Lesson **Weekend**

Introduction to Programming (/introduction-to-programming) / JavaScript and Web Browsers (/introduction-to-programming/javascript-and-web-browsers) / Using MDN Documentation for JavaScript

Text

With practice and repetition, you'll remember the terminology and syntax of code. However, you won't be able to remember everything, nor should you try. Generally, it's more important to understand concepts and remember terminology so that you can write a good search query. However, even that will take some time to develop. For example, at this point, you may be able to remember how to create a hyperlink in HTML, `<a href="url">link text</a>` but you made need to research it to be sure:

```
How do I create a link in HTML?
```

Similarly, you may remember how to write a CSS rule, but you might forget the technical terminology to describe the parts of a CSS rule. A quick search can help you get this information:

```
What are the parts of a CSS rule?
```

It's okay to not remember everything. It's also good to research what you don't remember, because it builds up your researching skills. Ultimately, we ask that you be patient with yourself and trust that with practice, the concepts and terminology will stick, and whatever you don't remember, you can always look up.

You can also use the lessons in LearnHowToProgram.com to find information, however, LearnHowToProgram.com is not meant to be a complete reference for the languages and tools we teach, so it's important to develop your research skills as we go along.

In this lesson, we'll explore how to use Mozilla Developer Network (MDN) documentation (https://developer.mozilla.org/en-US/docs/Web/JavaScript) to empower you to use this resource effectively. MDN is one of the best sources of documentation on JavaScript. The site includes guides, tutorials, resources, and technical reference for developers using JavaScript.

MDN Documentation

Before we jump in, it's worth covering the different terminology related to documentation. On MDN, we can find the following:

- **Documentation:** this is the umbrella term for written resources that explain a programming language, library, framework, software, or tool. Documentation is often made up of references, guides, tutorials, tools, and resources.
- **Reference:** this is what we call documentation that focuses on explaining syntax, technical terminology, and the structures of a programming language, etc. For example, it is in a reference that we will find a list of all JavaScript data types or all string methods. We'll also find explanations of technical terminology.
- **Tutorials and Guides:** these will teach you how to use a a programming language, etc. These are like "walkthroughs", taking you step by step through new concepts. These often include instructions to build example projects to apply the new concepts.
- **Tools and Resources:** a list of tools to help you along with the technology you are learning. For JavaScript, MDN has compiled a list of tools and resources to write and debug JavaScript code.

MDN Homepage Features and Finding the JavaScript Documentation

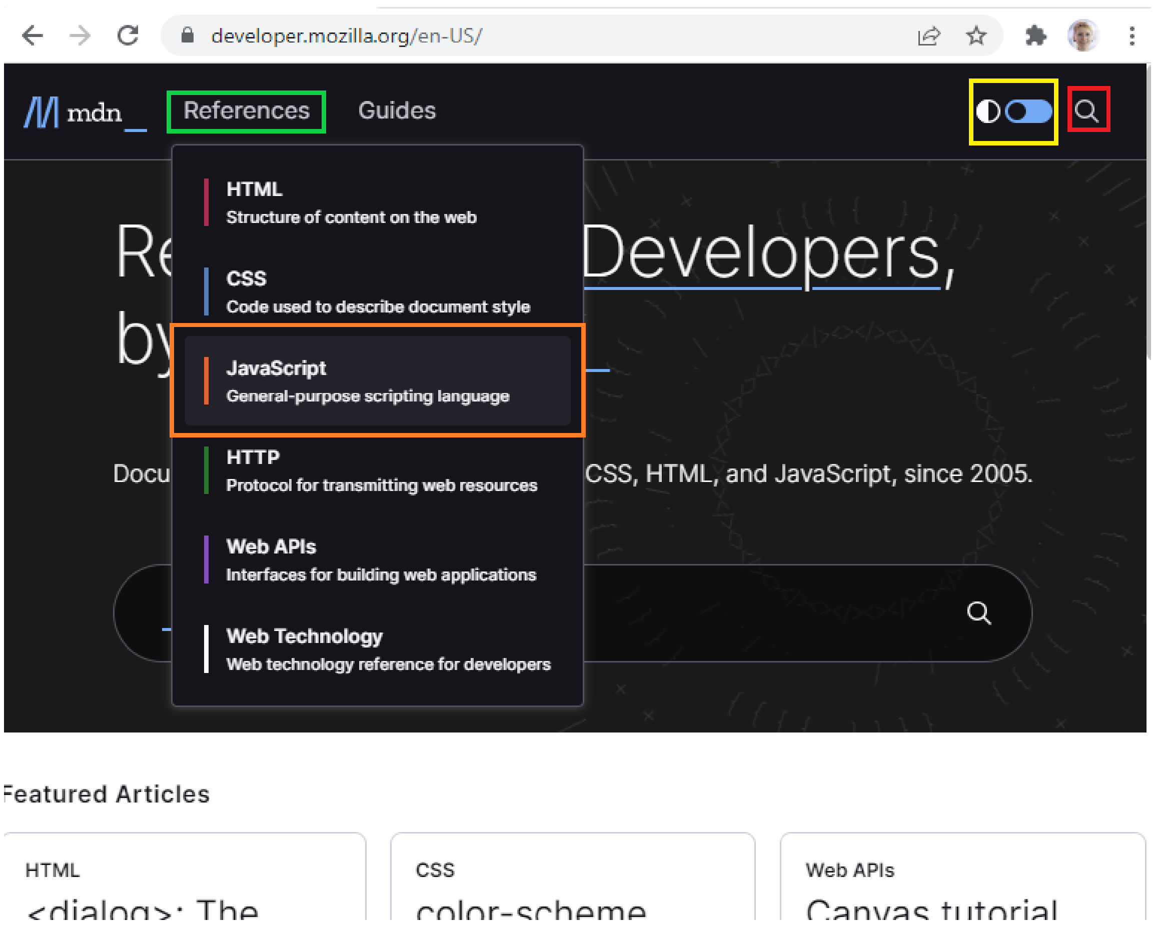
MDN has references for HTML, CSS, JavaScript, and more. We'll focus on working with the JavaScript and Web APIs (more on this later) documentation in this section. From the MDN homepage, you'll be able to find all references from the "References" drop-down menu. To find the *References* menu, follow the instructions below and reference the image of MDN's homepage:

