Module outcomes

By the end of this module you will be able to:

understand and use JavaScript variables
handle mouse events with JavaScript
create JavaScript code to make decisions using logic
create JavaScript code which does repeated behaviours (loops)
create and manipulate data in a JavaScript array
generate random numbers using JavaScript

1. Learning JavaScript

So, now we have a series of videos exploring JavaScript.

So how am I gonna teach JavaScript, at least the introductory level of JavaScript? Very similar to HTML. Slides give us the backbone. And we've got some practical demos. And all those examples that you'll see, are all available on the website. And then of course we have assessment task at the end.

So, if you know some little bits and pieces of JavaScript, you could check the after this presentation message, which is at the start of every one of these videos. And also you could check the list of JavaScript which I include in an early slide.

2. About JavaScript_

Okay, now let's talk about JavaScript without actually looking at any JavaScript code.

So after this presentation, you'll appreciate the popularity of JavaScript and you'll understand where JavaScript code could be used.

So JavaScript. Is it popular? It it worth learning? Absolutely. It is the dominant web programming language. And one way to measure it is to go to some famous websites, such as Stack Overflow and see how many queries are related to that particular language, JavaScript. So just for interest, that's what we've done, and over the last few years, we have a measure of the different languages for this very well-known language where people ask queries about languages and programming, and you can see the red line up here is JavaScript. So the most popular language for this particular website, and indeed, generally the Internet in general, is JavaScript. So clearly, there's a strong trend towards JavaScript and learning JavaScript.

So we're building Internet systems. We have The browser we have The Server and they worked together both hubs. Where would we put our JavaScript?

So we can think a little bit more detailed like these different layers if you like, and this our browser sometimes called the client side. Those three layers, there's our server. Sometimes called the server side and if you imagine the person using a website, they might be up here and they would communicate through these different levels right down to the database. So, where would we find JavaScript.

So first of all we would certainly find JavaScript in the browser. You would have your basic JavaScript working with HTML and CFS and also libraries which build upon JavaScript. That's the first area. And

that's the area where we look at JavaScript in the following videos. In addition, much common these days, JavaScript is over on the server side as well. You can have a JavaScript server, program using JavaScript.

Even the database can be related to JavaScript using JavaScript data structures. So, we have JavaScript Client Side and also JavaScript on the Server Side including the data. So, very good idea to learn JavaScript. Here's a quick idea of the server layers on the server side. And so you might have a server written entirely in JavaScript. And there may be modules and layers and frameworks. Which, again, all use JavaScript. And you would build on those to build your server application.

On the client side, which is where we start, we have Java Script interacting with HTML and CSS. So you can think of it as three overlapping areas, if you like. But actually it's not quite like that, it's more like this. It's more like We have HTML which gives us our main display. That HTML is powered by CSS. When I say powered, I mean the different display parameters such as the margins, the padding, the colors, font size, font alignment. All of that stuff is supposed to be handles in CSS, and that goes to change the HTML display. And then JavaScript comes along, and JavaScript can change any of those things. JavaScript can change the CSS rules, okay. JavaScript can change the HTML, change the display. It can add stuff, delete stuff, change stuff it has almost total power to do things with the webpage.

So that diagram we just saw is down here that's your basic level of interaction between JavaScript, HTML, CSS down here.

Later, not in the following videos but later, perhaps in another course, we might talk about things like JQuery, which is a library built upon JavaScript, Bootstrap, which is a framework built upon JavaScript, and other things as well. And those greatly add to the power of this layout. However, so that we learn in an appropriate way, first we start with this, and then this is actually in other courses. So, that's the end of our brief discussion about JavaScript.

3. Getting To Know JavaScript

Okay, now let's start doing some pretty simple JavaScript.

So after this presentation you'll be able to write this simple JavaScript, and also you'll be able to use variables, and some simple functions which are part of JavaScript.

These are the functions that we're talking about, alert, prompt, confirm. They're all pretty straightforward to use.

So, we're going to do programming of JavaScript. Where should we put our JavaScript in the webpage? Well, JavaScript code can go almost anywhere in the webpage, however there is a common pattern. And this is the common pattern, which is you put your code near the end just before the slash body just the end there that's where you would typically put your code so the webpage has a chance to load before your JavaScript loads.

Now if you're using some libraries typically you would refer to those in the head. These are not exactly rules but they're kinda guidelines to make life a bit easier.

So, we've got some JavaScript code, we know where to put it. Well, what are the two main ways to

handle JavaScript code? So, one is you just put your code like this, script, /script and you insert that right into the HTML file where you want it, or at an appropriate place. As long as you've got scripts as script you can do that, that's one approach. Another approach is you put the JavaScript code in another file and then in the HTML you have this. Tell the HTML to grab the JavaScript code from another file and then in that file you just have your JavaScript, no need to put script slash script. In the other file, okay? So two possible ways to handle JavaScript.

So let's do some simple JavaScript, let's go through those three functions, which are part of JavaScript on the browser side. So let's have a look with alert, which we mentioned before. Bit of text, alert. Pretty straight forward. It brings up a window. Just brings up a little window right in front of the webpage with whatever message you asked it to show.

So let's try that one out.

Okay, it's loaded the example. Up comes the little window and you can see the message, welcome to my page.

Very straight forward.

All right, let's look at confirm. So confirm, pretty simple again, displays a pop-up box, but this time you get a chance to make some kind of decision, OK and Cancel. Typical example looks like this. Some kinda message, OK, Cancel. So, now you can, that is the user can make a decision. And your code can react appropriately. So let's try this one. Very straight forward. Up comes a little window. Want to go to Disneyland? And if we do okay then the browser is forced to go to Disneyland website.

What happens if you don't say OK? Then lets try that. Want to go to Disneyland? Cancel.

And then nothing happens for this particular example. Let's look at our code, Let's look at our code for this example. It looks like this.

There's our web page, script is being added in the head area which is no problem, and we've said immediately run some JavaScript, as soon as this webpage loads it will come down here and then it says, okay, immediately I must run this JavaScript.

We run the confirm. We show the message. And if the user presses okay, then we get a positive result from the confirm and then we go onto the next piece of JavaScript. And that next piece of JavaScript, it forces the browser to go to a particular website.

However, if you look at the window and you press the cancel button, the second button, then it will not go to any of this code. And in this example, it will do nothing.

Okay, let's talk about variables very briefly. Now if you've done any kind of programming before you'll know about this. Basically variable is like a box and you can but something in the box.

So here I made a variable and I put a number 7000 in that variable.

Later I can take the number out of the box, 7000, and I can use it. I can display it, do some maths, whatever I like. All right. I could also change what is stored in that variable whenever I like. Now, we're talking about variables so we can use it for our next example. So, here we make a variable and

then we use the third function we're looking at, prompt. And, that displays a message. And it gives the user a chance to type some text in. Whatever they type is then stored in the variable, and later we can use it.

So, let's do that, let's use our prompt. It looks like this. So you get a chance to type something in. Let's try it for real.

So there's the little window. What is your name? Type in your name. Something like that. Press OK. In this example, the name is stored in a variable. And it is used.

So I entered it. It got stored into a variable. And then it got used in the webpage. The webpage shows whatever I typed in. So there's one piece of text. There's another piece of text. That piece of the text is what I typed in. And there's also a third piece of text right at the end. One single character.

Let's continue with the presentation.

So here's the code of the example we just saw. As soon as the web page is loaded, it makes a variable, and then it runs the prompt function, what's your name? Whatever the person types in is stored in the variable.

And then we use that. We use that in a little message. We have our third line of code and it says browser. Immediately write something inside the document. In other words, inside the webpage. And write a message. Welcome to my page. And then, whatever the user typed in from the prompt window, show that. And then, the third piece of text exclamation mark. So I typed in David, so it shows David. Welcome to my page David exclamation mark.

Just as an illustration of how we could use prompt. And that's the end of our examples for this presentation.

Using A Code Editor For JavaScript

Lets have a look at using a code editor for JavaScript. For this, we're gonna use the Atom editor which runs on a Mac, but there are many other editors where the same principles apply. So the first thing is colors. These editors show the code using colors, and we saw that before in a previous video where we looked at HTML in one of these editors. So now we're focused on JavaScript. You can see for example the editor has used blue for JavaScript command like prompt, and it's used yellow for texts like this, and it uses some other colors for some other reasons. Those are added by the editor, and they don't go in the source files.

