What is JavaScript?

JavaScript is a programming language that adds interactivity to your website. This happens in games, in the behaviour of responses when buttons are pressed or with data entry on forms; with dynamic styling; with animation, etc.

P.S. It ws invented by Brendan Eich (co-founder of the Mozilla project, the Mozilla Foundation, and the Mozilla Corporation).

It is versatile and beginner-friendly. You can create games, animated 2D and 3D graphics, comprehesive database-driven apps and much more!

It is relatively compacy, yet very flexible. Developers have written many tools on top of the core JavaScript language, unlocking a vast amount of functionality with minimum effort. For example:

- Browser Application Programming Interfaces (APIs) built into web browsers, providing functionalitty such as dynamically creating HTML and setting CSS styles; collecting and manipulating video stream from a user's webcam, or generating 3D grpahics and audio samples.
- Third-party APIs that allow developers to incorporate functionality in sites from other content providers, such as Twitter or Facebook
- Third-party frameworks and libraries that you can apply to HTML to accelerate the work of building sites and applications.

A Hello world! example

To get comfortable with Java Script, you have to start small and progress gradually. Use the following steps to add JavaScript yo your age for creating a Hello world! example.

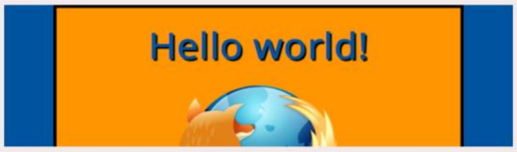
1. Go to test site and create a new folder named scripts. In this folder, create a new file called main.js, and save it
2. In your index.html file, enter this code on a new line, just befroe the closing </body> tag:

```
<script src = "scripts/main.js"></script>
```

3. This is doing the same job as the <link> element for CSS, It applies the JavaScript to the page, so it can have an effect on the HTML (along with the CSS, and anything else on the page).
4. Add this code to the main.js file:

```
const myHeading = document.queryselector('h1');

myHeading.textContent = 'Hello world!';
```

5. Make sure the HTML and Javascript files are saved. Then load index.html in your browser. You should see something like this:



NOTE: The reason the instructions place the </script> element near the bottom of the HTML file is that **the browser reads the code in the order it appears in file.** If the JavaScript loads first and is supposed to affect the HTML that hasn't loaded yet, there could be problems. Placing JavaScript near the bottom of an HTML page is one way to accomodate this dependency. To learn more about alternative approaches, see Script loading strategies .

What happened?

The heading text changed to Hello world! using JavaScript. You did this by using a function called querySelector() to grab a reference to your heading, and then store it in a variable called myHeading. This is similar to what we did using CSS selectors. When youwant to something to an element, you select it first.

Following that, the code set the value of th myHeading variable's textContent property (which represents the content of the heading) to Hello world!.

Language basics crash course

The following are some core features of JavaScript to better understand how it works.

Variables

Variables are containers that store values. You start by declaring a variable with the var (less recommended, dive deeper for the explanation or the let keyword, followed by the name you give to the variable:

```
let myVariable;
```

NOTE: A semi-colon at the end of a line indicates where a statement ends. It is only required when you need to separate statements on a single line. There are other rules for when you should and shouldn't use semicolons. For more details, see Your Guide to Semicolons in JavaScript

NOTE: You can name a variable nearly anything, but there are some restrictions.(See this section about naming rules.) If you are unsure, you can check your variable name to see if it's valid.

NOTE: JavaScript is case sensitive. Ths means myVariable is not the same as myVariable. If you have problems in your code, check the case!

NOTE: For more details about the dfference between var and let, see The difference between var and let.

After declaring a varable, you can give it a value:

```
myVariable = 'Bob';
```

Also, you can do both there operations on the same line:

```
let myVariable = 'Bob';
```

You retrieve the value by calling the variable name:

```
myVariable;
```

After assigning a value to a variable, you can cn=hange it later in the code:

```
let myVariable = 'Bob';

myVariable = 'Steve';
```

Note that variables may hold values that have different data types :

Variable	Explantion	Example
String	A sequence of text known as a string. To signify that the value is a string, enclose it in single quote marks	let myVariable = 'Bob';
Number	This is a number. Numbers don't have quotes around them.	let myVariable = 10;
Boolean	This is a True/False value. The words true and false are special keywords that don't need quote marks.	let myVariable = true;
Array	This is a structure that allows you to store multiple values in a single reference..	let myVariable = [1,'Bob','Steve',10]; Refer to each member of the array like this: myVariable[0], myVariable[1], etc.
Object	This can be anything. Everything in JavaScript is an object and can be stored in a variable. Keep this in mind as you learn.	let myVariable = document.querySelector('h1'). All of the above examples too.

Variables are necessary to do anything interesting in programming. If values couldn't change, then you couldn't do anything dynamic, like personalize a greeting message or change an image displayed in an image gallery.