- Go to https://developer.mozilla.org/en-US/ (https://developer.mozilla.org/en-US/).
- Select *References* from the navigation menu at the top left of the window. This is highlighted in green in the image.
- Select *JavaScript*. This is highlighted in orange in the image.

Also note that the image below highlights two additional features located in the upper right hand corner of the window:

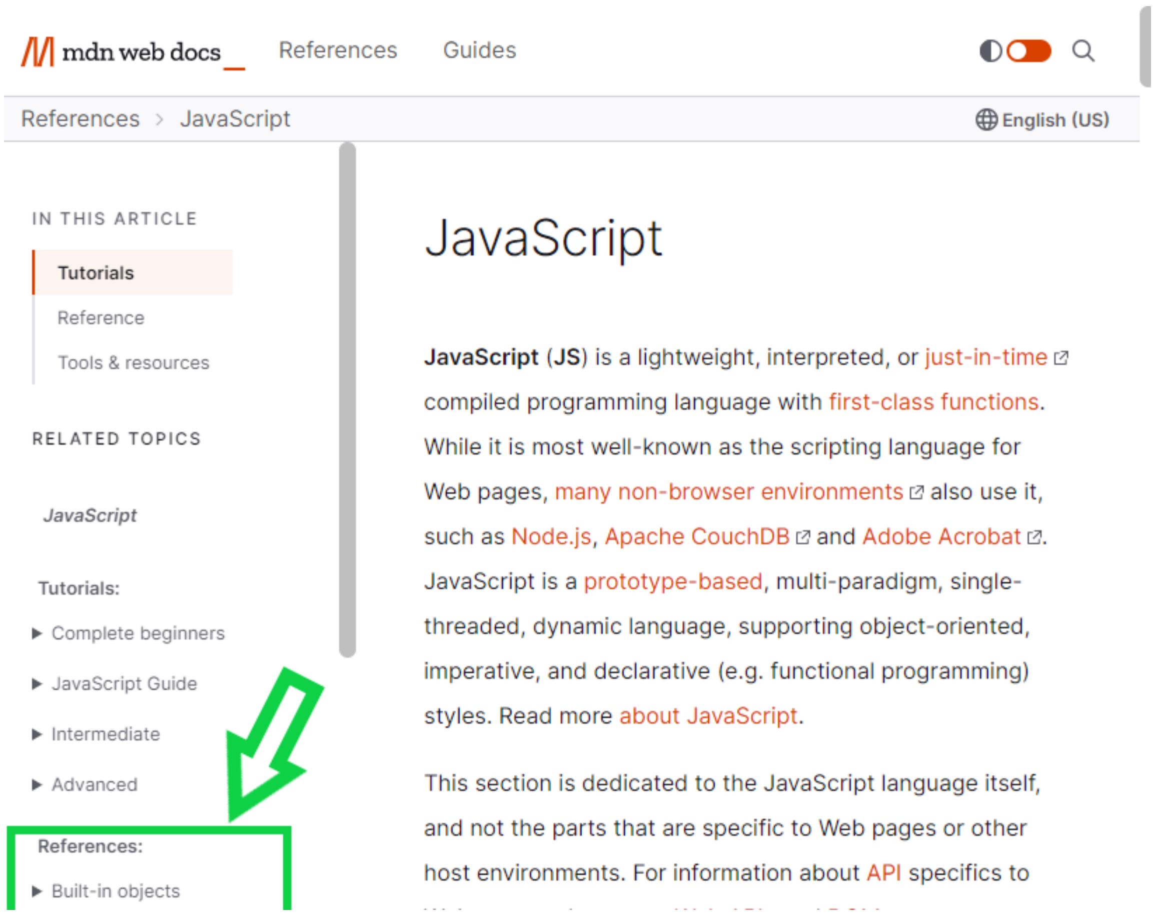
- Selecting your preference for a light or a dark theme, highlighted in yellow.
- Search the site, which is highlighted in red.