All right, so that helps because if we make a mistake for example, let's say we accidentally forget to finish our speech marks. Then you can see colors completely change, and that helps you identify there's a problem. And you can find out where fairly easily and then fix the problem.

Okay, so, that's one typical way to use one of these editors. Another one is auto complete, and let's demonstrate that. So, for example, let's say we're gonna write some new code, and we're gonna use the variable here. And all we have to do is type a few letters. The editor already knows that you have a variable called user_name, and it suggests it. So you could select it and then carry on typing if you want to, but we're not gonna do that for this particular video. So, it tries to help you. It tries to recognize things you've used before and suggests them when it looks like you're gonna use them again.

Okay, let's have a quick word on indentation. So indentation is the spacing on the left side of the code, and for JavaScript it doesn't care if you have indentation. It doesn't care if you have any spaces, or no spaces, or if you use tabs, or you press the space bar many times, it doesn't care. So it's your choice whether or not to use indentation with JavaScript. But it's a good idea because it helps you understand what's happening, especially when you use loops and things like that. But fundamentally, it is your choice how to do indentation in JavaScript.

All right, let's have a quick look at comments. So, there are two ways to do comments in JavaScript, so let's look at the first way. Let's pretend that our first big line of code, we want to have a difference. We want to change it and make a kind of difference. But we don't wanna forget what we used to do. Then all we have to do is do // in front of the line. So that's different to completely deleting it because we can see what it used to be. And then in our new copy, we can make an edit, and perhaps we're gonna try this slightly different language. What is your nickname? Instead of What is your name? But I still want to remember what I used to do, so I want to keep the previous line. I've just changed it to a comment, and then it gets ignored by the browser. But I can still see it myself. So that's one way to use a comment.

Another way is, let's say we have a bigger area, then you do /* at the start and */ at the end. And that will do any size of JavaScript you like. Maybe ten lines of code, no problem. You just do /* at the start and then */ at the end. And then everything between those two will get hidden from the browser, but you can still read it which is very useful if you want to know what used to be executed. So those are the two types of comment you can use.

Now finally, let's have a quick look at semicolons very briefly. In JavaScript you're supposed to have a semicolon(即;) at the end of each line, and that's what you see here. There is a semicolon. And that is very good practice. You should really do that, have a semicolon at the end of each separate line of JavaScript. Now you can actually omit it. You can actually do programming of JavaScript without a semicolon. However, it's not a good idea. Sooner or later, you're gonna make some kind of mistake, you're gonna get confused or the browser will get confused. So, in general, good idea, at the end of each line of JavaScript, have a semicolon like we've done here. All right, that's the end of our quick introduction to using a code editor for JavaScript.

4. Variables

So let's look at variables in JavaScript in a bit more detail.

So, let's focus on variables, especially the different types of variables in JavaScript. These are the two commands, if you like, of JavaScript we're going to look at.

So, Java Script has different types of variable or data and these are the three basic types, number, string and Boolean. And there's other types as well, such as, object, which we're not gonna look at in this presentation.

So number, first of all. Java Script has only got one type of number data type, variable. Some other languages have an integer type and also a floating points type. Floating point means a number with a decimal place. They have two, but JavaScript just has one type of number.

You can use a decimal place or not use a decimal place. As far as JavaScript is concerned, it's the same type of number.

You can also use scientific notation, so this is 1 2 3 to the power of 5. That gives you this number here. And we can have 123 to the power of minus 5. That gives you this number here. Those are all okay. Those are all types of number variables in JavaScript. Second type, strings. String simply means text in the world of computers. In JavaScript, you can use double speech var, like this.

Or you can use single speech var, like this. Both are okay, as long as you use the same type to finish it that you started with.

So here, for example, we have a double speech mark and we start it and then we finish with a double speech mark. That's great, everything looks good. That means we can use the other type of speech mark in the middle.

Double speech block on the outside so you can use the single speech block on the inside. No problem at all. Now, the third type that we're looking at is Boolean. So, Boolean is basically true or false. It's not text. It's the concept, the idea of true or false. So here's a couple of variables. One has true one has false. Those are both Boolean variables. Don't think that this is the same as text. So, this second variable here has the letters t, r, u, e stored in it. That it text. That is a stream. And this one has the Boolean idea of true stored in it, totally different. And that is a Boolean variable.

All right. Now, in JavaScript, there's one interesting feature. Maybe you start putting something in a variable like this. This is a string text, goes in a variable. And then a little bit later, you put a totally different type of data in the variable, like 98. That's a number. Number goes in the same variable. All right, that is totally okay in JavaScript. Anytime you want to, you can put a new value in. And it could be a totally different type of data. JavaScript is totally happy with that now, if you want to. Possibly, might be useful. There's a command TYPEOF. That command can be used to check the type of a variable. So, here's an example, and we are checking the text, John. John, with speech marks, is type string, which is correct. That message here comes from the type of variable. We'll see the code in a minute. Here a number, 3.14 is type number.

Any number with, or without, decimal place is stored in a number variable in JavaScript. Here false is type boolean in JavaScript.

Here's the code which created that.

It's quite straight forward. Alert brings up a little window. And then, we build all of that text and we show it in the window. We use \n\n simply to do a little gap. \n here means go to the next line. So, this goes down to the next line twice, because there's two slash ends.

But the main thing we're focused on is type of. So, we say, okay, John is type and then, we run the actual function. And then it tells us the type of John, which is text. Up, comes the message string, okay. Message string. Second one, okay, 3.14, up comes the answer, and the answer is, number. Okay? And then here, what's the type of false? And there's no speech marks here because that is false the boolean message, false, and that is boolean. Okay? So might be useful, possibly in some situation, checking what type of data you're using now, very common that we do things with variables. We change the values of them in some certain ways, so for example here, we take whatever number is stored in that and then we add one to it. And then we put it back in the same place. So we're kind of counting, okay. If that had 10 in it, we take 10, we add one, now it has 11. So it's a kind of counting idea. That's some typical Java script code. And we can type it quicker like this, count++. It does exactly the same thing.

Whatever is in that variable, we add one to it.

Okay, what about we take a variable, and then we subtract one and put it back in the same place? No problem, subtract.

We can type that quicker as well, count--. That will do exactly the same thing. All right, now does it have to be plus one or minus one? Well, here's another example where we do plus 10. Take the number in there, plus 10 and put it back in the same place. We could type that quicker in JavaScript like this count +=10. All right, what about if it's text? Okay, this has gotta be some text and we're adding some more text. This is a variable which stores text. Let's add some more text. When you have text and text like this, it will glue the text together, okay? This is different from numbers. If it's number and number, then it will add them together. If it's text and text, it will glue them side by side. This is text and text, and put it back in the same place. Again, we could type that quicker, text plus equals, then the extra piece of text you want to add on the end.

All right. So, we could do minus, plus, divide by, multiply. All of these typical things that you do with numbers, we can do them with variables. And we can do them long version and we can do quicker, typing version. So, for example, let's say pigs is pigs * 5, we could type that quicker. Pigs *= 5. Okay? Same thing. Cakes = cakes / students. Okay, cakes /= students. Quicker ways to type things which saves you time.

So another thing you might do is convert from one type to another. So there's a few functions and, very briefly, we've got parseInt(). Parse is a computer word which means interpret, analyze, and get the result. So whatever you put in the brackets, it will analyze them, and it will convert it to an integer. And then the result, you could put it in a variable, if you like. X equals parseInt, and then whatever you put in the brackets. There's also parseFloat, same idea, put something in the brackets, it will convert it to a float, which means a number with a decimal place, and then you could put it in a variable. String, same idea, you could put a number and it will convert it to text. And you could put it in a variable and start manipulating it. So there's a few commands for converting from one type to another. All right, that's the end of our discussion of variables in JavaScript.

5. Introduction to Events and Functions

So an important part of many programming languages is functions and for many systems, events as well. So we're gonna have a look at those in JavaScript.

