JavaScript For Cats

An introduction for new programmers

So easy your human owner could do it too!

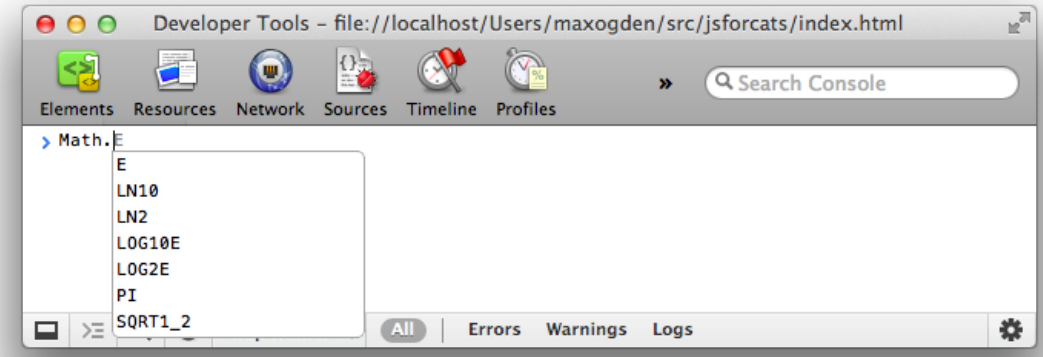
People will tell you JavaScript is complicated, ugly, a toy language, only used by hipsters and many other things. Let's ignore all that stuff forever and instead learn some JavaScript basics so that you can get up and running in no time*!

** actual time: more than none. probably an hour or two. Also since you are a cat you are less likely to run and more likely to lay around in the sun*

The absolute basics

There is JavaScript running on this page right now! Let's play around with it a little. For the sake of simplicity I'll assume you are using Google Chrome to read this page (if you aren't its probably easier on both of us if you follow along with Chrome).

First, right click anywhere on the screen and hit **Inspect Element**, then click on the **Console** tab. You should see a thingy that looks like this:



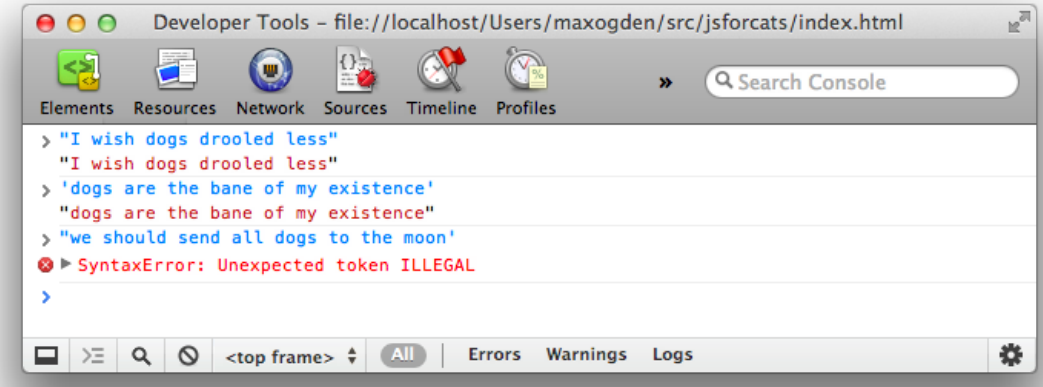
This is a console, otherwise known as a "command line" or "terminal". Basically its a way to type one thing at a time into a computer and immediately get the computers answer back. They are super useful as a learning tool (I still use a the console nearly every day I'm coding).

The console does some pretty cool stuff. Here I have started to type something and the console is helping me out by giving me a list of all the possible things I could continue to type! Another thing you could do is type `1 + 1` into the repl and then hit the `Enter` key and watch what happens.

