What to focus on in the baby phase

The focus of the baby phase is to get you familiarized with the JavaScript syntax. In this phase, you'll:

1. Learn how to declare variables.
2. Learn what are Strings, Numbers, Booleans, Null and Undefined.
3. Learn what are Arrays, Objects and Functions.
4. Learn to use `if` and `else` statements.
5. Learn how to compare variables.
6. Learn to use the `for` loop.

Here are some resources that'll help you through the baby phase. Feel free to go through any one of them. If you want to, you can go through all of them to get a solid understanding before moving on.

Free resources:

- Udacity – Intro to JavaScript
- You don't know JS – Up and going
- Code School – JavaScript Road Trip Parts 1, 2 and 3

Paid resources:

- Frontend Masters – Introduction to JavaScript Programming
- Frontend Masters – From Fundamentals to Functional JS

As you go through these resources, focus on learning how to write JavaScript. Follow through with the examples. Make sure you type your code word for word in your text editor so you get used to writing JavaScript.

Don't worry if you make a mistake. You probably had a typo error you couldn't spot. Learn how to use `console.log` to debug your errors so you know what went wrong.

Also, don't worry if you're not using best practices at this point. They key here is just to get familiar with JavaScript and be confident in using the basic things listed above.

At the end of this email, you'll find a list of questions you can use to check your understanding before moving on to the next phase.

But before that, let's tackle some common questions or challenges that'll surface at this point:

Common questions and challenges

1. How long would the baby phase take?

It depends on how much time you're willing to commit to learning. Generally, emerging from the baby phase with firm fundamentals would take you anywhere between one day to one week.

2. How do I use these syntax stuff that I just learned?

The easiest way to start using what you've learned is to link your JavaScript file to your HTML file with the `<script>` tag.

Once linked, open up your developer tools and navigate to the "console" section. (In Chrome, its `view > developer > developer tools`). You'll be able to use `console.log` to output your statements into the console.

Don't worry about using them to manipulate HTML and other stuff right now. You'll cover them in the next phase. Trust the system.

3. What is this ES6 thing that people keep talking about?

JavaScript is also named ECMAScript. The number 6 tagged at the end is a version of JavaScript. Traditionally, the most popular ones are version 3 (super old now), 5 and 6.

ES6 is one of the recent advancements in JavaScript. Now, almost all ES6 features have been integrated into JavaScript, so you can effectively use anything ES6 related.

At this point, diving too much into ES6 would be confusing for you. If you're interested though, read this article to find out more. Focus on the "const and let" part and the "arrow functions".

4. Why learn JavaScript, not jQuery?

Great question. jQuery is a library that's built on top of JavaScript. It used to be crucial to manipulate HTML. But, with recent advancements in JavaScript, jQuery is no longer needed.

Eventually, its more beneficial in the long run to learn JavaScript because you can apply its concepts to any library you use in the future.

5. Why does JavaScript seem so illogical?

Every language is different. They have their pros and cons. Their idiosyncrasies. Just like how English is completely different from French, JavaScript is different from any other language you may have came across, even though there may be similarities.

Instead of feeling that JavaScript is illogical, try embracing JavaScript's weirdness. Be open to it. Accept it for what it is for and you'll learn it quickly. Maybe you'll think JavaScript is awesome once you finally get it?

A list of questions to check your understanding

The final part of this lesson is a list of questions you can use to check your understanding.

Don't be overwhelmed by how big it looks. Try to answer each of the questions within. When you can answer everything, you know your fundamentals are firm enough to move on.

Here we go:

1. What are the five primitive values in JavaScript? (There are six if you consider ES6).
2. How do you declare and assign variables in JavaScript?
3. What's the difference between `const`, `let` and `var`?
4. What do each of the following operators do?
    1. `+`
    2. `-`
    3. `*`
    4. `/`
    5. `%`
5. What do each of the following comparison operators do?
    1. `===`
    2. `!==`
    3. `>`
    4. `>=`
    5. `<`

    6. `<=`
6. How do you use the following conditionals?
    1. `if`
    2. `if else`
    3. `else`
7. How do you use a `for` loop?
8. What is an array?
    1. How do you put values into arrays?
    2. How to you get values out of arrays?
    3. How do you remove a value from an array?
    4. How do you loop through every value of an array?
9. What is an object?
    1. How do you put values into objects?
    2. How do you get values out from objects?
    3. How do you remove a property from an object?
    4. How do you loop through every value of an object?
    5. What is a method on an object?
    6. How do you define methods?
    7. How do you call/invoke a method?
10. What is a function?
    1. How do you define functions?
    2. How do you call/invoke/execute functions?
    3. How do you pass arguments into a function?
    4. What does the `return` keyword do in a function?