So, you'll appreciate concept of events in Java script. We're talking about the browser and also how to use functions in JavaScript, which applies anywhere you use JavaScript. These are the commands we're gonna look at.

So, events what does it mean? An event in the world of computers, is when something happens, simply speaking. Now we are thinking of browsers, so basically we are thinking of perhaps clicking on something. Or maybe moving the mouse, that's an event. Pressing a key on the keyboard, those are events. Lots of things which trigger events. And then, you write some code, that code handles that event. That's the idea.

So let's look at our first event, which we've probably seen before. This one is the onload event, and pretty straightforward, it just says that this particular item, whatever item that it is next to has loaded in the browser. So in this case, it means the body, in other words, the webpage has loaded. When it's

finished loading, then run this line of JavaScript, straightforward. So, webpage has loaded, run this piece of JavaScript. And we've seen this before. This just shows a little window with a little message.

All right, so, complete example here, and that's filling in something in our body, which is not very important and there's our message, our instruction, webpage loads, run this JavaScript.

All right, now what could we put in that area here? What JavaScript could we add? Well, we could add whatever we want. We could add a huge sequence. So there's the first thing, do this browser. And then do this browser, another little window. And then something else, another window. And as long as you separate them with semicolons, then you could have a big, long series of JavaScript, if you really want to. However, that's not very efficient. Not a great idea to squeeze lots of JavaScripts in here. It's not smart. So really, what we should do, by the way this is what it looks like. If you run it, you get a window, hello, another window, start soon, another window, excited. Just to show you what it looks at. But we're really focused on the actual function and event idea. We're not really focused on what does it look like. So, let's think what could we do better. Instead of squeezing everything into a little space, which is very hard to manage, we want to really squeeze it into, not squeeze it, but put it together in a collection. And then call the collection, and we call that collection a function. So, first we define the function with any kind of name, like this. Open bracket, closed bracket. Braces, open braces, closed braces and then you put your JavaScript code in the middle, and then you run that function like this.

So, you group together code and then you run all of that code. And you can run it once like this or twice or three times, simply by doing this every time you want to run that code. Here's a complete example.

So this is the webpage, and we said when we load the webpage, finished loading, now we're using a function, greet_the_user(). Come up here, greet_the_user(). It's a function we've made to put together that code and so it does the same thing. alert() window, alert() window, prompt() window, and then it's finished, all right? But it's much more efficient to organize things like this with functions.

All right, so what can we do with functions? Well, if you've done any kind of programming before, then you'll know these kind of things. We could pass something to a function. So, that comes up in here. So, you run the function like this, whatever the name is. And then, you pass something to the function like this. That comes up in here. cats has the value 10, if we run it like this, cats has the value 10. You do something clever using the variable, cats. And that is how we can pass a value to a function.

All right, what else can we do? We could get an answer back from the function, a response from the function. So, let's say we say, okay, run that function. Up it comes, it comes here, it runs the code, whatever's in here. And this time it's going to return some kind of response. Maybe there's a variable called answer and in here you work out the value to put in answer. And then the value comes back and then it goes to the left-hand side and it gets stored in a variable and it gets used in some way, okay? You don't have to use a variable, you could return ten, something like that. Return text, dogs, something like that. You can return text. You can return a number. You can return a lot of things in JavaScript and then you might store them, and show them on the screen, hundreds of things you could do, okay? So that's the basic idea of returning an answer from a function.

So here's a simple example here. When you load this file you get to see a little message. This is my naughty homepage and then it runs the JavaScript code. Check the user's age. When the webpage is loaded check the user's age. That's down here. So what does that say? It says, okay, let's run another function, just for example, two functions working together. Check the age of the user. Is it less than 18? If it is, up comes a little window. Please go to another page. You cannot look at this webpage. What is

this, age_of_user? That's down here, age_of_user. What does it do? It basically just gets some input using prompt from the user, "What is your age?", stores it and then converts it to an integer, all right? And then stores that and returns the integer number, the number gets returned. Whatever they type in, that number comes here. So if they type in 99, okay, then 99 comes here.

Is 99 < 18? It is not. So there will be no message shown. What happens if they load the page and then they type in five, okay? Five will come in here as text, then we have to convert it to a number, like an integer over here, return the number over here. Is five < 18? Yes it is, so show the little window, okay? Let's run this example. Okay, so here we're running the example, very simple what is your age? All right, let's do what we just said, 99, press Enter. My age is 99, what happens, it runs through those functions and does nothing, in other words, I can still see the webpage. Let's try the second example, reload, okay, what is your age? The other example is like 5. So my age is 5, it goes through those functions and it says, wait, wait. Please go to another page.

And then, actually, it does not take you to another page. But still, that's our simple example, an illustration of using functions.

Let's go back to our presentation.

All right now, there's other things you can do with functions. Many things and here's one. We're just gonna quickly mention it, a recursive function. Now recursion is a little bit more advanced. Don't worry if you haven't heard of it or you don't know about it. The idea is simple. A function calls itself.

So, let's say we have a function like this, and it has some kind of variable going in. And then that function has some code, that code calls the same function all right? It's calling do_something. That is its own function name. It's the same thing, so that is a recursive function. It's a kind of special loop. A function that calls itself is a special kind of loop. How would you start it? Well you would start it in the usual way. But probably you would pass it some kind of number, some kind of control variable. That comes up in here and then when that function runs itself, you would adjust that control variable. And then it would come up here and come down here. And up here, and come down here. Each time you would adjust the control variable. That's the typical way of using recursion. So here's a simple example, and we're not gonna do much recursion. So we load the page, and then we have some code. That code immediately gets executed. It's my and then it has something here. It runs a function and then it says grandmother. Okay, so it's my, and then this is some text.

This text is constructed using a function. And what it does is, it calls itself. So that function is here, build_great, build_great. It calls itself in a special kind of loop. And we're not gonna spend a long time discussing it. The idea is, it starts with a control number like five. Five comes here, and then we come down here. Oh, add some text, add the word great in front of another load of text. The other text is gonna be created using the same function, but this time we adjust the control variable. So that comes up here and then we add some more text, the word great in front of more text, it adds the word great. great, great. We have a special kind of loop, a recursive loop, and it constructs some text added on the front of other text. Eventually it finishes. So, this is just a quick illustration of a recursive function in JavaScript. Don't worry too much about it. You don't have to use recursive functions in JavaScript. This will build a series of texts like this, okay? It's my great great great great great, five greats for this particular example, grandmother. That's what's happening with that example. Again, don't worry too much about recursion. It's clever stuff, slightly more advanced. You can do it in JavaScript but most people don't. All right, so that's the end of our introduction to functions in JavaScript.

Handling Bugs

Now when you develop JavaScript code sooner or later, you're gonna make some kind of mistake, which we call a bug. So how can you handle these bugs? Let's explore that idea. So pretend you're making this example program here, you would save the file. And then to try it out, you'd go to the browser and load the file. So we've just done that and nothing happened, not very good. We thought something would happen. Let's go back to our code. For example, there's an alert instruction here. That should bring up a little window, but we didn't see anything like that at all. So we do not have to guess what the problem is, we can use the browser to help us understand. All modern browsers have this kind of feature. We're using Chrome on a Mac. But basically, every browser has something similar. Now down the bottom, there's an area called Console (Google Chrome 中, 點右上角, More Tools, Developer Tools). And it says, Uncaught SyntaxError: Unexpected string. Remember, string just means text. And then if we go over to the right, it has the name of the file. And :6 and that is the line number where it found the mistake. If you click on that, it brings up a little window, it shows you the code. And you can go straight to line six and try to work out what the problem is and you probably say, yes, the string it was talking about was the double speech marks and the reason it's complaining about that is because we forgot the equals in front of it. So now, we go back to our editor and then we fix it. We say, that's what I meant to do. Let's save the file. Let's try again, check it out. Back to browser, reload. We still don't get the little alert windows. Now, what's the message now? It's changed. There's a different error now. So it says, Uncaught ReferenceError: promptttt is not defined. That's weird, why is there so many ts after prompt? And it says, oh, it's line number seven now. So if you come up, look at line number seven and you can see it highlights, this is a problem. I don't know what you're talking about, what is promptttt? So that's clearly an error. Let's go back to our code. Let's fix that and save it and go to our browser and try it again by reloading. Now we have it working. What is your name?

