kind of task, we can add another parameter to our parenthesis.

Just add a comma, and type the extra parameter name you want to add.

For example, to make our message cooler, we will add another parameter called curtainNumber , and then prompt in our message “You opened curtain number ” and then we will have our parameter, and we will continue our text with what he won like we have done in the last example:

```
function ShowCurtain(curtainNumber, prize){
        alert(“You opened curtain number ”+ curtainNumber + “, You have won” + prize);
}
```

Now we will add the different curtain numbers to our buttons, so the parameter knows how to be set up.

```
onclick=” 1, ’a shiny brand new tooth pick!’ ”
```

But note that the order of the parameter is important, if we have our curtainNumber the first parameter, we will need to set it first as well…

Alright, let’s test our curtains!

Works!

yay!

Chapter 14
Return Statement

In the previous lectures we talked about functions and our functions just did something, right?

We print things on the screen, press one button and a message popped out, stuff like that…

But in some cases you want your function to return some value, so you could use it later.

That's why we have our return statement.

Let’s have an example:

Write a function called BasicAddition, that request 2 parameters, num1 and num2 . It will return the addition of both of them.

```
function BasicAddition(num1,num2){

        return num1 + num2;

}
```

Now we will call the function, and we will place the parameters 1 and 2. And we will expect to get the number 3 be out, right?

```
BasicAddition(1, 2);
```

If we will run it right now, nothing will shows.

And this is because we didn’t tell our function to do anything with it. It didn’t print, and didn’t even store it anywhere.

Let’s make a new variable, called result . And we will get the result right out of our BasicAddition function. And then, we will print result to the page.

```
var result = BasicAddition(1, 2);

document.write(result);
```

Let’s save it and refresh it, and… bam! We’ve got it!

We return the result of the number we added, we stored it on a variable called result, and then, we used the variable called result, and print it on the page.

Chapter 15
Calling function from another function

Ok so you already know how to write a function, and how to write a parameter for this function, and even multiple parameters, and even… (God forbid!), a return statement for this function!

Now I'll show you a little trick that will come handy one day…

You can actually call a function… within a function!

Let's try it out, create a function called GoodNight, and another function called GoodMorning. Without taking any parameters.

Inside of the GoodNight function we would have document.write, and we will say "Good night!" and in the GoodMorning function, you guessed it right, we will type "Good morning!"

```
function GoodNight(){

        document.write("Good Night!");

}


function GoodMorning(){

        document.write("Good Morning!");

}
```

And now we will have another function, called Start, it and we will call both functions from the Start function.

```
function Start(){

        GoodMorning();

        GoodNight();

}
```

Right now, if we will refresh our page, nothing will happen, because although we called GoodMorning and GoodNight, we have not initiate the Start function. Only when we will call the Start function, we will see stuff going on.

Let's do that!

```
Start();
```

See? When we started Start function, we run also both functions within it.

Now let’s try to break our browser apart. (don’t try it at home! Nahh… kiddin’. You can!)

We will take the GoodMorning function, and place it within the GoodNight function, and vice versa, and then we will call start by telling it to call GoodMorning. (and for the sake of readability, add a br tag to each write statement.

We will expect the browser to go to GoodMorning and say good morning, and then go to GoodNight and say “good night!”. And then back to good morning, good night, good morning, good night good morning, good night good morning… He supposed to get inside of some sort of a limbo… leaving its wife and its children… And even its own sanity!

```
function GoodNight(){

        document.write(“Good Night! <br/>”);

        GoodMorning();

}

function GoodMorning(){

        document.write(“Good Morning! <br/>”);

        GoodNight();

}

function Start(){

        GoodMorning();

}

Start();
```

Let’s refresh it, and… oh… wow… Better than fireworks!

Chapter 16
If statement

Before I wanted to be a web developer, I always wanted to be a hamster-tamer.

You know, just like a lion tamer that works with lions… But… with hamsters.

You suppose to tame them, so they can… do stuff.

Anyway, sometimes I wonder what if I was going with my dream.

And to make this thought come true, I will use my javascript skills!

In Javascript, you can have conditions in your code.

For example:

If x is true, then… execute this code.

If x is false… Then… do that code.

If x is greater than y, then run those functions…

Let's make it happened!