Using the console (aka REPL) is a very important part of learning JavaScript. If you don't know if something works or what the command is for something, go to the REPL and figure it out! Here's an example:

Strings

Since I am a cat I want to replace every instance of the word `dog` on the internet with `those blasted dogs`. First go into your REPL and type in a few sentences that contain the word `dog` at least once. In JavaScript a bunch of letters, numbers, words or anything else is known as a **String** (as in a *string* of characters). Strings have to begin AND end with a quotation mark. Single `'` or double `"` is fine, just make sure you use the same at the beginning as you do at the end.



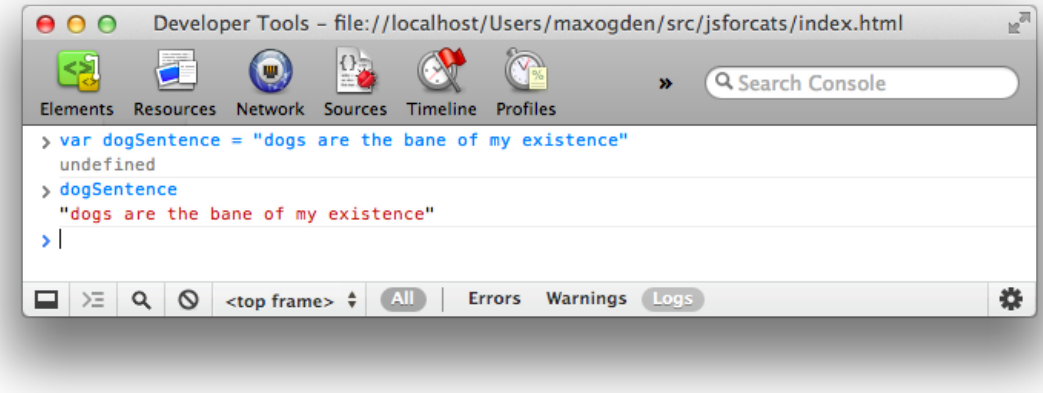
See the nasty error message? Don't worry - you didn't break any laws. SyntaxError ILLEGAL is just the way it sounds when robots tell you that your program has a problem. The first two sentences had matching quotation marks at the beginning and end, but when I mixed single and double

quotation marks it freaked out on me.

Ok, to fix up one of these sentences (by replacing `dog` with our enhanced version) we have to first save the original sentence so that we can call it up later when we do our replacing magic. Notice how the string gets repeated in red when we type it into the console? This is because we haven't told it to save the sentence anywhere so it just gives it right back (or it gives us an Error back if we messed something up).

Variables

To store things we use things called **variables**. The word 'variable' means 'many' and is used because variables can store many different types of things. They are pretty much like mailboxes. We put something in a variable, like our sentence, and then give the variable an address that we can use to look up the sentence later. In real life mailboxes have to have PO Box numbers but in JavaScript you usually just use lowercase letters or numbers without any spaces.



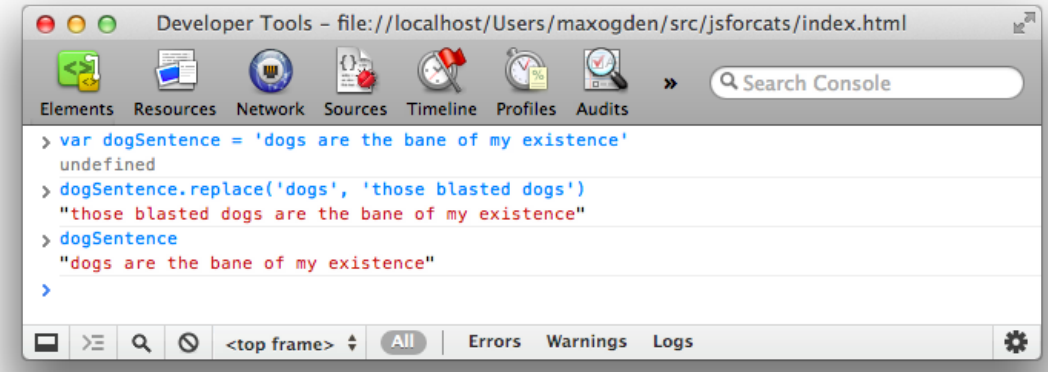
`var` is shorthand for variable and the `=` means *store the thing on the right hand side in the thing on the left hand side*. Also as you can see, now that we are storing our sentence in a variable the console doesn't just return our sentence right away, but instead gives us `undefined` which means *there was nothing to return*.