And then a message comes up. The value entered was Dave and then it displays a message in the web page.

Everything is working fine now, everything is good. So let's talk about this console window some more. Now there's a small version down the bottom here, but there's also another bigger version over on the right-hand side. Which is exactly the same, but it gives you more space.

So let's look at this Console. One of the interesting things is that you can try out any JavaScript you like right here in the Console. So let's try adding a couple of numbers.

We'd said, tell us what is 2 + 2 and then the answer came back 4. Correct. Let's try pieces of text, adding pieces of text together. What is speech marks 20 plus speech marks 2, the answer is 22. Because if you add together two strings, then you get a string, which is just both of those strings glued together. What else could we do? We could do alerts, just remind ourselves. Alert, some little message and see what happens and up comes the alert window. No problem. Now there's a message here undefined.

That is telling us from the alert function, nothing came back. It's telling you what came back from the piece of JavaScript you just executed. In this situation, nothing came back. Let's try another one. Let's try prompt and give it some text and then we run that. And remember, we can type in something that's what prompt is for and something like that. Press OK. And again, it shows us what came back from the prompt function. We typed in high, so back comes high from the prompt function. Let's try another one we know. Let's try confirm. Confirm give it a piece of text and up comes the little window, you OK. And remember, we have a choice. We can either do a positive button pressing OK or a negative button

pressing Cancel. Let's press the positive button and it tells us what came back was true. Now at this point, let's just illustrate some useful feature, which is if you press the arrow keys going up, the up arrow key, it will take you through all the things that you typed. And if you press the down arrow key, it'll go down through the list of things you typed. So you can press up and down. So let's use it to choose the last thing we typed. Let's press Enter, we're running confirm again. And this time, let's press Cancel. And now, back from the confirm function is false. You chose cancel, that is the negative button back from confirm comes false.

So this console is very useful. Now let's go to our code, back to our editor. Before, we were using alert to help us understand what's happening. Basically, we were using it for debugging. Now that's not very good, because anybody who goes to this web page will see that alert window. Really, it'll be better if only you or other developers could see the message. We can do that using console, console.log. So let's replace alert with console.log and that is gonna be saved and then we go to our browser. And then we load it in our browser and it's running, no problem. What's your name? Dave. And remember, we had for an alert. The value entered was Dave, but now we've replaced it with console.log. And now, we see exactly the same message right here in the Console.

This is great, because we can see it. But people who are regular users of the browser page, they can't see it unless they know how to turn on this special display and most people don't. So this is excellent, very good for developing extra messages. And the messages could do whatever we like. This is especially useful if you have lots of messages. For example, maybe you have a loop and you're keeping track of what happens. In every time you run a loop, maybe you have a message. If you used alert, you might have 20 little windows, very troublesome. But here, you would see 20 messages, which is much, much better.

That's the end of our discussion of handling bugs in JavaScript.

6. Making Decisions

Let's look at making decisions in JavaScript.

So after this presentation, you'll be able to make decisions using if statements, and also switch statements.

These are the JavaScript structures we're gonna look at.

So, if, pretty similar to regular English.

Lots of variations when you actually apply that idea in JavaScript, so we're gonna have a look of those.

And it's all about comparing things. So how do we compare things in JavaScript? These are the main ways to compare things in JavaScript. So there's some obvious things like, is something less than something else. Less than or equal to. Greater than something else. Greater than or equal to. Is it equal to something else, then you need to have two equals in JavaScript.

And how do you say, is not the same as? Then you use this way, okay? Again, you need to type two characters. So those are where you have to be a bit careful, but these are pretty straightforward when you're comparing things. So, first example. We are asking some simple questions. So, what is your name? And then we get a response from the user. We're using the prompt function. The response goes

in to user_name variable. And then we check, is whatever they returned,

which is stored in the variable, is it the same as dave? d-a-v-e. Remember two equals when you're comparing. Okay, is it the same as D-A-V-E? If it is, okay, alert. Great name! Up comes a little window with a message.

So that is our first simple example of an if statement in JavaScript.

All right, now what about these braces? These little braces here at the start, at the end? So basically, in JavaScript if you have an if statement and then you have one line of code, there is no need for these braces. No need to do that at the start, that at the end. But if you have more than one line of code, yes, you must have those braces to

glue them together, if you like to think of it like that, okay? So here we have two lines of code, so we must have those braces. One at the start, one at the end.

All right, so we had a simple if statement.

Let's do an if, and then an else. Looks like this, so, what's your name, stored it a variable. And then, is the name entered by the user the same as, d-a-v-e? If it is, we do this thing. A message appears on the screen. If they did not enter, d-a-v-e, then we do else and we do the second thing.

So, if it's the same we do the first thing, otherwise, we do the second thing. Else is the same as otherwise.

All right, so that was if else. Now lets do, if else if, to add more tests. So enter your name. All right, first check, is it the same as dave? Then do something. Else, if that test does not work, we go tot the second thing. Another test, if it was the same as jogesh, then we do another message. So, else if, takes us to a second test, to another test.

So we have two tests there. What about if that test fails and that test fails, then nothing happens. If the first test passes we do this thing. If that fails we go to the second test and then if that passes we do this thing.

So that's the logic we have so far. So let's extend that structure. Let's add an else at the end to capture the situation where they don't type in dave and they don't type in jogesh.

So we're extending our if, so if it was dave, then you do this. If it wasn't dave, we go to our second test, was it jogesh? Then we do this. But what about if wasn't dave, and it wasn't jogesh, all right we could do the third test, was it oz? If so we do this. What about if it was none of those, then we can catch the situation where none of those tests was true, and we capture it with some kind of a message at the end, if all of those failed then we go to a else. Okay, so that's the idea of the else. So, as you can probably guess, we can have as many of these else if, else if, as many as you like right there in the middle. The first one is the if, the last one is the else, and then you have as many else if as you like in the middle.

Now, that's one way to handle a series of tests. Another way is switch. So, switch looks very different, but it's quite similar in the actual result of what it does.

So you're checking something, probably a variable name.

And you say, okay, if it was this, then do this, break. We'll talk about break in a minute. And then you keep going in that pattern. If it was something else, then do something, break, and you keep repeating, keep repeating. And you have several different tests, which is similar to the ifs, lots of tests. And so, here's the last test, perhaps, and if that's true, then we'll do this, break. Now what about if all those tests didn't work? Well, we can catch that, a bit like the if. We can catch that with a default at the end. It means this wasn't true, this wasn't true, this wasn't true. So, catch the situation where none of them was true and do something. Right to the end, you don't need to have a break.

Okay, so what's this break? Well, this break stops any more tests being made. So, if this was true, it will do this, and then tell the browser, don't do any more checks, by typing break.

That's the idea. Otherwise, it might go and it might do more checks and possibly do more code, and maybe that's not what you want. If you don't want it, you have a break.

So here's an example, similar to our if statement before, get your name and check the name. If it was dave show a message on the screen and then break, which means go to the next piece of code.

If it wasn't dave then we'll go to the next test. Was it jogesh? If it was jogesh, okay, show a message, break and then carry on underneath if there was more Java script. If it wasn't dave, wasn't jogesh, then we can catch with default. Anything else they typed in, okay, your name isn't great, something like that. All right, so let's focus on that break a little bit more.

We used break before, but perhaps you don't want to use break. So let's show an example where we haven't got break in every single test.

So, what country would you like to visit?

Type it in and then check. If you say Canada, or if you say France, then show a message, okay, take me as well. And then, break and it won't do anymore tests. So, if you typed in what country, Canada, will check Canada and also France, they will both be treated the same. They will both get this message, and then we'll break, we don't wanna do any more tests. It will ignore this and it will carry on to the next JavaScript.

Alternatively, maybe you typed in what country, you typed in Japan, or Philippines.

So again, show a message if you type in Japan or Philippines. And then break, and that will stop any more tests. Or perhaps you typed in North Korea, all right, show a message and then break, no more tests. If you typed in any country except those five countries, we can catch that in default and just give some kind of general response.

All right, so that's the end of our discussion about if and switch case.