That's it!

When you're done answering these questions, send it over to me by hitting reply. I'll help you look through it to make sure you're on the right track for the next phase.

Stay awesome Zell

[JSR] – Mastering the DOM (The Child Phase)

Hey {{ subscriber.first_name }},

Were you able to answer the questions listed in the baby phase? If you could, you're ready to move into the child phase. That's what this lesson is for.

Now, if you were unable to answer the questions, take some time to figure them out before moving on. It's important to get the fundamentals in place, or the child phase would be confusing and overwhelming for you.

If you're way past the child phase, you might still want to review this lesson and see if you missed anything fundamentally important. (Hopefully not).

Moving on, here's what you'll learn in today's lesson:

1. What should you focus on for the child phase?
2. How to overcome challenges that may surface for you
3. What do you need to know before moving into the teenage phase.

Now, without further ado, let's dive into what to focus on for the child phase.

What is the child phase again?

Just a quick reminder, you're in the child phase if you're comfortable with the basic JavaScript syntax.

Maybe you can copy-paste and modify other people's code after hours of googling, but you're not confident about building things from scratch yet.

What to focus on in the child phase

Your focus for this phase is to get yourself familiarized with the DOM, so you can change it and make anything you want. You want to focus on learning to build things like accordions and carousels.

(See the questions section if you have no idea what the DOM is)

To understand how JavaScript works with the DOM, you first need to brush up a little more JavaScript fundamentals. To be specific, you need to learn these three things:

1. What are scopes and closures
2. What is asynchronous JavaScript
3. How to write asynchronous JavaScript with callbacks.

Once you have a firm understanding of these three things in JavaScript, begin to focus on learning methods you can use with the DOM, things like:

1. How to select elements
2. How to add or remove classes
3. How to add or remove attributes
4. How to add or remove elements
5. How to use event listeners to add interactivity
6. What events are there and how to use them

Here are some resources that'll help you get through the child phase. Go through ALL of them (in order).

Free resources:

- You don't know JS – scopes and closures
- What is a callback
- Altering the DOM with JavaScript
- JavaScript 30 by Wes Bos (You may get confused while going through this one if your JS skills aren't stable enough yet. Work through it slowly).

Useful references:

- MDN Event Reference
- MDN Element API Reference

Once you've completed these free resources, think about what components you want to build. Work on easier ones like accordions first, then move towards tougher ones like carousels.

The key is to keep building things. It's okay to use other people's code and ideas at first. Once you complete a component, build it again. This time, don't refer other people's code. Make sure you write every line of code yourself so you internalize what needs to happen.

At this point, don't worry about best practices or code quality. You can get to them later. If you try to incorporate best practices now, you're trying to learn two things at once. It's a surefire way to get confused and overwhelmed.

(If you don't know how to think like a developer, I got you covered, see below).

You will feel like giving up

This phase is hard. Most half-hearted learners give up here, thinking JavaScript is too hard for them. Getting from theory and syntax no building stuff that work is hard because you have no prior knowledge of how stuff should work.

So, expect to google a lot. Watch videos and pay attention to how other people build things. Slowly, you'll begin to learn and understand the mechanics to it all.

To help you out, I'm building a JavaScript course that'll teach you everything you need to know from the baby phase to the teenage phase. If you're interested in this course, hit reply and let me know. I'll send you more info.

Common questions and challenges

1. How long should I spend in the child phase?

Depends on how much time you're willing to invest in learning JavaScript daily. If you learn for 2 hours a day, you might spend anywhere between a week to a month. Maybe longer.

Take the time you need to learn this phase properly. If you don't you'll just face more overwhelm and confusion later.

2. When can I move on to the next phase?

In this phase, you ideally want to build at least 6 components before you move on. If you build more things, you have more confidence in your ability to solve problems, which would help you in future.