If you simply type variable names into the console it will return the value stored in those variables. A note about variables is that by default they go away when you switch to a different page. If I were to hit the Refresh button in Chrome, for example, my `dogSentence` variable would get wiped and it would be like it never existed. But don't worry about this too much for now -- you can just hit the up or down arrows on your keyboard while in the console to go through everything you've entered in recently.

Functions

Now that we have our sentence stored in a variable lets give that variable to something that will replace words! We call things that perform actions like this **functions** because they... serve a function (or purpose) for us. Calling them "purposes" sounded weird I guess so they went with the word "function" instead.

JavaScript has a function called `replace` that does exactly what we want! Functions take in any number of variables (zero, one or many) and return either nothing (`undefined`) or exactly one variable (functions can't return two or more variables at a time). The `replace` function is available to use on any strings and takes in two variables: the characters to take out and the characters to swap in. It gets confusing to describe these things so here is a visual example:



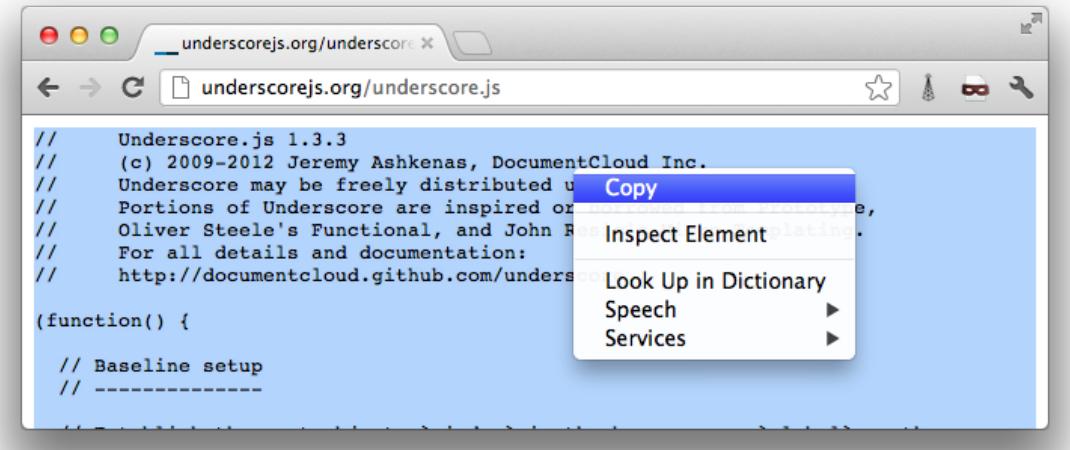
Notice how the value of `dogSentence` is the same even after we run `replace` on it? This is because `replace` (and most JavaScript functions for that matter) take in an existing variable and return a **new variable** instead of modifying the existing variable. Since we didn't store the new variable (there is no `=` on the left side of the replace function) it just printed out in our console.

The "standard library"

You might be wondering what other functions are available in JavaScript. The answer: A TON. There are lots **built in, standard libraries** that you can learn about at MDN (A site run by Mozilla that has lotsa nifty information about web technologies). For example **here is the MDN page on JavaScript's Math object**.