The light/dark theme and search features are on every page of MDN.


developer.mozilla.org/en-US/
mdn_
References
Guides
HTML
Structure of content on the web
CSS
Code used to describe document style
JavaScript
General-purpose scripting language
HTTP
Protocol for transmitting web resources
Web APIs
Interfaces for building web applications
Web Technology
Web technology reference for developers
Re Developers,
by
Docu CSS, HTML, and JavaScript, since 2005.
Featured Articles
HTML
<dialog>: The
CSS
color-scheme
Web APIs
Canvas tutorial

Working with the JavaScript Reference

Once on the JavaScript References homepage, note the left vertical menu bar. Here there will be links to tutorials and references. Scroll down to find the references section — this is the documentation that we'll primarily be working with. We've highlighted the start of this section with a green square and an arrow in the image.



JavaScript

JavaScript (**JS**) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles. Read more about JavaScript.

This section is dedicated to the JavaScript language itself, and not the parts that are specific to Web pages or other host environments. For information about API specifics to

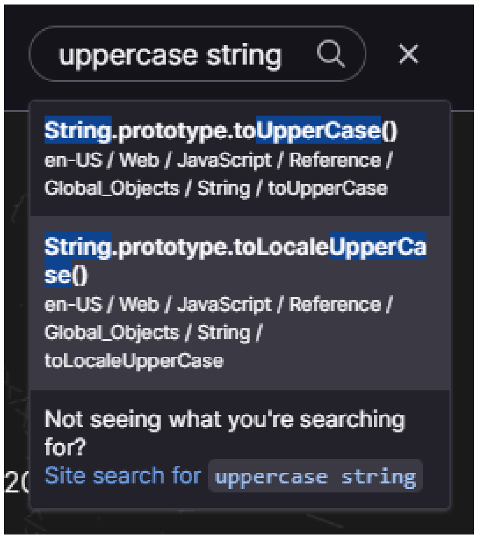
Tips to Using MDN

Use MDN's Built-In Search Functionality

MDN has great search functionality! Check out the image below that shows a search for "uppercase string". This is a good search query — we know we are looking to uppercase a string, and we're including those exact key words. MDN returns a few search hits, which we'll list below and you can also see in the image below:

- `String.prototype.toUpperCase()`
- `String.prototype.toLocaleUpperCase()`
- Not seeing what you're searching for? Site search for `uppercase string`.

If we click the last option, we'll be taken to a search page with more search results. However, in the first two search hits for "uppercase string", we have new, unfamiliar syntax, so let's cover that next.


uppercase string

String.prototype.toUpperCase()
en-US / Web / JavaScript / Reference / Global_Objects / String / toUpperCase

String.prototype.toLocaleUpperCase()
en-US / Web / JavaScript / Reference / Global_Objects / String / toLocaleUpperCase

Not seeing what you're searching for?
Site search for `uppercase string`

Understanding `prototype`

So far we've introduced methods by their data type and by their name, like in the lesson on String and Number Methods (https://www.learnhowtoprogram.com/introduction-to-programming/javascript-and-web-browsers/practice-string-and-number-methods):

String Methods

toLowerCase

Here's the opposite of toUpperCase, making a string all lower case:

```
> "HOWDY, NEIGHBOR".toLowerCase();
"howdy, neighbor"
```