To write an if statement, you should type if, then open parenthesis, and then inside of those parenthesis you should write your condition.

Then you should open curly brackets, and inside of those curly brackets, your code will be execute, only if the condition is true.

```
if(##YOUR CONDITION HERE##){

        ##YOUR CODE HERE##

}
```

So let's say we want to check if the number 2 is *really* equals to the number 2. (you'll never know, right?)

Let's make two variables, num1 and num2 . And we will set both to 2.

So in the condition, type num1 == num2 .

Note that we are having two equal signs, and not just one equal sign. The reason for that, is that if we will type just one equal sign, the browser will might confuse and think we want to set a new variable. So two equal sign is the way to go, if you want to check if something is equal to something.

Let's change our code here to just say "If I was a hamster-tamer, my cow was broken", because she might be someday!

```
var num2 = 2;

var num1 = 2;


if(num1==num2){

        document.write(“If I was hamster-tamer,my cow was broken”);

}
```

And let’s run this thing!

And indeed that’s true, 2 *is* equal to 2, after all, and we have a broken cow just now.

What if we want to check rather if 2 is not equal to 2.

We should type instead of two equal sign (let’s erase that), we need to type exclamation mark, followed by equal sign. That’s “not equal”.

So let’s test just that.

```
if(num1 != num2){

        document.write(“my cow is broken”);

}
```

Let’s refresh that, and yup! As we suspect! We don’t get any message, that means that 2 still *does* equal to 2, and because we check for if it’s *not* equal, the code doesn’t fire…

If we will change num1 into the number 3, and run the code again, we will see that the code run perfectly, because 3 is not equal to 2.

Let’s learn another one! What if we want to check rather if 3 is greater than 2.

We will change our condition to this funky angle bracket…

```
var num2 = 3;

var num1 = 2;


if(num1 > num2){

        document.write(“my cow is broken”);

}
```

If we want to check if 2 is greater than 3, we need to change it to a greater sign.

```
if(num1 < num2){
	document.write(“my cow is broken”);
}
```

Now we can check if one number is “less than” the other, or equal…

```
if(num1 <= num2){
	document.write(“my cow is broken”);
}
```

So if we will check the number 4, it won’t run, 3, it won’t run, 2… ? bam. Got it.

And we can also check if it’s “greater than” or “equal”

```
if(num1 >= num2){
	document.write(“my cow is broken”);
}
```

Cool!

Chapter 17
Else & Else if

Let’s say you have a condition, for example….the word “banana” equals to the word “banana” :

And when it does, you will have a function called “ChoppingBanana”.

```
if(“banana” == “banana”){
        ChoppingBanana();
}
```

Till now, nothin’ new, right?

Banana is a banana, and the code will be fire.

And if I will replace one of the bananas into a Pineapple… the ChoppingBanana function will not be fired… right?

But let’s say I want some other function to be fire if the condition is not true? What do I do ?

In this case, we need to type right after our last curly brackets an “else” statement. and another curly brackets, and inside of those curly brackets we can write our code.

```
if(“banana” == “pineapple”){
        ChoppingBanana();
} else {
        ChoppingPineapple();
}
```

If we want, we can write another condition as well, we will write “else if” and another condition, say “banana” equals to “cherry”, and then another curly brackets.

```
if(“banana” == “pineapple”){
        ChoppingBanana();
} else if(“banana” == “cherry”){
        ChoppingCherry();
}
else {
        ChoppingPineapple();
}
```

So in this example, the browser will go first into the first if, and ask the condition rather if a banana is a pineapple. If it’s not true, then he will go to the next if, the “else if”, and ask… “Does a banana looks like a cherry?” , well it doesn’t… Oh! One more thing… You can have as many else if as we can wish!!! We can have like… A gazillion else if … If some condition is true, it will be fire the code inside of it, and leave the whole bunch of ifs and “else” and “ else ifs alone… But if it doesn’t, it will continue to ask until it gets to the last else.

The else is the final try of our browser, if no if statement and no else if was fired, than our browser will be satisfied and go in.

If your browser visited any condition, the else statement will not be fired.

Chapter 18
Switch

Who knows, maybe someday you will have your own web development company.

And as any other big company, it has its own… vending machine!

But we are not like any other big company, right ? our vending machine will be special!

So we will have the… *drum*

The Vending Pending Machine!