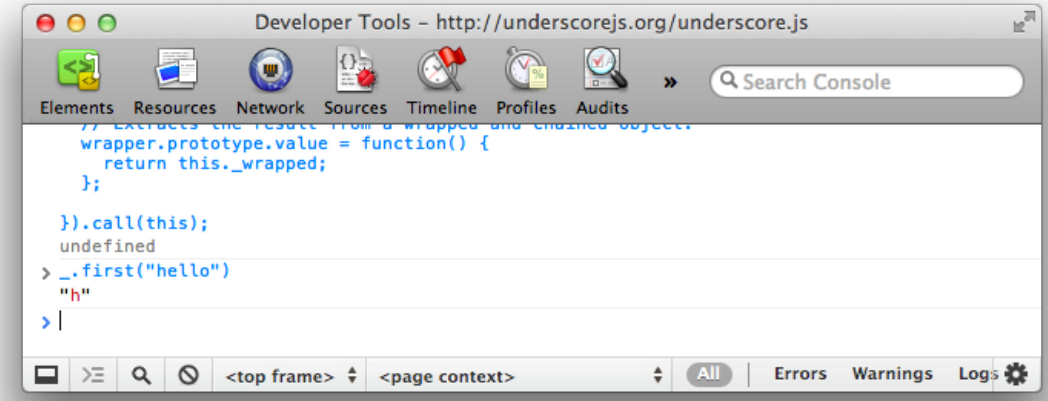
Third-party JavaScript

There is also a lot of JavaScript code available that is **not built in**. JavaScript from third parties is usually referred to as a "library" or "plugin". One of my favorites is called **Underscore.js**. Let's go and grab it and load it into our page! First go to the Underscore site, **http://underscorejs.org/**, click on the download link (I usually use development versions because they are easier to read but both will give you the same basic functionality), and then copy all the code onto your clipboard (you can use Select All from the Edit menu to select everything).



Then paste it into your console and hit enter. Now your browser has a new variable in it: `_`.

Underscore gives you a ton of helpful functions to play with. We'll learn more about how to use them later.

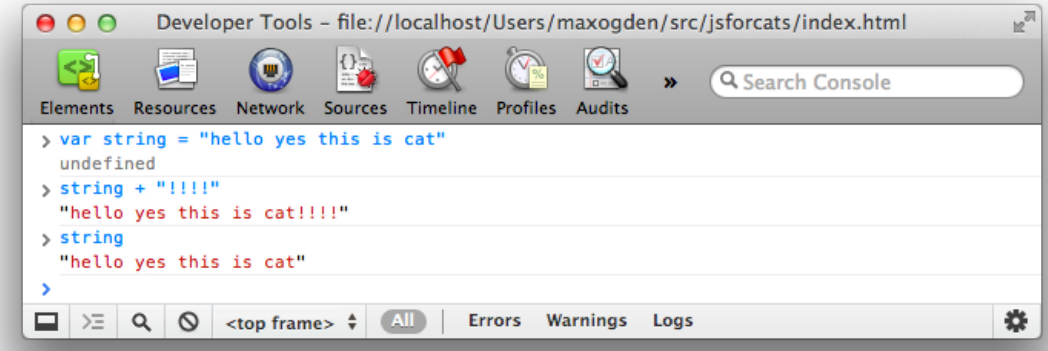


Making new functions

You aren't limited to using other peoples functions -- you can also write them yourself. It's pretty easy! Lets make a function called `makeMoreExciting` that adds a bunch of exclamation points to the end of a string.

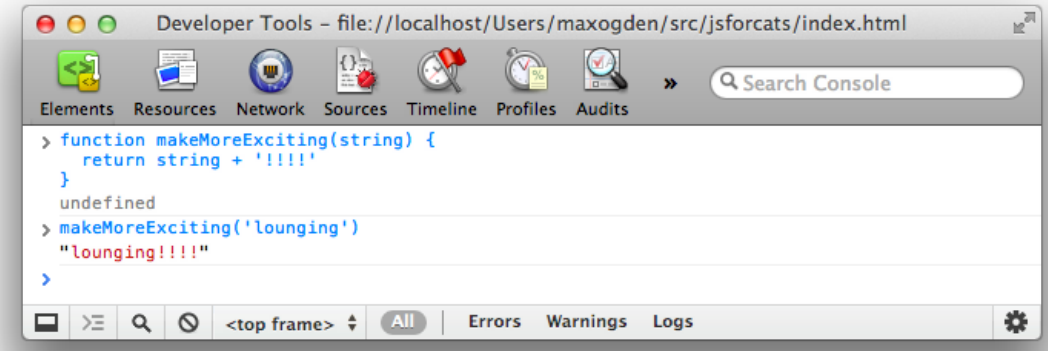
```
function makeMoreExciting(string) {
  return string + '!!!!'
}
```