Remember, methods always belong to a specific data type, so it's important for us to always name the data type that a method belongs to, like in the example above. On MDN and elsewhere, the official syntax for the name of any JavaScript method is the following:

```
// This is not real code! This is just to highlight the official syn
tax for naming methods!
dataType.prototype.methodName()
```

What's important about this syntax is that it always includes the data type that the method belongs to in the name of the method. Let's apply this syntax to the string methods we've learned about so far:

- `String.prototype.toUpperCase()`
- `String.prototype.toLowerCase()`
- `String.prototype.concat()`
- `String.prototype.charAt()`

And to the number methods we've learned about:

- `Number.prototype.toString()`
- `Number.prototype.toFixed()`

So what does the term `prototype` mean? For now, all we need to know is when we see `prototype` in the name of a method, it indicates that we're referencing a built-in (JavaScript-created) method that belongs to a specific data type, and any instance of that data type. In other words, we're not actually using (calling) the method — we're just referencing it by its official name.

For example, with `String.prototype.concat()`, we're referencing the `concat` method belonging to strings only. We're also indicating that this exact method belongs to all strings, meaning that `String.prototype.concat()` can be called on any string.

Going forward, we'll use the `dataType.prototype.methodName()` syntax when we're referencing JavaScript methods in order to clearly indicate which data type the method belongs to. This will also align us with the syntax that MDN uses.

Links to Documentation on Number and String Methods

There are many more string and number methods, so let's take a look at that now. Visit these links to see a list of all of these methods:

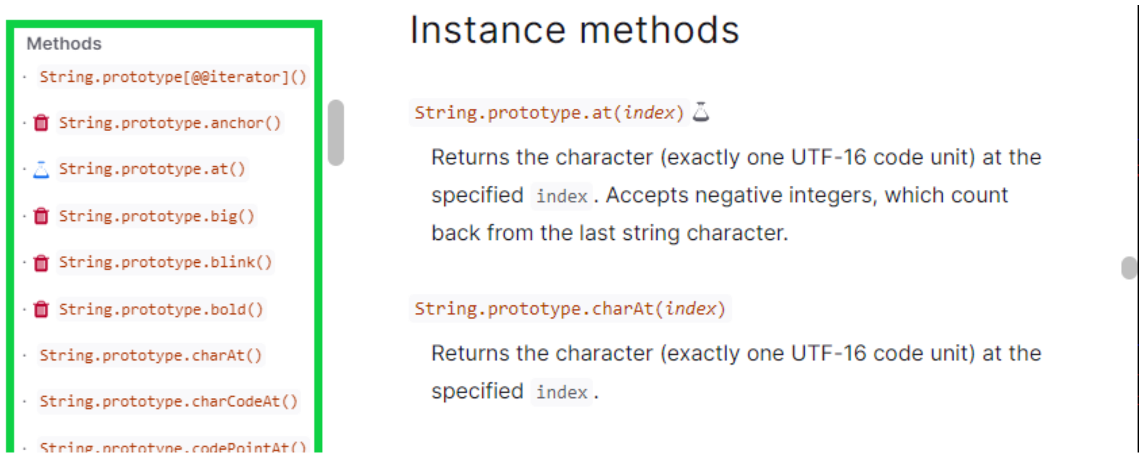
- **String Methods (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String#instance_methods)**

- **Number Methods (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number#instance_methods**

The links above take you to a section called "Instance methods" within the String or Number reference page on MDN. (If the link does not take you to "Instance methods" — scroll down on the page to find it.) All JavaScript methods are divided into two categories, "instance" and "static", and we'll be learning more about these two categories soon. For now, just keep in mind that when we talk about methods, we're talking about **instance** methods.

There are 6 Number methods and 33 String methods (specifically instance methods) that are built-in and available for us to use!

Note the left vertical sidebar menu — here we can see an overview of every topic covered on this page. Scroll down on this menu bar and we'll find a section called "Methods". In the image below, we've highlighted the start of the string "Methods" section in green:


Methods
String.prototype[@@iterator]()
String.prototype.anchor()
String.prototype.at()
String.prototype.big()
String.prototype.blink()
String.prototype.bold()
String.prototype.charAt()
String.prototype.charCodeAt()
String.prototype.codePointAt()

Instance methods

`String.prototype.at(index)`

Returns the character (exactly one UTF-16 code unit) at the specified `index`. Accepts negative integers, which count back from the last string character.

`String.prototype.charAt(index)`

Returns the character (exactly one UTF-16 code unit) at the specified `index`.