Once you built your 6 components, head over the end of this email and see if you can answer every question there. If you can, . You're ready to move on.

3. What is the DOM?

The DOM stands for Document Object Model. It is a representation of your HTML. It's your "HTML" when you open up your developer's tools. It is here where you can modify your HTML with JavaScript methods. More info in this article

4. There are so many DOM methods. Which should I learn?

Focus on the ones listed in the question section below. They are the ones you need to master. The rest are additional things that are good to know.

5. How can I remember so many methods?

We remember things by using them a lot of times. The easiest way to remember the methods is to use them. So, as long as you keep building things, these methods will stick to you for life.

It also helps to download a JavaScript snippet pack like JavaScript completions for your text editor. They reduce the amount of code you type, which helps in remembering things.

6. How can I think like a programmer/developer?

If I'm not mistaken, you're asking this question because you don't know how to start.

The best answer I can give you is to Google. Search online for how you'd build an accordion to begin with. Notice how someone else builds the component. How does he think? How does he break down the problem?

As you build more components, you'll internalize this process of breaking down big problems into smaller ones, then solving the smaller problems to fix the big problem.

This is a process that you'll need later in your development career when you can't find any of your answers on Google. It's just you, your code and your brain. It will come, so take the time to practice now.

Here's an article that goes through the steps in detail . Read it. It'll help.

(Just a reminder, don't fall into the victim trap).

7. Writing HTML in JavaScript is difficult...

Yes it is. Template strings make them easier. Check this article out.

A list of questions to check your understanding

As before, the final part of this lesson is a list of questions you can use to check your understanding.

Before looking at the list, you'd want to make sure you've built 6 components from scratch. If you did so, you'll already have the answers to most of the questions in the list.

For questions that you don't know the answer to, feel free to build 2-3 more things that involves that concept so you get some practice.

Note: This list is HUGE. Don't feel overwhelmed. The questions are just broken down into small parts so its easier for you to understand and check things off. When you can answer everything, you're to move on.

Here's the questions:

1. What is the JavaScript scope?
    1. Why should you keep global variables to a minimum?
    2. What is a closure?
    3. Why do you use closures?
2. What is a callbacks
    1. How do you use a callback?
    2. How do you write a callback?
    3. How do you write a function that accepts a callback?
    4. Is `setTimeout` a callback-accepting function?
3. Asynchronous JavaScript
    1. What does asynchronous and synchronous JavaScript mean?
    2. How do you write asynchronous JavaScript?
    3. What is an event loop?
    4. How does the event loop work?
4. DOM methods and questions
    1. What is an Element?
    2. What is a Node?
    3. How do you select an Element?
    4. How do you select multiple Element?
    5. How do you loop through multiple Element for all browsers?
    6. How do you select the parent Element?
    7. How do you select sibling Elements?
    8. How do you select children Elements?
    9. How do you add a class to a Element?
    10. How do you remove a class from a Element?
    11. How do you check if a class is present on a Element?
    12. When should you add a class to a Element?
    13. How do you add an attribute to a Element?
    14. How do you remove an attribute from a Element?

15. How do you check if an attribute is present on a Element?
16. When should you add or remove an attribute?
17. How do you create a HTML Element?
18. How do you add your Element before another Element?
19. How do you add your Element after another Element?
20. How do you change the style of your Element?
21. Should you change the style with JavaScript? Why or why not?
22. How do you get the contents of a Element?

5. Events
    1. How do you add an event listener?
    2. Why do you add event listeners?
    3. How do you remove an event listener?
    4. When should you remove event listers? Why?
    5. What are the common mouse events?
    6. What are the common keyboard events?
    7. What are the common form events?
    8. How do you get the value of a event target?

[JSR] – Improving your JavaScript fundamentals (The Teenage Phase)

Hey {{ subscriber.first_name }},

You're in the Teenage Phase if you already know how to build stuff from scratch. You're somewhat confident that you'll be able to build anything DOM related. At this point, your code is still messy and unorganized. You're probably not too happy about it.

Your focus for the Teenage Phase

Your focus for this phase is to build even more things. Build things your boss ask you to. Build things your friends and relatives want. Build things that are fun for yourself.