In my head I read it out loud like this: "theres a function called 'make more exciting' that takes in a string and returns a new copy of that string that has a bunch of exclamation points at the end". Here is how we would write this in the console manually if we weren't using a function:

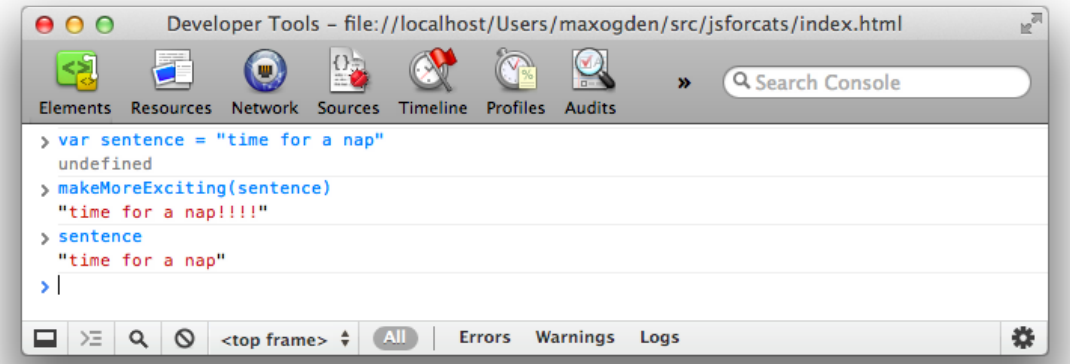


The expression `string + '!!!!!'` returns a new string and our variable called `string` stays the same as before (since we never updated it to anything else with `=`).

Lets use our function instead of doing it manually. First, paste the function into the console and then **call** the function by **passing in** a string:



You could also call the same function by passing in a variable that points to a string (in the above example we just typed the string straight in there instead of saving it to a variable first):

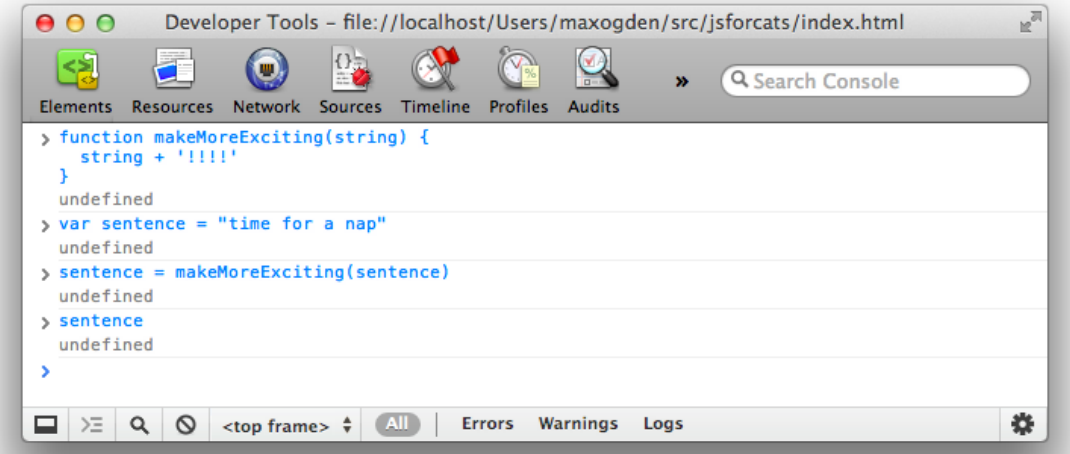


The line `makeMoreExciting(sentence)` is equivalent to saying `sentence + '!!!!!'`. What if we wanted to **modify in-place** (aka update) the value of sentence? Simple save the return variable of the function back into our `sentence` variable:

```
var sentence = "time for a nap"
sentence = makeMoreExciting(sentence)
```

Now `sentence` will have the exclamation marks in it! Note that you only have to use `var` when you are **initializing** a variable -- the first time you ever use it. After that you shouldn't use `var` unless you want to re-initialize (reset/clear/empty) the variable.