In the left sidebar menu we can view all methods at a glance by name.

Some methods have icons next to them: a trash can, thumbs-down, or chemistry beaker. You can hover your mouse pointer over them to get the meanings: deprecated, non-standard, and experimental. These icons help developers quickly know major concerns about methods:

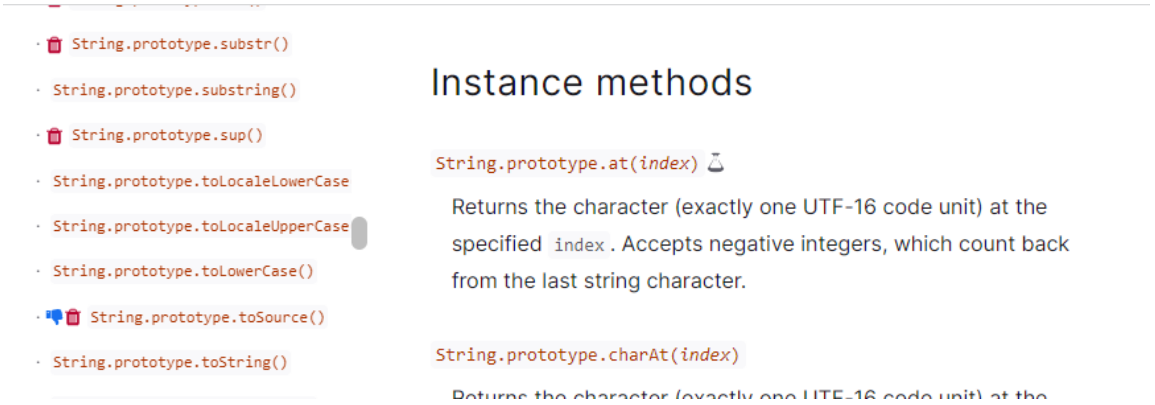
- **Deprecated** (trash can icon) means that the method is no longer in use, and is not recommended for use.
- **Non-Standard** (thumbs-down) icon means that the method may not work for everyone on every platform, so it's not recommended for use.
- **Experimental** (chemistry beaker) means that the method is new and currently in the process of being added or being considered to add to

JavaScript.

See the image below that shows these icons applied to three methods:

- The first method on the page underneath the "Instance Methods" header, `String.prototype.at(index)`, is experimental.
- On the vertical left menu, we can see that `String.prototype.sup()` is deprecated.
- Also on the vertical left menu, we can see that `String.prototype.toSource()` is both deprecated and non-standard.

We recommend staying away from deprecated, experimental, and non-standard methods while you are learning JavaScript. However, it's up to you if you want to explore more.


String.prototype.substr()
String.prototype.substring()
String.prototype.sup()
String.prototype.toLocaleLowerCase
String.prototype.toLocaleUpperCase
String.prototype.toLowerCase()
String.prototype.toSource()
String.prototype.toString()

Instance methods

`String.prototype.at(index)`

Returns the character (exactly one UTF-16 code unit) at the specified `index`. Accepts negative integers, which count back from the last string character.

`String.prototype.charAt(index)`

Features of MDN Reference Pages for Methods

Let's now look at the reference for a specific method. Open the page on `String.prototype.toUpperCase()` by clicking this link (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/toUpperCase), and take a look around. There's a lot of information! This reference page covers:

- **Syntax**: a section that describes the syntax of the method, including how it is called, what arguments the method has (if any), what is returned from the method (if anything), among others.
- **Description**: a section that describes the functionality of the method.
- **Examples**: a section that shows usage examples.
- **Specifications**: a section that links to ECMAscript's official specifications for the method.
- **Browser Compatibility**: a section that lists desktop browsers, mobile browsers, and backend environments (more on this later), and whether

or not the method is supported.
- **See Also**: a section that has links to additional resources (like guides or tutorials) or related topics.