Comments

Comments are snippets of text that can be added along with code. The browser ignores text marked as comments. You can write comments in JavaScript just as you can in CSS:

```
/* Everything in between is a comment */
```

If your comment contains no line breaks, it's an option to put it behind two slashes like this:

```
// This is a comment
```

Operators

An operator is a mathematical symbol that produces a result based on two values (or variables). In the following table, you can see some of the simplest operators, along with some examples to try in the JavaScript console.

Operator	Explantion	Symbol(s)	Example

Addition	Add two numbers together or combine two strings.	+	6 + 9;
			'Hello' + 'world!';

Subtraction, Multiplication, Division	These do what you'd expect them to do in basic math.	-, *, /	9 - 3 ;
			8 * 2 ; // multiply in JSS is an asterisk
			9 / 3;

Assignment	As you've seen already: this assigns a value to a variable.	=	let myVariable = 'Bob';

Equality	This performs a test to see if two values are equal. It returns a true/false (Boolean) result.	===	let myVariable = 3;
			myVariable === 4;

Not, Does-not-equal	This returns the logically opposite value of what it precedes. It turns a true into a false, etc..
	When it is used alongside the Equality operator, the negation operator tests whether two values are not equal.
		!, !==	For "Not", the basic expression is true, but the comparison returns false because we negate it:
			let myVariable = 3;
			!(myVariable === 3);
			"Does-not-equal" gives basically the same result with different syntax. Here we are testing "is myVariable NOT equal to 3". This returns false because myVariable IS equal to 3:
			let myVariable = 3;
			myVariable !== 3;

There are a lot more operators to explore, but this is enough for now. See Expressions and operators for a complete list.

Conditionals

They are code structures used to test if an expression returns true or not. A very common form of conditionals is the if ... else statement. For example:

```
let iceCream = 'chocolate';

if(iceCream === 'chocolate') {

alert('Yay, I love chocolate ice cream!');

} else {

alert('Awwww, but chocolate is my favorite...');

}
```

The expression inside the if (...) is the test. This uses the identity operator(as described above) to compare the variable iceCream with the string chocolate to see if the two are equal. If this comparison returns true, the first blovk of code runs, If the comparison is not true, the second block of code - after the else statement - runs instead.

Functions

Functions are a way of packaging functionality that you wish to reuse. It's possible to define a body of code as a function that executes when you call the function name in your code. This is a good alternative to repeatedly writing the same code. For example:

1. let myVariable = document.querySelector('h1');
2. alert('hello!');

These functions, document.querySelector and alert, are built into the browser.

If you see something which looks like a variable name, but it's followed by parentheses— () —it is likely a function. Functions often take arguments : bits of data they need to do their job. Arguments go inside the parentheses, separated by commas if there is more than one argument. For example, the alert() function makes a pop-up box appear inside the browser window, but we need to give it a string as an argument to tell the function what message to display. You can also define your own functions. In the next example, we create a simple function which takes two numbers as arguments and multiplies them:

```
function multiply(num1,num2) {

let result = num1 * num2;

return result;

}
```

NOTE: The return statement tells the browser to return the result variable out of the function so it is available to use. This is necessary because variables defined inside functions are only available inside those functions. This is called variable scoping .

Events

Real interactivity on a website requires event handlers. These are code structures that listen for activity in the browser, and run code in response. The most obvious example is handling the click event, which is fired by the browser when you click on something with your mouse. To demonstrate this, enter the following into your console, then click on the current webpage:

```
document.querySelector('html').onclick = function() {
alert('Ouch! Stop poking me!'); }
```

There are many ways to attach an event handler to an element. Here we select the <html> element, setting its onclick handler property equal to an anonymous (i.e. nameless) function, which contains the code we want the click event to run.

NOTE: The return statement tells the browser to return the result variable out of the function so it is available to use. This is necessary because variables defined inside functions are only available inside those functions. This is called variable scoping .

Events

Real interactivity on a website requires event handlers. These are code structures that listen for activity in the browser, and run code in response. The most obvious example is handling the click event, which is fired by the browser when you click on something with your mouse. To demonstrate this, enter the following into your console, then click on the current webpage:

document.querySelector('html').onclick = function() {
alert('Ouch! Stop poking me!'); }

There are many ways to attach an event handler to an element. Here we select the <html> element, setting its onclick handler property equal to an anonymous (i.e. nameless) function, which contains the code we want the click event to run.

NOTE:

```
document.querySelector('html').onclick = function() {};
```

is equivalent to

```
let myHTML = document.querySelector('html');

myHTML.onclick = function() {};
```

Supercharging our example website

Let's add a few features to our example site.

Adding an image changer

Here you will learn how to use JavaScript and DOM API features to alternate the display of one of two images. This change will happen as a user clicks the displayed image.