Keep building. The more you build, the more experience you accumulate. As you build, learn to incorporate these four things:

1. Object-oriented Programming (OOP)
2. Functional Programming ideas (FP)
3. Asynchronous JavaScript (AJAX)
4. JavaScript Best Practices

Let's go through each of them in more detail.

Object-oriented Programming

Both OOP and FP are popular programming styles in JavaScript. In order to get good with JavaScript, you need to know both. There is no need to dive too deep into either programming styles right now, but you need to know the basics of them both.

OOP is a style of programming that revolves around objects. At this stage, strive to learn these concepts for OOP:

1. `this` in JavaScript
2. JavaScript prototypes
3. The Module and Factory patterns for creating objects

Functional Programming

FP is a style of programming that revolves around actions you perform with functions. In FP, you manipulate data and pass them around through functions. It is completely different from OOP.

For FP, strive to learn these concepts:

1. Reduce side effects
2. Write pure functions
3. Write immutable code

Forget about currying and partial application (you'd see them as you research FP) for now. Trying to learn these two principles would surely confuse you at this stage. You can learn them later when you're better.

Asynchronous JavaScript

JavaScript is single-threaded (it can only do one thing at a time). The key to JavaScript is to be comfortable with asynchronous JavaScript.

If you followed the roadmap so far, you would have already dipped your toes into AJAX with callbacks. The next step is to learn to use AJAX with the Fetch API and JavaScript Promises.

You may also want to learn to read API so you can use third-party (like Github's or Twitter's) API.

Here are some articles that will be useful:

1. Using Fetch
2. JavaScript Promises
3. Reading APIs

JavaScript Best Practices

Best practices are important. But they're the hard to learn, for good reasons.

First, there isn't a compendium of best practices lying around for JavaScript.

(I'm making one with Sitepoint now).

You often find best practices within articles and books. But the confusing thing is, best practices may conflict with each other.

That's because best practices are written by people. Different people have different opinions. You have to learn which ones to keep and which ones to throw.

Second you need to change the way you're coding now to adapt to best practices. You need to rewire your brain again.

I recommend you try to incorporate best practices by modifying your existing code. Make them better. In technical jargon, we call it refactoring.

Don't try to write code with best practices from scratch at this point. If you do so, you'll try to think logically and in best practices at the same time. Brains can't handle that. You'll end up in a mess.

Alright. So, best practices are hard. There's no one place where you can learn them. How do you learn then?

The best way is to read other peoples' code.

See how they're different from yours. Ask why. Understand why. Then, use them in your code.

Where to find other peoples' code? Your best bet is books and courses. As you read through examples, notice how they structure their code.

Your second best bet are plugins, modules and open-sourced code. They're harder to read than books and courses, but they show you how people code for real.

Don't try writing the perfect piece of code

You'll never reach it. Not now, at least.

Three months later, you'll look back and vomit at the shitty code you've wrote. Another three months later, you'll do the same. The cycle repeats.

Keep building new things. Once in a while, come back, refactor, then move on.

Note: Best practices change overtime as technology evolves. Some practices that are celebrated three years ago may be frowned upon now. The bulk of them did stay the same though. In the list of questions below, you'd find a list of best practices to know by heart.

Some questions to check your understanding

1. OOP
    1. How does `this` changes in different context? How many contexts are there?
    2. What is a prototype in JavaScript?
    3. How do you create objects in JavaScript?
    4. What is the module pattern? When do you use it?
    5. What is the factory pattern? When do you use it?
2. FP
    1. What is immutability?
    2. What array methods are immutable?
    3. How do you change JavaScript properties while not mutating the object?
    4. What is a pure function?
    5. How many kinds of actions should a function contain?
    6. What are side effects?
    7. How do you handle side effects when you write pure functions?
3. AJAX
    1. What are JavaScript promises?
    2. How do you chain promises?
    3. How do you catch errors when using promises?

    4. How do you use the Fetch API?
    5. What does CRUD stand for?
    6. How do you query Github's API to get a list of your own repositories?
4. Best practices
    1. Why do you avoid global variables?
    2. Why use strict equality (===) instead of normal equality (==)?
    3. How do you use ternary operators to help you write terser code?
    4. What ES6 features help you write terser code?
    5. What is event bubbling and capturing?
    6. How do you delegate events?
    7. How do you remove event listeners? When should you remove them?