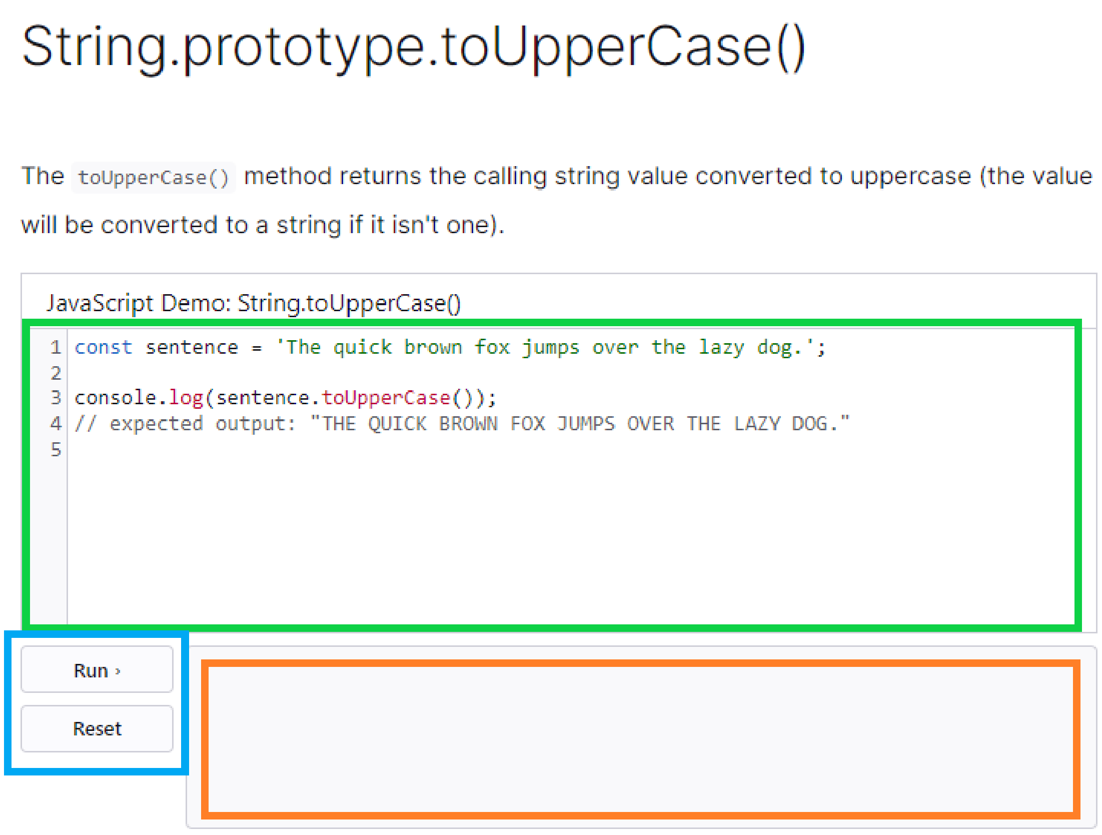
MDN JavaScript Console

You'll also be presented with a **built-in console** at the top of the reference page with some pre-written code. Taken from the `String.prototype.toUpperCase()` reference, the image below shows what this console looks like.

MDN's console is just like the browser DevTools console, and the pre-written code is meant to offer a demo on the method so that you can better understand it. Note that sometimes the demo code in the console will be too complicated to understand. When this happens, scroll down to the "Examples" section of the reference page for more usage examples. You can copy these examples and use them in the console. If the examples are still too challenging to understand or use code that you are not familiar with, ask for help.

Using the image below of the console on the `String.prototype.toUpperCase()` reference page, let's review the options we have to interact with the console:

- **Write code**: In the green box in the image is where we can write code, or adjust the existing demo code. We also don't have to adjust this area — we can simply hit "Run" to execute the demo code. Note — if you want to comment out code, add two forward slashes `//` to the front of the line of code.
- **Execute or reset our code**: The blue box highlights two buttons: "Run" to execute our code or "Reset" to set the console to have its original demo code in it.
- **View results**: The orange box highlights the output area where the results of running our code goes.



String.prototype.toUpperCase()

The `toUpperCase()` method returns the calling string value converted to uppercase (the value will be converted to a string if it isn't one).

JavaScript Demo: String.toUpperCase()