7. While Loops

Let's look at loops in JavaScript, starting with the while loop. So, this is all focused on while loops, and there's also a related structure called a do while loop. So, those are the commands we're gonna look at. So, what is a loop? If you've done any programming before, you'll know a loop simply repeats some code again and again. And we're gonna look at two of those ways of doing that in JavaScript.

So, the first one is while, and this is the structure. And you have some code, and then around that you have a while something is true, do that code. So, this particular structure, it will check if this is true, whatever this is. This is some JavaScript code, perhaps. Or maybe a variable, something like that. When this is giving a result, then we will do this code. When this code is finished, then it will come back up the top. And it will check again, is this true? If it is, it will do it again. Back at the top, check, is it true? Okay, do something again. Is it true? Do it again. And we have a loop pattern. That loop pattern will be executed again and again as long as that is true.

However, it's possible this particular pattern never gets executed. If that is not true then it won't do this code, not even once. All right, we call it the word iteration when we do a loop. We'll come back to that. Now, in the next example, we use indexOf, so just quickly remind ourselves or explain what indexOf is. And indexOf is giving you the location of something. So, you have some text, a string as we call it in programming. And then inside that text, tell me the position of whatever you're searching for.

So let's do an example here. There's some text. "The cat's hat was wet." And then, let's do searching using indexOf. Search for "at," the letters A-T. So, it will go through this, and it says, there. There is the first A-T. It doesn't do the other A-Ts. It just does the first one.

That will give us a number, and then you can use the number. Here, it will give us the number five, okay? Because A-T starts from position five. This is not really related to loops, but we use it in this example just to help us understand. So, we have a loop here, and let's run it just to show the idea. So, here's the example. It shows your statement. Rossiter is a great name. And so, you can say OK. That's great. And then it asks you, do you agree? So, you can give your response. Type it in. Do you agree? So you could say, for example, no, I don't agree. And then you enter that, and it comes back, and it shows you the message prompt again. Do you agree? And you could say whatever you like, perhaps no way, whatever, and press enter, and it keeps giving you a chance to get the correct answer. Sp, you could do whatever you like. No no no no no. And you could even say, perhaps, something like that. And it's not happy with any of those answers. It's looking for something where the first letter is a Y.

So, eventually you kind of give up, and you could do some, lots of different answers. They won't trigger stopping the loop. We are stuck in the loop. And the way to stop our loop is to do something like yes, or yes yes yes, or yeah, or whatever you like, anything which has the first letter as a Y. Or even just do the letter Y. So, let's type in yes, press Enter, and it has broken the loop. It has stopped the loop. The loop checking is now true. It's not doing any more loop.

Let's go back to our presentation. Okay, so that was a demonstration of our little program here, simple loop, and to do that we built a little variable called finished. That is a Boolean variable, so it has true or false. And we start it with false. We have not yet finished our loop.

And then our while statement said, while we have not finished. Remember, that exclamation mark, it means the opposite, right? Not. While we have not finished, do this. And then finish the loop. Come back at the top. Have we not finished? Do it again. Have we not finished? Do it again. Now, when do we change the value of finished? We change the value of finished in the loop, and we get some input from someone, and we store it in a variable. And then we check, is the very first letter the letter Y? Okay? If the first letter position 0 of Y is there in what the person types, then the finished variable is changed to true.

Now that means the next loop, when we do the checking, it won't do any of this code. So, when you

type yes, or yeah, or yep, or something like that, finished becomes true. Back up here, the next check, then it will not do this. All right? So, that's our first example of a while loop, and so some examples we just played with. Rossiter's a great name. Do you agree? I do not. That will not stop the loop. Do you agree? No. That will not stop the loop. Do you agree? Yeah. Okay. That would stop the loop. All right, second type we're gonna look at is do while. Very very similar, right? It's the same word, while, but now we put do at the top. So it will do whatever is inside those braces, it will do that code. First it will do it once, and then it will check, is something true? This part's like the same. The difference is here, it's guaranteed to do this once, and then it does the check. That's different from the previous one. The previous one maybe it never does anything, because it does the check first. This one is different. It does something, and then it does the check, okay? So it's a little bit different. Kind of an upside down version of a while loop.

So, what I did here was, I just took that previous example. It behaves the same, and I just changed it. So it's a do while loop instead of a while loop. All right? So, that means it will certainly execute whatever is in the loop, and then it will do the check. And that makes sense. That's totally okay for this example. You know, Rossiter's a great name. And then ask them, what do they think? So we wanna ask them at least once. So, it makes a lot of sense to use a do while loop which forces that to be executed at least once. It's a suitable approach.

All right, so that's the end of our discussion on while and do while in JavaScript.

8. More on Variables

Let's have a little bit more discussion about how variables work in JavaScript.

So we're going to focus on these two types of variable, local variables and global variables. It sounds a bit scary, but actually it's pretty straightforward.

So, variables. We use variables all the time. You've declared them. If you declare a variable in JavaScript within a function, remember VA X, something like that? That is declaring a variable. Then, that variable can only be accessed within that function, not outside the function.

And the language we use to understand that is we say the variables are local to the function, we use the word local.

So they're local variables.

So let's have a quick illustration of that. Here's an example and this is the part of the code which gets executed first.

Because there's a function, so that's not going to be executed first. There's no events here. All right, so first line of code and it carries on down here and we have a jump to that function in the middle. So we've got a variable. Money is 99, and then we've got a message. Okay in the main part, the value of that variable is, up will come the number 99. We put 99 in the variable, and then we check what is in the variable. 99 will come up, right here, a little message.

And then we've run this function here. This function comes up here. What does it do? It makes a variable with the same name.

So variable with the same name, however, it is inside the function. Now, because this is declared inside the function, like here, var money is 2, declared inside the function, that money will work, but it will only work inside the function.

So, when we run the function, we will get a message and it would tell us money has the number 2. This message will show as the value 2. Okay. Because his talking about this function here, or this variable here, excuse me.

local variable:
最上的 alert 中, money = 2
中間的 alert 中, money = 99
最下面的 alert 中, money = 99

All right. So, first one shows 99. Then we run the function. And the function shows the number 2. And then finally, what do we do? We show another message, and we check what is in this value. And it is 99 again, okay? 99, 2, 99. And that proves that this has a value, which is different to those. Even though it is the same name, that is a different variable to this variable with the same name. All right, so opposite of local variable, which works inside a function, is a global variable. Global variable basically works anywhere.

So let's have a look at that, and here's our code. We've made a variable here. Now that variable is a global variable, that's the language we use. Money is declared here. It's not declared inside a function. It's declared outside of all the functions. That means it's a global variable. You can use it anywhere, in your JavaScript code anywhere.

global variable:
最上的 alert 中, money = 99
中間的 alert 中, money = 99
最下面的 alert 中, money = 99

All right so, it's got the value 99. If we run this it will message here, up comes a new window and it shows us it has a value 99. We run the function, all the function does, same thing. Show me what is inside that variable called money. That will also show the number 99. And then, back here in the main part it will also show 99. Okay, so all of these three will show 99, because they're all referring to the same variable, which is this variable, which is a global variable that works everywhere.

Okay, now what about if you have global variable and a local variable, and they have the same name? Okay? Definitely not a good idea, I wouldn't do that if I was you, you're going to get confused and make programming mistakes. But still, what would happen if you have local variable, global variable with the same name, such as money? Well, in that situation, JavaScript gives priority to the local variable. All right, also, what about if you create a global variable inside a function? So if you assign a value to a variable that has not been declared in a function, declared meaning var x, something like that, so if you're in a function and you assign a value to a variable that has not been declared, it will automatically make it a global variable. Okay? Quick example here. So, first thing we do is we run that function show_money up here, we have show_money. It hasn't got any var money. There's nowhere there's any var money. It just says money. Now, JavaScript does make a variable called money. If you simply use a variable and it hasn't been created, JavaScript automatically creates it. Fantastic.

What does it create? Local variable or global variable? In this situation, it creates a global variable. In

other words, this variable can be used anywhere by any JavaScript code in the web page. When you do this, if it hasn't been already created with var, then it is a global variable, and that's what's happened here.

So, in here, if we check what's in that variable, it will come up with a number 2. What about back after the function, and we check in the main part, it will also come up with the number 2, okay? It is proving that is a global variable.