That's it for the Teenage Phase. It's tough, but not impossible to complete. It might take you a few months if you quick, and probably 1-2 years to grasp it all if you can't spend time on learning JavaScript. Once again, don't let this stop you. You can move on anytime.

You'll learn more about the Adult Phase next. This is where it gets exciting for some of you.

Till then. Work on the Teenage Phase.

Stay awesome, Zell

[JSR] – Exploring JavaScript peripherals (The Adult Phase)

Hey {{ subscriber.first_name }},

You're in the Adult Phase if you know enough JavaScript to be dangerous. You can build almost anything you want. You're confident with your code. It's mostly clean; it follows many best practices. It's not perfect, but it's good enough.

Now, it's time to leave the nest and look for something new, something related that brings you closer to where your goal.

You have a few options here. You can:

1. You can learn a frontend framework (like Angular or React).
2. You can learn Node to build a backend.
3. You can dive even deeper into JavaScript.

Let's talk about these three options in more detail.

Learning frontend frameworks

When you use a framework, you're locked into its philosophy. Change is costly. So, before you learn a framework, ask yourself if you need it first.

Not everyone needs a framework. Sometimes, vanilla is the way to go.

Consider reading this article to help you decide whether you need one.

Let's say you want to use a framework instead of going vanilla.

You need to decide on what framework to learn next. That's a hard choice. Many people get stuck here.

There are many articles out there comparing different frameworks. I suggest you spend an hour (max two) to read through these articles, then choose one that feels right for your situation. Here's one that may be helpful.

It doesn't matter which framework you choose. It doesn't matter if you choose the wrong one. Keep going and learn that framework deeply first.

I say this because frameworks are similar to each other. They do the same things (routing, dom manipulations, controlling state, etc). Once you have a grasp of what to look out for in the framework you chose, you'll have an easier time learning another framework later.

If you can, construct a small project for hands-on practice as you learn it; you'll see things that you wouldn't see if you just read the documentations.

Other libraries

Aside from frameworks, there are many libraries built on top of JavaScript. A few examples are:

1. GSAP – GSAP is a ultra performance animation library that works all the way back to IE6. If you want to build complex, cool animations, be sure to check out GSAP.
2. D3 – D3 is a library for manipulating documents based on data. If you want to learn to visualize huge amounts of data, be sure to check out D3. You'd also be interested in Data Sketches – a place where Shirley Wu and Nadieh Bremer showcases awesome visualizations they've built.
3. Lodash – Lodash is a utility library that makes it easier for you to work with JavaScript. It's great if you want to learn to manipulate data.
4. Webpack – Webpack is the most popular library for bundling your JavaScript assets right now. If you master it, you can concatenate and minify your JavaScript with a one-step build process. It also allows you to use ES6 imports in your frontend code, which lets you create a better code architecture.
5. Gulp – is a popular task runner. In addition to bundling your JavaScript files like webpack does, you can create amazing workflows that save you tonnes of time. I even built a static site generator with Gulp for my website at zellwk.com. Here's ten free chapters from my Gulp book to help you get started.

Learn to build a backend

If you want to build a backend, I highly recommend you learn Node; it's JavaScript on the server. Node is a dedicated backend language, comparable to the giants like Ruby and Python. Many huge companies, like Netflix, PayPal and LinkedIn, use it.

The benefit to you learning Node is twofold.

First, you already know JavaScript. You'll be able to pick up Node faster than other backend languages (assuming you don't already know them).

Second, if you know Node, you can take advantage of it to use many frontend tools, like Webpack and Gulp, that are built on Node. You can also use npm as a package manager for your frontend libraries.

Besides learning Node, you need two more things – a server framework and a database language.

A server.

A server framework helps you spin up your backend quickly without having to type lots of code. You can learn to build a server manually without a framework if you want, but I don't think there's a need to.

The most popular server framework for Node is Express.js. I highly recommend learning it.

A database.

Think of databases like the hard drives on your computer. When you create a server, you need something like a hard drive to organize and store information.

There are many database languages, mostly split between Sequel based (SQL) and no-SQL databases. The most popular SQL based database is Postgres while the most popular no-SQL database is MongoDB.