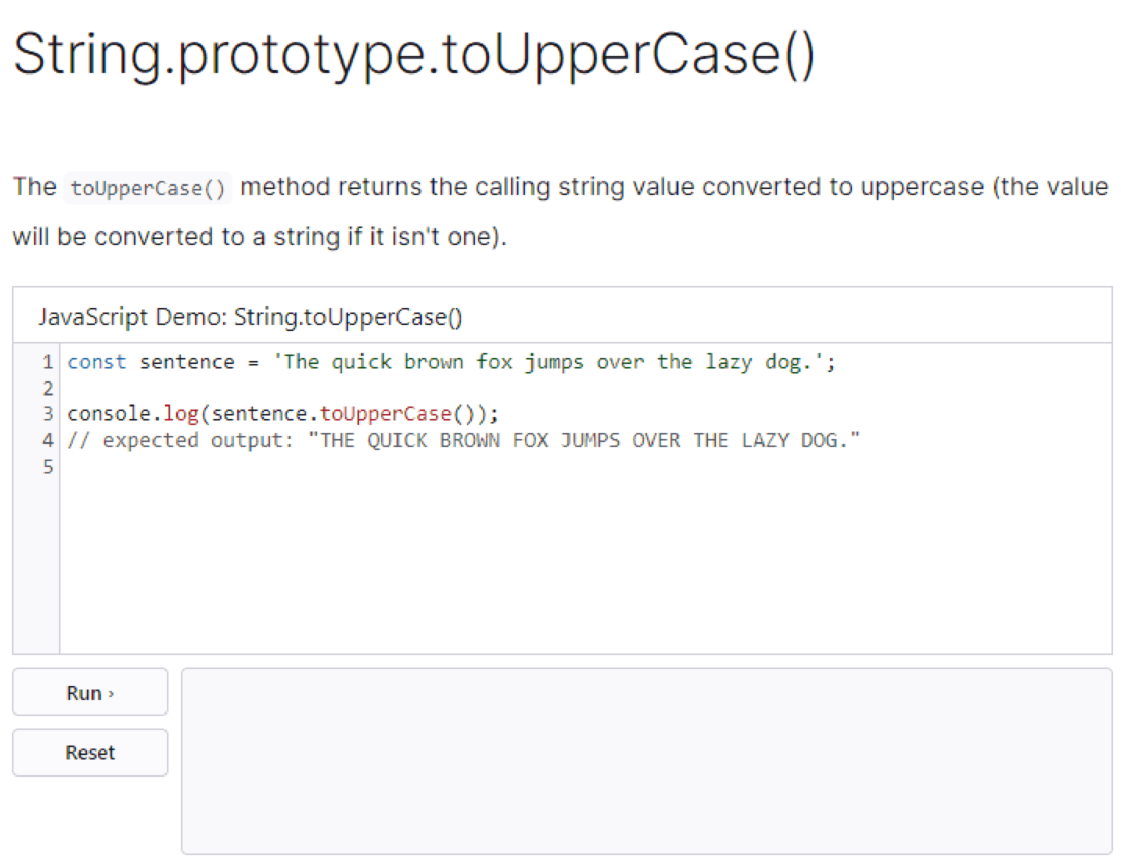
```
const sentence = 'The quick brown fox jumps over the lazy dog.';

console.log(sentence.toUpperCase());
// expected output: "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG."

```

Demonstration of the MDN JS Console

Optionally, watch the gif below that demonstrates how to use the console.



String.prototype.toUpperCase()

The `toUpperCase()` method returns the calling string value converted to uppercase (the value will be converted to a string if it isn't one).

JavaScript Demo: String.toUpperCase()

```
const sentence = 'The quick brown fox jumps over the lazy dog.';

console.log(sentence.toUpperCase());
// expected output: "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG."

```

What is `console.log`?

`console.log` is a method that allows us to log messages to our console — this could be the console built-in to the MDN documentation, or it could be the browser DevTools console. Like in the examples above with `String.prototype.toUpperCase()`, any argument we pass into `console.log()` is logged to the console. This could be a data type or a variable. Try the following code in your browser DevTools console or the MDN console:

```
> console.log("hello world");
// returns "hello world"
> console.log(3);
// returns 3
> const aNumber = 4;
> console.log(aNumber);
// returns 4
```

We'll revisit `console.log()` in future lessons and learn how we can use it to debug our JavaScript code.

Beware of Rabbit Holes

When you are in a rabbit hole, you are jumping from one concept to the next, to the next. This is really easy to do when referencing documentation. We may initially reference the documentation to learn about a specific method. In the process, we see new terminology and go to investigate it. This in turn leads us to more new terminology, concepts, and questions that we investigate. And here we are far down a rabbit hole.