All right, so that's the end of our quick discussion about global variables and local variable in JavaScript.

9. Logical Operators

Let's look at how to handle logical operators in JavaScript. Sounds a bit scary, but actually it's very straightforward. So, after this presentation, you'll appreciate Boolean values, and what they mean in JavaScript, and also how to manipulate them, logical operators these are the different bits of JavaScript we're gonna look at.

And so, Boolean, that basically means true or false, right? So, if you have a variable which has a Boolean value, that is a Boolean variable, it either has true or false.

Now, what about manipulating and checking, testing these true or false values? Well, there's three ways to test and check and manipulate them, and they are called and, or, and not. Now just to be extra clear that this is about true or false. People often say Logical And, Logical Or, Logical Not, as opposed to general English, or something like that and, Or, or Not.

These things work with those Boolean values, which have true or false.

So let's go through them. First one, AND, Logical And. You're gonna test two inputs. Two inputs that are true or false.

The result if you do logical and checking those two inputs is true if both of the inputs are true. Otherwise, the result of logical and is false.

All right, example helps to make this clearer.

Sometimes people summarize it in this kind of diagram which is called a truth table. This is the possible first input, this is the second input and then this is the result of logical and. Those are the possibilities. False, false. False, true. True, false. True, true. Those are the possible input combinations, and this is what you'll get coming out. And basically, in a quick English summary, if both inputs are true, the result is true. Otherwise, the result is false. That's the way I remember it. So, here's some examples. Here we have three variables not even two variables but three variables which are working together these are Boolean variables because they have true or false and don't think that this is anything to do with text It's nothing to do with text it's just these special words true or false. So, we have three variables, and one of them has false, and then true, and then true.

So, we have a kind of fun example here, and the idea is, is life fantastic? Is life fantastic, life is fantastic if, and here's the logic here. Life is fantastic if you are rich.

And, logical and, you have a partner, girlfriend or boyfriend, and you have a flat or a house.

If all three of those are true, then the result of logical and is gonna be true. In other words, life is gonna be fantastic.

So, what happens if you run this code, what would you see? You'll see a message come up, and that message will be, life is fantastic is, false. Right? If you run this code and you get to hear, you're gonna get a result of false. This is a bit disappointing, life is not fantastic at the moment. Why? Because remember logical and gives you a true result when the inputs are true. But we have one input which is not true. All right? So we've got two inputs which are true and one input which is false. So, that's this input here. You are not rich. That means that Life is not fantastic, because you need all the inputs to be true for the results to be true.

Now, what happens if we change that variable? You are rich, let's change it. Now, you are rich is true. Okay, let's do our test again. life_is_fantastic = you_are_rich && you_have_partner && you_have_flat. Test them all together, losing logical and, and look at the result. Now the result will be true because the second one is true, the third one is true, and now the first one is also true. Remember, all inputs must be true for the results to be true of logical-AND. Okay, so that's our first little example.

Now there's an interesting feature with JavaScript, and also some other programming languages. Let's say it is going to do a test. Something logical and something. If that first part is the result of some code, like here, there's some code, giving us the first part. And also, there's some code, giving us the second part.

Then JavaScript does a clever thing. It says okay, something logical, and something. So it knows. Well, both inputs must be true for the result to be true. So, if the first input is true, then it will continue and to the second input because both inputs must be true.

So, what does that mean? Here, we'll get this message here, and also, we'll get this message here. This one returns a true, so that is the same as true. So it must continue and do the second test as well. It runs the second code and it runs false, and then we get an answer. So here, both these functions are executed.

Now, interesting surprising thing. What happens if you swap them. So, let's do this one logical, and this one. This first one is a false result.

Now, JavaScript's clever, it says, well, hang on. The first input is false. And I know that both inputs must be true for the result to be true, so I know straightaway I don't even have to look at this. I don't have to run this. I know the results will be false.

So surprising thing is it doesn't even execute this. This function here is not even executed at all in any way because the first part was giving it a false so it knows it's gonna waste it's time.

Now that can be a bit surprising. Here, it's just a kind of a funny little thing. But if this is doing some serious work, maybe accessing a database, updating variables, something like that, then you may not realize that function is not executed at all. Because the first thing returned the false. So, you have to be a bit careful when you do programming.

Okay, let's go on to logical OR, so logical OR, this is how you type it. Two lines in JavaScript. Logical

OR is different to logical AND logical OR, the result is false if both inputs are false, otherwise the result is true. Okay, that's different to and.

And again you could summarize it like this, those are the possible combinations going in to the or and then there's the results. And again, if both inputs are false, the result is false. Otherwise, the result is true.

Okay, so here's an example, and now we're gonna try to find is life good, we're not saying, is life fantastic, life fantastic, the previous example, three things had to be true for life to be fantastic. This one, no, we just We're not being greedy. We're just trying to get one of these things to be true. So we're using logical or. If any of them is true, one is true or two is true or three is true, great. Then if we do logical or. Life will be good but not will be true. If one or more is true so let's do a test you are rich or you have a partner or you have flat. If one of those is true, then the result will be true otherwise it will be false. So is one of those true? Yes. The second one is true. Unfortunately the other two are not true, but the second one is true. So if we run this line of code, and then we check the result, we are gonna get a true coming out. Life is good. Yes, we're gonna get a true. Because one of more of them were true, so the result is true when you use logical or. All right, now what happens if something happens, you lose your girlfriend, boyfriend, whatever. And now, you have partner is false.

All right, now we have false, false, false because that second one has changed. We do our check again. Check the first input or second input or third input. If any of them is true, the result is true. But now all of the inputs are false. So now the results of this are false. Life is good. No, that is false now. And if we check the result that will come up with a false, because all of them are false. You're not rich, you don't have a partner, and you don't have a flat. Life is good is false.

All right, now, what about that funny thing that we saw before? We just call SHORT-CIRCUIT. Is there a similar thing with or? Yes, there is.

So here's an example program, and we're testing one Item or another item. So it does apply. It's a similar idea. Remember, both inputs must be false for the result to be false, because we're dealing with logical or.

So what happens if the first one is true? If the first one is true, then there's no need to do the second one, it's a waste of time. So, it will not do this second piece of code, it will not do this at all. It won't even start it. Because both inputs must be false for the result to be false. And the result is true in this example. So I know the result is gonna be true, because it's logical or. So this will not be executed at all, so you have to be a bit careful about that, if you're doing some programming.

All right what about Not. We've looked at and/or, the other one is Not, and you right it like this in JavaScript, exclamation mark.

Very simple, it does the opposite. So if you give it one input, because it only uses one input and then you do not, logical not. Then you're gonna get the opposite. Give it a false, it gives you a true. Give a true, it gives you a false. it gives you the opposite. So here's a simpler example. Some variables. Lets say you are male is true. Just for example. Then, what about you are female. Well, logically, if you are male, then you are female is false. But we don't have to write the word false. We could say, you are female is opposite, is not, of you are male.

Your female is not you are male. Now the clever thing about that piece of code is if I edit the code, I

change this to force. Then immediately, this one will be automatically updated. I don't have to change this line of code at all because it is guaranteed to be the opposite of this value.

That's the type of part using logical not. In this example, it will automatically make sure that this variable, your female, is always correct.

If this is true, your female will be false. If this is false, your female will be true.

All right, that's the end of our discussion about booleans in JavaScript.

10. Arrays

Arrays are an important element of programming in any language and JavaScript also has them. So let's have a look at arrays in JavaScript. So, you'll understand and appreciate this idea of arrays and also look at some common ways of handling arrays in JavaScript. These are the different functions we're gonna look at.

注意數組是用的[], 而不是{}
如 var arr = [1, 2, 3];

So array, what is an array? Basically, it is the idea of a series of boxes, if you wanna think in a simple way. And in JavaScript, you can put anything you like in those boxes, okay? So there's an order to the boxes, and the order starts from 0. So it goes 0, 1, 2, and so on. So that's the basic idea of an array, so let's see an example in JavaScript. Here's an array with three elements inside it. There's element 0, element 1, element 2, they're very common in computer science, to start counting from 0, and that's what we do here, 0, 1, 2. And so that is an array with three elements inside it. Create the array and put stuff inside it on one line of code. Here is another example. We're creating an array. It has ten items, but we haven't actually put anything in that array, okay? So there's just 10 empty boxes, if you like.