If you're starting out, I highly recommend MongoDB because it's syntax is similar to JavaScript. It's easier for you to learn.

Here are some resources that'll help you spin up your first server quickly:

1. Building a Simple CRUD Application with Express and MongoDB
2. Wes Bos's Learn Node course (paid).

Diving further into JavaScript

Even though you're in the Adult Phase, there's still a lot you can learn about JavaScript. Feel free to choose any of these to dive into:

1. Object Oriented Programming

In the Teenage phase, you only learned the basics to OOP. If you want to dive further, you might want to check out these articles:

1. Subclassing vs the Mixin Pattern
2. How to use `bind` in JavaScript

You're pretty much done with OOP once you understand the above two concepts.

2. Functional Programming

If you want to dive further into FP, I suggest reading Professor Frisby's mostly adequate guide to Functional Programming.

In it, you'll learn about partial application, currying, monads, monoids, functors, endofunctors and many more FP terms.

3. The latest JavaScript improvements

JavaScript as a language has improved tremendously over the years. The latest stable version of JavaScript that's usable across all browsers is EcmaScript 6 (ES6 or ES2015).

If you want to, you can learn about future versions (like ES7 and ES8) of JavaScript before they become stable in browsers.

If you want to use these future versions today, you need to use a complier called Babel. If you need Babel, you'd want to either learn Webpack or Gulp as well.

4. Test Driven Development

Test driven development (TDD) is a software development process that emphasizes writing tests as you code.

It helps you improve your JavaScript skills by forcing you to write testable code. In doing so, your code becomes terser, purer and more maintainable.

To get started with TDD, I suggest reading Test-Driven JavaScript Development by Christian Johansen.

So many options, which to choose?

Keep a lookout for the next email. I'll share a strategy with you.

Stay awesome, Zell

[JSR] – How to choose what to learn

Hey {{ subscriber.first_name }},

In the previous email, you saw a lot of options to take your JavaScript education further. The question is – what should you learn?

You'd have your answer if you made yourself a learning map in lesson 3.

If you didn't make a learning map, or if you've already learned what you wanted with JavaScript, you should make a new learning map.

How to create your learning map

Here are the instructions, with a few modifications this time:

Step 1: Ask yourself what you want to learn or make next.

What you want to learn next doesn't need to have anything to do with JavaScript. Do you want to build a single page application with React? Awesome. Are you more interested in learning to design? That's cool too.

Remember, your motivation must be something you want for yourself. You must be able to visualize something concrete at the end of the path.

Step 2: Work backwards and create a list of things you possibly need to learn to get to your goal.

This list of things is your learning map. Order it in terms or priority.

Step 3: Reorder your learning map so it's listed in terms of priority for you.

The topmost item in the list should excite you. It may scare you too, but you can't wait to start.

If it doesn't excite you, you're not learning for yourself, but for the someone or something else. Your motivation would be weak. You'll lose interest quickly. You'll dread learning.

Life is too short for suffering like what you're going through! Redo your learning map from step 1. This time, make sure you're learning for yourself!

Step 4: Lift off.

Go and learn the first thing on your list. Once you're done, learn the second thing. When you're done, the third, and so on until you've completed the list.

Step 5: Rinse and repeat.

Start a new goal and learn again.

One thing to note: Whatever you set your sights on learning, go all the way. Make sure you learn enough to wield it fluently before moving on to the next thing.

Here are two articles that'll help you learn quickly:

1. How to learn to code quickly
2. How to remember what you learn

Never stop learning. That's why it's fun to be human :)

Go make an impact

As you learn new things, use them to build meaningful things for yourself, your loved ones and the world.

For me, that something meaningful is to teach you – and many others like you – how to design and code. I want to help you become good enough to build your thing, share your creation and make an impact in the world in your own way.

Together, let's make this world a better place!

Stay awesome, Zell

[JSR] – The final lesson

Hey {{ subscriber.first_name }},

You've come a long way in the past three weeks. You've learned how to learn JavaScript, you've became aware of the the traps that lie in your way.

What remains now, is for you to get your hands dirty with code. Go do the work. Go and learn.