"GOING LOOK SOMETHING UP REAL QUICK ON THE INTERNET....
NOT SURE WHEN I'LL SEE YOU AGAIN."
imgflip.com

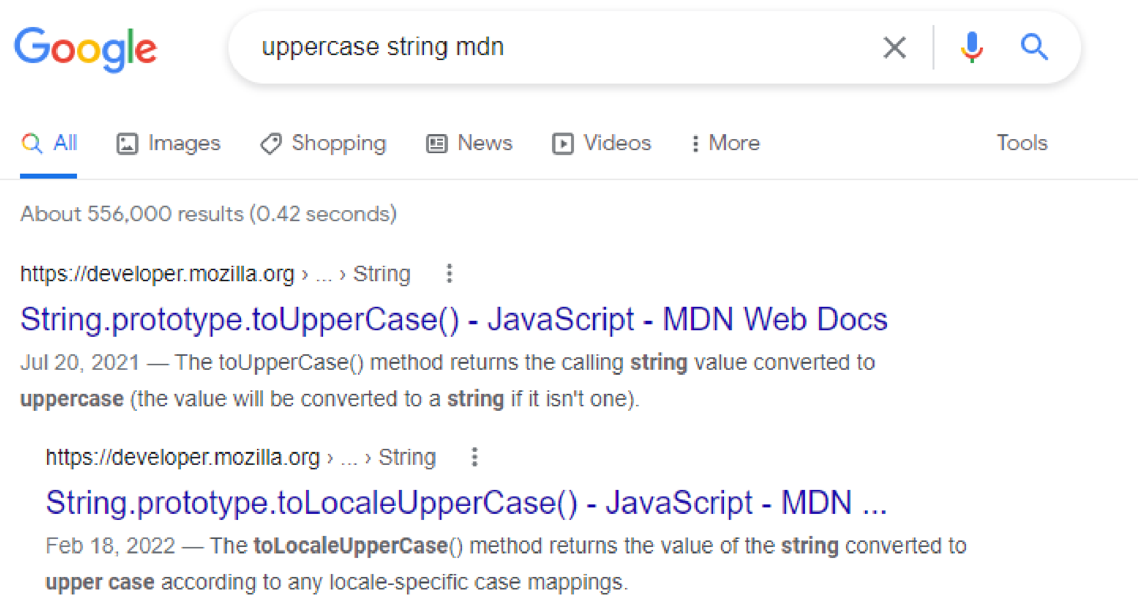
Rabbit holes can be good and bad. The good is learning about new things. The bad is getting overwhelmed and lost, and potentially not understanding what you originally were trying to look up. It's at this point that I want to remind you that **you don't need to know everything about how JavaScript works in order to use JavaScript**.

There's a lot to learn about the inner mechanisms of JavaScript, and you will get there over the course of a few years. Even then, it's highly likely that you will never know everything about JavaScript. That expectation is reserved for developers who have been working on JavaScript for many years, and who work with JavaScript closely as the standards and expectations change year after year. Remember, all programming languages, frameworks and other tooling (like computers, their operating systems and hardware) are changing everyday! The point isn't to know everything, but to know the concepts behind programming so that you can effectively research and learn what you don't know, and adapt to new technology.

So, be patient with yourself. As you learn JavaScript and how to use MDN's documentation, take notes on what you don't understand. Then, talk about it with your dev team, in Scrum, or with your pair. It's totally okay not to understand something the first time you learn about it. It's also normal to forget something that you do understand, like how to write an anchor tag in HTML.

Search Engines Can Help You Access MDN

Doing a search on Google or another search engine can help you quickly access MDN. If you want to target MDN, make sure to include "mdn" in your search query. See the example below, where we've updated our previous example search query: "uppercase string mdn".



About 556,000 results (0.42 seconds)

https://developer.mozilla.org › ... › String
String.prototype.toUpperCase() - JavaScript - MDN Web Docs
Jul 20, 2021 — The toUpperCase() method returns the calling **string** value converted to **uppercase** (the value will be converted to a **string** if it isn't one).

https://developer.mozilla.org › ... › String
String.prototype.toLocaleUpperCase() - JavaScript - MDN ...
Feb 18, 2022 — The **toLocaleUpperCase**() method returns the value of the **string** converted to **upper case** according to any locale-specific case mappings.

Other Links to Specific Documentation

MDN has a lot to offer, so we're going to keep our focus narrow. So far, we've learned about JavaScript data types, numbers and arithmetic, variables, strings, methods, and functions, so we'll take a look at that area of MDN documentation. Here are some direct links to check out:

- **JavaScript data types (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#javascript_types)**
- **Variable declaration with `var`, `let` and `const` (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Grammar_and_types#declarations)**
- **Functions (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/function)**

In other lessons throughout this section, we'll continue our discussion on how to use MDN documentation. Next up, we'll take the time to practice using MDN to find resources on it.

Previous (/introduction-to-programming/javascript-and-web-browsers/practice-string-and-number-methods)
Next (/introduction-to-programming/javascript-and-web-browsers/practice-using-mdn-documentation-javascript)

Lesson 16 of 75
Last updated March 24, 2023

disable dark mode

Epicodus (http://www.epicodus.com)