So let's look at some things you can do with arrays. Let's start with join.

So join is basically showing the content of an array. So there's an array, like this. And you say, okay, please join all of those elements together to produce some text, a string. And when you join them together, put this word between each of those items, okay? So it's going to be Dog and Cat and Rabbit. That is the final result, which is shown here, Dog and Cat and Rabbit, okay? So it's joining together all those separate elements.

Now, you could say, okay, take that array and then join them together, and don't have any particular parameter here at all. And if you do that, it will do Dog, Cat, Rabbit. In other words, it uses a comma as the default separator, okay? So nothing there, it's gonna use a comma there, a comma there, okay? So that's join. It's useful way to convert an array into a text, a list of items.

So let's look at getting out of the array. So here's our example array with three elements. Let's extract item 2. So there's the name, square brackets. It has to be square brackets. Item 2. Item 0, item 1, item 2. It will extract item 2, which is Rabbit. So that's how you get something out of the array. It's still in the array but that's how you extract it, so you know what it is.

Okay, so let's try that, let's change it. Start with our array like this, and let's change it. So, we say go to that array called pets. Find item 2 and then change it to Rabbit. Okay, 0, 1, 2, that is now going to

become Rabbit because we just changed it. So now the whole result will be Dog, Cat, Rabbit, okay? So we can extract things. We can change things very, very easily. Another thing that we can do is ask JavaScript how many things are inside that array. So here's our array. It's got three items. We can see that. Of course it could have a million items, there's no reason why not. And we wanna check, okay. That array name pets, pets.length. That will tell us how many things we have in our array. In this particular example, it will show us three.

All right, so very common, you want to manipulate an array. So here's one of the things you might want to do, add something to the end. So one way to add something to the end is to push to the end of the array. So you take the array name .push. And then you write whatever you want to the add to the end of the array. So now after that, we have a result, Dog, Cat, Rabbit, Hamster, because we've added an extra item on the end. And so the indexing would be 0, 1, 2, 3. Hamster will have item 3, okay? So it's just common sense.

Now that was adding at the end. You can also add at the start. And the way to do that is unshift, okay? So take that array name, unshift by adding Hamster. So in other words, that will push everything over to the right and then make a little space, and then Hamster goes into the first element. So Dog, Cat, Rabbit will now become, after this, it'll become Hamster, Dog, Cat, Rabbit. Okay, we've added it at the start.

Okay, what about deleting things? All right, well let's look at removing from the back, from the end. And so one way to do that is to pop. You take the array name and you say pop the last item, by default, it's the last item. So get the last item and throw it out of the array. So we've got these three things. And then we do pets.pop. And then this item will be taken out of the array, okay? In fact, it will be taken out of the array, and if you want to, you can actually use it. The result will have the word Rabbit inside it. Take it out, put it in here, there's Rabbit. It's not in the array anymore.

All right, what about the opposite, removing from the front?

So removing from the front is shift. We saw unshift before, here is shift. So here's our array, three items, let's do a shift. That means grab Dog and then take it out of the array. So now it's just got Cat and Rabbit left. And then, if you want to, you could use it. So we took out Dog, and you could put it in a variable and do something with it, if you like. The main point is, it's taken out of the array. The first item is taken out.

All right, now sometimes you have operations with more than one array. Let's look at two arrays. And sometimes you want to combine them.

So the way to do it is you say, array name .concatenate, you just type concat, and then the second array, okay? Concat is short for computer word concatenate, which just means stick together one after the other, all right? So, we're gonna merge, if you like, these two arrays together. We're gonna stick them together. So here's our first array right here. We've got four items in our first array. Here's our second array right here. We've got five items in our second array. We wanna merge them together.

Okay let's take this one, pets, and then add on at the end the other one, which is called primes. There we are, primes there. So that will add primes onto the end of the first one, the first array. So now we're gonna get Dog, Cat, Rabbit, Hamster, 2, 3, 5, 7, 11, all stuck together in one single array.

Okay, so those are some common ways to handle arrays in JavaScript.

11. Generating Random Numbers

Let's have a quick look at generating random numbers with JavaScript.

So after this presentation you'll be able to generate and manipulate random numbers. These are the two main pieces of code we're gonna have a look at.

So, overview, we are gonna generate a random number, and set up a range, and if you need to, throw away the decimal place.

So generating a random number with JavaScript is pretty straight forward. It's just Math.random(), that gives you a random number from zero up to one and does not include one, does include zero. Okay, so if you want to write that mathematically it looks like this up to but not including one. So we have a random number from zero up to one, not including one. It might look something like this, very very accurate, but anyway it is a random number. Let's look at the code that made that. Straightforward code. Variable and then Math.random(); put it in the variable, and then show us the random number. Simple. So, that's great, it works. That gives you random number, zero to one.

Usually, you want some different range, zero up to one is not the perfect range for whatever you wanna do, so you want to set up your own range. So the way to do that is basically, multiply. Multiply the random number from zero up to one and multiply it with whatever you want, let's say eight, to produce a particular range which is not zero up to one.

So, you might get something like this, we'll show the code on the next slide. Random number from 0 to 7.9999. Looks a bit strange, but we'll explain it in the code. And yes, we've got a nice random number, 6.512 whatever, whatever. Okay. So that is not zero to one. That's a bigger range. So we did that by taking the random number zero up to one, multiplying by eight. So that is 0 to 0.99999, does not include 1, and then we do times 8. So that means the whole result is 0 to 7.999999, okay? That's what it will produce. It could produce 0, it could produce 7.999. And then we have a simple message just to show us the number. Okay? So this will give us a number, with a decimal place, from 0 to 7.9999999. All right, now perhaps you don't want the decimal place. That's quite common. You want an integer. So, we can simply use Math.floor to throw away the decimal place.

All right, so Math.floor doesn't do any rounding up. It literally just dumps the decimal place. So, if you say Math.floor 2.8 something, something, it will become 2. All right? So, there's no rounding up or rounding down. Okay? It simply throws away the digits after the decimal place.

Simple example. Let's say you want a range 0 to 49. Integers this time, no decimal place. And there's an example.

You might produce something like this. Math.random zero up to one.

Multiply times 50, okay, 0 up to 50, so that includes 49.99999.

And then we could dump the decimal place, Math.floor, that value, and then show the result. So that will give us an integer result and gives us a result in the range 0 to 49. Possibly you want 1 to 50, okay, no problem, you could just take the random number and add 1 to it. If you add 1 to it, then it's 1 to 50, including 1, including 50.

Okay, so that's some examples of how to manipulate random numbers in JavaScript.

12. An Example JavaScript Project (沒看視頻)

Okay, let's have a look at an example project.

After looking at this project you should have stronger JavaScript skills. These are all the different functions and code which we're gonna use in this project. And the basic idea is to put together a lot of the techniques and code that we've seen before, and to do that we'll build a guessing game.

So let's play the game. Okay so we are playing the game, and you can see it says, I am thinking of a number in the range 1 to 100. What is the number? Okay, so you have to think what number the computer is thinking of. Lets try, I don't know, 73, something like that. And it tells you, no, the number is too large that I've sent, so it's got to be smaller than 73. Press Okay, I get another chance. I'm think of a number in the range 1 to 100. What is the number? It must be less than 73. Okay, let's try 44. And the number is too large. Okay, it's less than 44. Now, of course, I could be a little bit more scientific than just typing in some of these numbers. Like I could do 50 and 25, but let's pretend that we are not thinking in a logical way. Let's say I do 44, let's do a lower number. Let's say 11, something like that. What does it say? Your number is too small. Okay, so it's somewhere between 11 and 44. All right, well let's try maybe 30. Okay, number is still too small. So clearly, it's between 30 and 44.

What's the number? Let's try 40. Number is too large. Okay, so we can begin to narrow down the number. Okay, 35 is too large so it must be between 30 and 35. Let's try 33. Number is too small. All right. Then, we don't have many numbers left. It should be 34. Yes, you got it. The number was 34. It took you 8 guesses to get to the number. All right, that is the whole idea of the game. It's computer thinks of a number, you have to get the answer as quickly as you can in as few guesses as possible.