What would happen if we took out the `return` statement in our function?

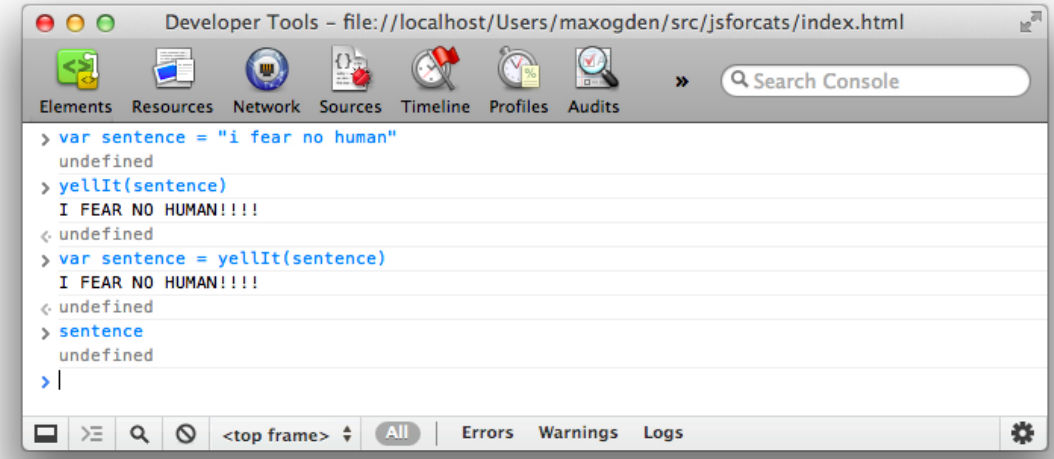


Why is `sentence` empty? Because functions return `undefined` by default! You can choose to return a variable by `return`ing a variable. Functions should take in a variable and, if they change the variable or create a new variable that is supposed to be used later, `return` a variable (fun fact: a fancy term for this style is *functional programming*). Here is another function that doesn't return anything but instead uses a different method to show us the output:

```
function yellIt(string) {
  string = string.toUpperCase()
  string = makeMoreExciting(string)
  console.log(string)
}
```

This function, `yellIt`, uses our previous function `makeMoreExciting` as well as the built-in String method `toUpperCase`. Methods are just a name for a function when it belongs to something -- in this case `toUpperCase` is a function that belongs to `String` so we can refer to it as either a method *or* a function. `makeMoreExciting` on the other hand doesn't belong to anyone so it would be technically incorrect to refer to it as a method (confusing, I know).

The last line of the function is another built-in that simply takes in any variables that you give it and prints them out into the console.



So is there something wrong with the above `yellIt` function? It depends! Here are the two major types of functions:

- functions that modify or create variables and return them
- functions take in variables and perform some action that cannot be returned

`console.log` is an example of the second type of function: it prints things out to your console -- an action that you can see with your eyes but that cannot be represented as a JavaScript variable. My own rule of thumb is to try to keep the two types of functions separate from each other, so here's how I would rewrite the `yellIt` function:

```
function yellIt(string) {
  string = string.toUpperCase()
  return makeMoreExciting(string)
}

console.log(yellIt("i fear no human"))
```

This way `yellIt` becomes more **generic**, meaning it only does one or two simple little things and doesn't know anything about printing itself to a console -- that part can always be programmed later, outside the function definition.

JSForCats.com is a labor of love and work in progress by @maxogden. If you would like to contribute and make this tutorial better there is a Github repo right over here.