Our vending machine will make you pending. And… that's it. No soda or anything. Just… wait for a while, and go away.

Alright, so first, of course, we have to build our vending pending machine title, so we will add a few shticks in HTML…

```
<h1>The Vending Pending Machine</h1>

<input type=”button” value=”1” />

<input type=”button” value=”2” />

<input type=”button” value=”3” />

<input type=”button” value=”4” />
```

Now Let's build the Vending Pending Machine function.

I will type a function named VendingPendingMachine , that asks for a pendingTime parameter, and inside of our function, I will initiate a switch case statement, let me type it down, and I will explain it further afterwards :

```
function VendingPendingMachine(pendingTime){

        switch(pendingTime){

                case 1:

                        // CODE FOR #1…

                break;

                case 2:

                        // CODE FOR #2…

                break;

                case 3:

                        // CODE FOR #3…
                break;
                Default:
                        // DEFAULT CODE…
        }
}
```

So the browser comes to our function, with a parameter in hand, let’s say it has the number 1 as a parameter value, then comes to the switch case statement, it sets the switch case with the pendingTime as 1, and then it search for the right case, if case is 1, then it comes to the first one, if case number 2, then it fires the second one… And so on and so forth… It could be even a name like… “Hamster”, if the switch is equal to the case, then it will fire our code.

If no case matches our cases, then it will be fire our default case, but… Of course… Its optional case…

Let’s populate our cases!

We will make a message function, so the vending pending machine could say to our users, how many minutes they are suppose to wait…

```
function PendingMessage(time){
        alert(“The Vending Pending Machine says
                you have to wait ”+ time + “ minutes”);
}
```

So I will write a function called PendingMessage , that gets a time parameter, and then prints the message with the time parameter we got.

```
function VendingPendingMachine(pendingTime){
        switch(pendingTime){
                case 1:
                        PendingMessage(pendingTime);
                break;
                case 2:
                        PendingMessage(pendingTime);
                break;
                case 3:
```

```
                PendingMessage(pendingTime);
            break;
            default:
                PendingMessage(pendingTime);
    }
}

function PendingMessage(time){
    alert(“The Vending Pending Machine says
                you have to wait ”+ time + “ minutes”);
}
```

Now we need to place our functions inside of our buttons:

```
<input type="button" onclick="VendingPendingMachine(1)" value="1" />
<input type="button" onclick="VendingPendingMachine(2)" value="2" />
<input type="button" onclick="VendingPendingMachine(3)" value="3" />
<input type="button" onclick="VendingPendingMachine(7)" value="4" />
```

Alright!

Let’s try our vending pending machine for the first time!

Let’s press button number 1.

We will have to wait 1 minute… (But we won’t… because… I don’t wanna wait.)

Let’s press number 4, we should get into our default case…

We have to wait 7 minutes… Oh no…

Ok… you go to the next lecture, I’ll wait here.

For…. 7 minutes I guess.

Chapter 19
For loop

Back in the days, and still in the Simpson's episodes, there was a widespread punishment for naughty students. Writing like a gazillion times what they did wrong… For example… "I must not talk in class" or "I won't forget to do my homework again"… And stuff like that.

So today we will break all the naughty students free, with our shiny new Machine, that write lines for you!

So first of all let's make our document looks like chalkboard…

```
<style>
	body{background-color: #333333; color: #f0f0f0;}
</style>
```

Alright, first thing first, let's see how can we repeat the same stuff over and over again…

And to make that happen, we will use our brother-from-another-mother, the "for loop".

The for loop is a technique that lets you loop through any code, until you hit some sort of a counter… Let's write it down and I'll explain it further:

```
for(i=0; i < 10 ; i++){
	##… CODE… ##
}
```

Alright, so to create a for loop, you have to write the word for, and open parenthesis… Inside of the parenthesis you have 3 sections divided by semicolons, the first section you suppose to reset your counter. Because the for loop is running until you hit that counter. So the first section is to reset your counter, and you can call your counter whatever you like, I named it i , that's short enough, and reset it to zero.