So, let's look at how we built that game.

Okay, so this is what we saw when we started. We see this little window, and then it lets us enter the number. And then we obviously have some intelligence, all handled by JavaScript for processing what number we enter.

So we get the idea, it thinks of a number in the range 1 to 100, including 1 including 100. And you have to guess and it gives you some feedback, and after several times, eventually, you can lose use logic and narrow down and get the right answer.

So, what we have is actually a flow diagram like this for handling the logic of the game.

And so I’m gonna go through this. Basically, first of all, we generate a random number in the computer’s memory, browser memory, we store it in a variable, target, okay. So that’s what the player, the person using the game, has to get. He has to get that number in target.

So then you get some input from the player, and you store it in another variable, guess_input.

And so then we look at guess_input, we look at that number and then we go through and check. We have a series of checks and if it passes all of those checks, you got it correct, okay. But there are several situations where it could fail.

So let's look at the first one. So the first one, let's look at what the person typed in. Is what the person typed in, not a number? This is a number guessing game, they can't type in infinity or something like that or I don't want to type a number. They cannot type in text, okay. So, if they type in text, or something silly, then give them some feedback, enter a number, and then we loop back, okay. What's the second test? Second test is if the input which the person typed in, it was a number, however it was a number in the wrong range. Our gain is from 1 to 100. If they entered -50 or +300, then we want to get another message, wrong range.

All right, now if it passes that second test then we go to our third test. Was the number typed in by the player, was it bigger than the number being thought of by the computer? If it was then little message, too high, your number is too high, and then we loop back again. And then our fourth test, if the number entered by the player was smaller than the target number. Then another message, too low. Back we go looping again, until we have a number entered by the user which passes all of those four tests. If it passes all of those four tests, then they are correct. They are finished. They have got the answer. So there's the flow diagram, which we have implemented in JavaScript. So let's have a look, so first of all the HTML part. Well, as you can see, there's really no HTML, it's really just those little alert boxes and prompts, those little windows, so there's very little HTML. And so we, just for this example, we put the HTML in a separate file like this, and then we just linked to the main JavaScript like this. So the browser will bring in the JavaScript from another file. The only clever bit is when the webpage loads, then we start one of those functions which is in that other file, okay, but mostly, there's no HTML.

You could add HTML if you want to, and do some text and things, but we have just done a simple version of the game. Okay, so you've looked at those webpage loads, triggered the first main piece of code, the main function, and that is actually in a separate file.

So, if we look at that file we'll see these components, and we're gonna look at it in a minute. So we've got global variables and we've got actually two functions, okay. These two functions work together to play the game. So this is the first function, do_game.

What does it do? Generates a random number in a particular range. And then has a while loop which repeats until the player gets the right answer, okay. So that's the main game function, we'll see it in a minute. And then here we've got another function, check_guess. The user does a guess inside the while loop, and then we have to check, is it correct? Remember we have four tests. So that is in another function check_guess. And so this function, the main game loop, that calls this function to check what the person has typed in, and also gives some feedback. So these two functions are working together. So, let's have a look at those functions and we'll go through those different elements. So let's have a look. So, here we have the code. And at the top we have our global variables, and which have been created. Some of them are given initial values like this. The game has not finished, so we put finish into that variable. And, remember, we have to remember how many times the player has played the game in a loop. So, we remember it with this variable, and I just put 0 in there at the start. This program I'm using is Notepad++, which is on a PC, but you can use any kind of editor.

So, here is our first game function. This is the main game function, do_game.

So what happens? Well we just explained it a few seconds ago, but here's the actual code. So the first one is generate a random number from, well so far this would be from 0 to 99.99999. That's what this particular line of code does. We've seen this before. And then we could throw away the decimal place. So now it will be 0 to 99 without a decimal place. We've seen that before. And finally, let's add 1, and

that means the range is gonna be 1 to 100, including 1, including 100. That is the target number the player has to guess. All right so, after that, we go into our main loop. So our main loop says, all right, keep repeating while we have not finished. Remember, this little exclamations mark is JavaScript way of saying, not. While we have not finished, then we have our loop, okay.

Our loop will finish when second function returns true. That second function is responsible for checking whether the user has got a correct result. When the user enters a correct result, true comes back to here and the entire loop finishes and doesn't get executed anymore.

What's inside the loop? Well, we've got a little message here and that's what gets shown to the player. I'm thinking of a number in the range 1 to 100, what is the number? And that is using the little prompt window. And whatever they type in goes to a variable.

And remember when you use prompt it returns text. So, we have to convert our text into integer. So we take the text, convert it into integer, that's what parseInt means. It means process the text and convert it to an integer number, a number without a decimal place. And then put the result here.

Now we're keeping track of how many times the person guesses. So we keep track of it in here. And that variable increases by one each loop. Then another function, as I mentioned, will check whether or not the correct answer has been entered. And so we repeat this loop until the correct answer has been entered.

Now let's look at that function, the second function.

Here it is, and it's really the same as the loop which we just sowed where we do four checks. So here's the first check.

Is the input entered by the player not a number? NaN, you can see this capital N, capital N. You have to do it exactly like that. Is it not a number, okay?

So, for example, if I type in big, or small, or happy, or something like that then this function will say, no that is not a number, and it will return true.

So, if the user types in something which is not a number, we show an appropriate message and then we return false. That means the user has failed to check correctly. Return immediately stops the entire function, there's nothing else is executed, that will stop the function right there.

Then, it goes back to the main loop.

All right so, second check. Second check, is it in the correct range?

So there's two tests here, really. If the guess entered by the person is negative or 0, or the guess entered by the person is bigger than 100, it is out of range. So, let's show a message. No, no, it must be in the correct range, like 1 to 100. And that is another fail situation, they have failed to guess the answer correctly. So we do return false, and that will immediately stop this function check guess.

Third check, okay here it is, third check. If the number entered by the user is bigger than the actual number in the computer's memory. All right, your number is too large and then it's still a failure, so we stop the function and we return false indicating still a failure to guess the answer.

This one, the final check. If the input entered by the player is smaller than the target, all right, then your number's too small. And again, you have failed to guess the answer, stop the function, and return false to the main game loop.

All right. Now if we've got this far, then we are happy because we have actually got the correct answer. We don't even have to do an if statement to check it. If it got through all the other checks, then it must be the correct answer. So, show a message, you got it, very good, the answer was this.

And also we can show how many times they typed in a number, just for fun really, and this is a success situation so we return true. Okay, that goes back to the main game loop, let's check our main game loop, here it is and that goes back here. The trues or the falses, they come back to this variable and then they get checked in the next iteration of the while loop, okay. So, let's play the game once again just to remind ourselves what's going on. So here we are once again. I'm thinking of a number. This will be a different random number to before. What is the number? Well, I suppose we could be very logical and we could try to divide the possible numbers into two. 50 is the number. Okay, it's too small. All right, well what is the number? Let's try 75. Okay, well it's too large. All right, between 50 and 75. Now let's try some of the other tests we saw. Let's try, what happens if I type in something which is not a number, and a word is not a number, okay. Infinity is not a number. Okay, so what happens? You have not entered a number. Please enter a number in the range 1 to 100. Okay, same for something else like in a no, no you have not entered a number. It keeps repeating that if I enter in some kind of numbers, even if I use this kind of idea. Nine, if I write it with English, it also will be rejected. It's not a number as far as JavaScript is concerned.

So where were we? Back to our game. What about outside the range? Let's do that check as well. What about if I say -4? Then it says no, please enter an integer in the range 1 to 100. All right, let's try 102. Okay, again, same message, please enter an integer in 1 to 100 range.

All right, so back to playing the game. So, we've got some number, let's try 60, and it's too small. All right, and then I think it was 75 was the highest so far. So let's do, I don't know, 67. And then, it's too small. All right, let's try 72 or something. It's too large. Let's try 70, it's too large. So let's try, I don't know, 69, whatever. You got it. The number was 69. It took you 12 guesses to get the number, which is not very good, okay. All right, so that is our little game that we've put together. You can play it on the website, we've discussed all the different elements, so that is a good place to finish right there.