In this final email, you'll get a summary of the important lessons you've went through. You'll also get a PDF of the course to keep.

Let's begin by going through the four phases of learning JavaScript.

The four phases of learning JavaScript

- **The Baby Phase** – This is where you start off when you're new to JavaScript. Here, you should focus JavaScript's syntax so you know what are things like Strings, Arrays and Objects.
- **The Child Phase** – You should be familiar with the syntax here. You already how to use statements like `if` and `for`. You also know how to use Arrays and Objects. What's next for you is to master the DOM. Try building simple components here. Don't worry about code quality at the phase.

- **The Teenage Phase** – In the teenage phase, you focus on improving code quality while learning advanced topics like AJAX, OOP and FP. Best practices go to no end. Feel free to move into the adult phase whenever you feel that you're good enough to explore something else.
- **The Adult Phase** – Here, you know enough JavaScript to be dangerous. This is where you begin exploring things built on JavaScript. You may explore frontend frameworks (like React and Angular), backend (Node, Express and MongoDB), libraries (GSAP, D3, etc) or even dive deeper into learning JavaScript.

The key to learning JavaScript is to take step at a time. Don't skip steps while you're learning, because they'll leave you confused and overwhelmed.

The Traps

Beware of the learning traps you set for yourself! Most learners face three self-defeating traps – the victim trap, the learn fast trap and the paralysis trap.

The Victim Trap

You're in the victim trap if you feel that you're powerless about your circumstances. Something out there is preventing you from learning JavaScript. You don't have enough time, money nor energy. Requirements for jobs are too insane, etc.

Break yourself out of the victim trap by realizing you **you control your life**. Carve out time for JavaScript. Stop making excuses for yourself, no matter how valid they are.

Start doing the work. Stop playing the role of a victim.

The Learn Fast Trap.

You're in the learn fast trap if you feel the urge to learn quickly. You've set yourself an unreasonable deadline.

Break yourself out of the learn fast trap by **preparing yourself to learn well**, not fast.

Take it slow. Pay attention and try to formulate arguments from what you read. While you do so, drop any preconceived ideas you have about the subject.

Experiment. Code. Fail with your experiments. Start another. Do do it over and over, tweaking till you succeed.

The Paralysis Trap

You're in the paralysis trap if you feel you need to make the right decision before starting to learn.

Break yourself out of the paralysis trap by **making decisions despite not knowing if you're right**.

Don't worry if you're wrong. It makes a valuable lesson. You know what not to do from now on. Thomas Edison didn't fail to invent the lightbulb for 9,999 times. He found 9,999 ways that wouldn't work. That's why he made it the 10,000 time.

Here's a PDF of the course

3 weeks is almost too short for a course of this magnitude. You probably don't want to lose these emails in your sea of emails. I get you.

That's why I prepared a PDF version of the course. You can download it by clicking the link below:

Download PDF of JavaScript Roadmap

Take this PDF. Put it somewhere you can see everyday. Use it to review the questions as you get through each phase. Use it to remind you of the traps you may face.

Most importantly, keep learning JavaScript. You can do it.

If you need extra help

If you need extra help with JavaScript, you'll be glad to hear that I'm making a JavaScript course to help you out. It's called **JavaScript Fundamentals**

In **JavaScript Fundamentals**, you'll get over 56 lessons to bring you through the baby, child and teenage phases of JavasScript Mastery. (You'd even learn some stuff from the adult phase). You'll also learn to build 21 things from scratch.

By the end of the course, you'll:

1. Understand JavaScript so well that JavaScript tutorials out there no longer seem foreign. They become understandable. Native to you, even.
2. Be able to build anything you want from scratch. No need to depend on plugins or modifying someone else's code.
3. Remember JavaScript for life, even if you didn't touch it for three months.

Hit reply and let me know if you're interested in JavaScript fundamentals. I'll send you more info when I hear from you.

If you need 1-1 help

I also offer direct help like 1-1 mentoring and code reviews. If you're interested, feel free to hit reply and let me know.

Lastly, thank you.

Thank you for going through the JavaScript Roadmap. I hope it has helped you find your way better around JavaScript.

If this has helped you in any way, I'd love to hear about it. Please feel free to hit reply and let me know!

Finally, good luck learning JavaScript (and everything else you want!).