Right after, we have the second section. It all about telling the for loop :"if that's condition is true… Well… just keep loopin' will ya?!" so I wrote " i < 10 ", which is in english : "as long as our counter 'i' is under 10, don't mention it, just keep looping… "

And in the last section, is the the counter increment. After every loop, do the plus-plus thingy, which if you are remember, means "increment yourself by one!", so if our counter i is 0, after one loop it will change to 1 , and then to 2, and then to 3 and so on…

And every time our middle section is satisfied, our code here will keep on running.

Alright, let's combine a lil' bit of for loops and functions, and create ourselves a Writing Line Machine!

?? ♬ …Writing Line Machineee…. yeahh… eaa… ea… a… ♬ ??

Yeah, so first of all, we will write a function called WriteLineMachine with a for loop inside of it, and that function will take a parameter called lineText .

And we will let the for loop running for 10 times for now…

```
function WriteLineMachine(lineText){

        for(i=0 ; i < 10 ; i++){

                //… Some write function…

        }

}
```

I will leave a comment now inside of the for loop , and now we will write a function that writing our text, and then we will insert it inside of our for loop, k ?

So let's write it down!

I will create another function called Write , and this function will ask for a parameter called lineText , and it will simply will write on our document, the same value the parameter has.

```
function Write(lineText){

        document.write(lineText);

}
```

And we will insert our new function inside of our WriteLineMachine … And link the parameter of the WriteLineMachine

```
function WriteLineMachine(lineText){

        for(i=0 ; i < 10 ; i++){

                Write(lineText);

        }

}
```

Alright, and now we will have another function that starts our machine, called StartMachine , and we will just insert our WriteLineMachine inside of it. With a parameter of lineText.

```
function StartMachine(lineText){
	WriteLineMachine(lineText);
}
```

Now, we will have a new variable, that holds a text. let’s say we want to repeat… this text:

“I will never ride my hamster-mobile to school ever again.” (I promise!!!)

```
var text = “I will never ride my hamster-mobile to school ever again.”;
```

And finally , we link all together with a StartMachine , and letting it use our text variable…

```
StartMachine(text);
```

So our code go to our StartMachine function, with our text, travels to
our WriteLineMachine function with our text, loop 10 times, and each and every loop he will visit our Write function, that just printing inside of our document, our text which is… Well… my forbidden rides on the hamster-mobile…

By the way, we can easily think of improving our machine, and give it dynamic variable of the number of loops… Like that :

```
var loopNum = 50;
for(i=0 ; i < loopNum ; i++){
	Write(lineText);
}
```

And that’s it.

Now we can trick our teachers and let our machine to write lines for us!

Chapter 20
Array

So now we will learn what Arrays is.

Sometimes you will want to list stuff on your Javascript, like a list of your user names, or dates, or whatever.

In order to do just that, you can surely declare a variable for every one of them, like var user1 , var user2, var user3, and so on… But it will be completely inefficient. (especially on bigger list…)

So for that kind of task, we have Arrays.

Arrays will store a list, but on just one variable.

Let me show you what I mean, I will create an array of names. all you need to do is to type " new " and then " Array ", and the Array word is something you have to type , you cannot type whatever you want, that's reserved name…

You open parenthesis, and inside of those parentheses you write your variables , I will type an array of string names as I said earlier.

Big Tony, Amazing Bob, Cool Jeff, and Awesome Jane.

```
var names = new Array("Big Tony", "Amazing Bob", "Cool Jeff", "Awesome Jane");
```

Now let's say I want to print on the screen the name Amazing Bob, what do I do ?

I will write document.write , and then inside of our parenthesis I will write name, because that's the variable of our array, and then we want to access Bob.

What we don't know is that each and every one of that array, has a secret index number, and instead of one, it starts with zero.

So Big Tony will be indexed zero, Amazing Bob will be indexed one, Cool Jeff will be indexed as two, and Awesome Jane will be three.

So to access amazing Bob, we need to type index one, right?

So to make that happened, we will open square brackets, and type 1.

Let's try it.

```
document.write(names[1]);
```

And here we go, we have amazing Bob right there.

And if we wanted to… get Awesome Jane printed on the screen… Let's see what's her index number… 0, 1, 2, 3.

Three is the index of Jane, let’s change it to 3 here :

```
document.write(names[3]);
```

And we’ve got Jane!

Now let’s do something funky! Let’s say our user Awesome Jane, decided to change her name to Tammy.