1. Choose an image you want to feature on your example site. Ideally, the image will be the same size as the image you added previously or as close as possible.
2. Save this image in your images folder
3. Rename the image firefox2.png.
4. Add the JavaScript below to your main.js file

```
letlet myImage = document.querySelector('img');

myImage.onclick = function() {

    let mySrc = myImage.getAttribute('src');

    if(mySrc === 'images/firefox-icon.png') {

    myImage.setAttribute('src','images/firefox2.png');

    } else {

    myImage.setAttribute('src','images/firefox-icon.png');

    }

}
```

5. Save all files and load index.html in the browser. Now when you click the image, it should change to the other one

This is what happened. You stored a reference to your <img> element in the myImage variable. Next, you made this variable's onclick event handler property equal to a function with no name (an "anonymous" function). So every time this element is clicked:

1. The code retrieves the value of the image's src attribute.
2. The code uses a conditional to check if the src value is equal to the path of the original image:

1. If it is, the code changes the src value to the path of the second image, forcing the other image to be loaded inside the <img> element.

Adding a personalized welcome message

Next, we will change the page title to a personalized welcome message when the user first visits the site. This welcome message will persist. Should the user leave the site and return later, we will save the message using Web Storage API . We will also include an option to change the user, and therefore, the welcome message.

1. In index.html, add the following line just before the <script > element:

```
<button> Change user </button>
```

2. In main.js, place the following code at the bottom of the file, exactly as it is written. This takes references to the new button and the heading, storing each inside variables:

```
    let myButton = document.querySelector('button');

    let myHeading = document.querySelector('h1');
```

3. Add the function below to set the personalized greeting. This won't do anything yet, but this will change soon.

```
    function setUsername(){

    let myName = prompt('Please enter your name.');

    localStorage.setItem('name', myName);

    myHeading.textContent = 'Mozilla is cool,' + myName;
```

The setUserName() function contains a prompt() function, which displays a dialog box, similar to alert(). This prompt() function does more than alert(), asking the user to enter data, and storing it in a variable after the user clicks OK. In this case, we are asking the user to enter a name. Next, the code calls on an API localStorage, which allows us to store data in the browser and retrieve it later. We use localStorage's setItem() function to create and store a data item called 'name', setting its value to the myName variable which contains the user's entry for the name. Finally, we set the textContent of the heading to a string, plus the user's newly stored name.

4. Add the if ... else block(below). We could call this initialization code, as it structures the app when it first loads.

```
    if(!localStorage.getItem('name')) {

    setUserName();

    } else {

    let storedName = localStorage.getItem('name');

    myHeading.textContent = 'Mozilla is cool, ' + storedName;

    }
```

This first line of this block uses the negation operator(logical NOT, represented by the !) to check whether the name data exists. If not, the setUserName( ) function runs to create it. If it exists (that is, the user set a user name during a previous visit, we retrieve the stored name using getItem() and set the textContent of the heading to a string, plus the user's name, as we did inside setUserName( ).

5. Put this onclick event handler (below) on the button, When clicked, setUserName( ) runs. This allows the user to enter a different name by pressing the button.

```
    myButton.onclick = function( ) {

    setUserName( );

    }
```

A username of null?

When you run the example and get the dialog box that prompts you to enter your user name, try pressing the Cancel button. You should end up with a title that reads Mozilla is cool, null . Null is a special value in JavaScript that refers to the absence of a value.

Also, try clicking OK without entering a name. You should end up with a title that reads Mozilla is cool, for fairl obvious reasons.

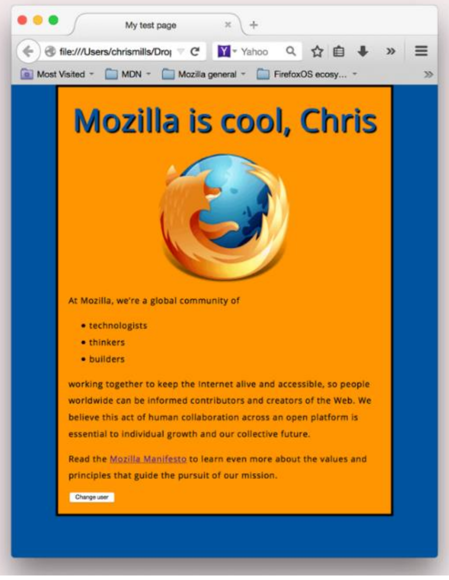
To avoid these problems, you could check that the user hasn't entered a blank name. Update your setUserName() function to this:

```
function setUserName() {

let myName = prompt('Please enter your name.');

if(!myName) {

setUserName( );

} else {

localStorage.setItemsetItem('name', myName);

myHeading.textContent = 'Mozilla is cool, ' + myName;

}

}
```

In human language, this means: If myName has no value, run setUserName() again from the start. If it does have a value (if the above statement is not true), then store the value in localStorage and set it as the heading's text.

Conclusion

If you have followed all the instructions, you should end up with a page that looks something like the image below.



Mozilla is cool, Chris

At Mozilla, we're a global community of

- technologists
- thinkers
- builders

working together to keep the internet alive and accessible, so people worldwide can be informed contributors and creators of the Web. We believe this act of human collaboration across an open platform is essential to individual growth and our collective future.

Read the Mozilla Manifesto to learn even more about the values and principles that guide the pursuit of our mission.