In that case , we would write names[3] to access her name, and then set it, just like we set a new variable, “Tammy”, and let’s add another document.write, to see that indeed it changed over time.

```
var names = new Array(“Big Tony”, “Amazing Bob”, “Cool Jeff”, “Awesome Jane”);

document.write(names[3] + “<br/>”);

Names[3] = “Tammy”;

document.write(names[3] + “<br/>”);
```

And yup. We’ve got it!

Chapter 21
Array Tricks

Alright! We have a few more tricks for Arrays!

We can add a new item with a push method

So let’s say we want to add 2 more names for example… Martha, and George.

```
names.push(“Martha”, “George”);
document.write(names + “<br/>”);
```

Let’s see if we’ve got it…

yup!

We can also sort it alphabetically… With a sort method…

```
names.sort();
document.write(names + “<br/>”);
```

Let's see… Yup! Sorted it out!!!

And we can also… reverse everything!

```
names.reverse();
document.write(names + “<br/>”);
```

Let’s refresh it and see what we got…. Yup!

Still got it!

Chapter 22
Javascript on Steroids

Alright, so far, we have learned lots of cool stuff, but right now, it’s time to take it up a notch.

I’ll write some advanced stuff, so bear with me, k?

Alright…

Do you know all of those movies that you have the villain, and he takes the hero starring as Tom Cruz, and he has like… A bomb… And everyone will die if he is not going to place the right combination of password… ?

Well, today we will be the villain that going to build this machine that prompt the password, and we will know if the password is valid or not!

So Let’s build stuff on HTML…

```
<div id=”sendItem”>

          <form>

                    Password: <br/>

                    <br/>

                    <input type=”password” id=”password”

                              placeholder=”password” />

                    <br/>

                    <input type=”button” id=”submit” value=”send!” />

          </form>

</div>
```

And let’s style it a bit, after all, it’s a special occasion!

```
<style>

          #sendItem{

                    background-color: red;

                    width: 200px;

                    margin: 30px;

                    padding: 30px;

          }
```

```
</style>
```

Now we will have a function, and we will call it CheckValidation , it will … check validation of the password combination… Of course…

Inside of our function, we will have a variable named password, and it will eventually get the value, of the input type element who has the id of password, let's type document, getElementById, value.

So inside of password we store the value of the input tag.

Then we will have an if statement, checking rather our combination is ok, so we will write password, is not equal to …. Let's see… what combination even the Pentagon can't crack??? …

Oh! 1234. That will do.

And we will have an alert , says "Caution! Password is incorrect!;

```
var password = document.getElementById("password").value;
if(password != "1234"){
        alert("Caution! Password is Incorrect!");
}else{
        // success!
}
```

Alright, let's place our function on the button , onclick…

```
<input type="button" id="submit" value="send!" onclick="CheckValidation()" />
```

And let's check if we type something , it's says error…

Yup!!!

It does!

Cool!

Now… note that we have a "send" button… So… we have to send our form somewhere, right?

Let me add a few shticks into our style tag…

```
<style>
```

```
    #sendItem{
        background-color: red;
        width: 200px;
        margin: 30px;
        padding: 30px;
        transition: 3s all;
        left:0;
    }
    .leftTrans{ left: 3000px !important; }
</style>
```

I added another class called leftTrans , and add a few more modifications to our sendItem id…

Now, after we will succeed our password combination, the form will be sent to us, the villains!

Let’s add in the else statement, var y, and it will take the id of sendItem , so we will access the element of sendItem … And now, we will take variable y, and add a class called, leftTrans .

```
else{
    var y = document.getElementById(“sendItem”);
    y.className += “leftTrans”;
}
```

And let’s try to fail only once… We got our message…

Let’s try again with our strong password 1234….

Ooop! We sent our form! Who knows where?! Maybe to mars.

Chapter 23
Now what ?

Congratulations!

You accomplished this course, you heard lots of really bad jokes in this book,

and I’m very proud of you!

You might ask yourself:

Now, that I’ve accomplished my first Javascript course, what should I do now? How can I improve my web developing skills?

Well, my young padawan, now it’s time to combine your Javascript skills with some Server Side.

I do have a C#course just for you, and because you purchased this book (or in compensation for my bad jokes), I will give you 50% off all of my courses in Udemy.

You can grab your coupon here: http://basicjavascript.gilad.me/discount50/