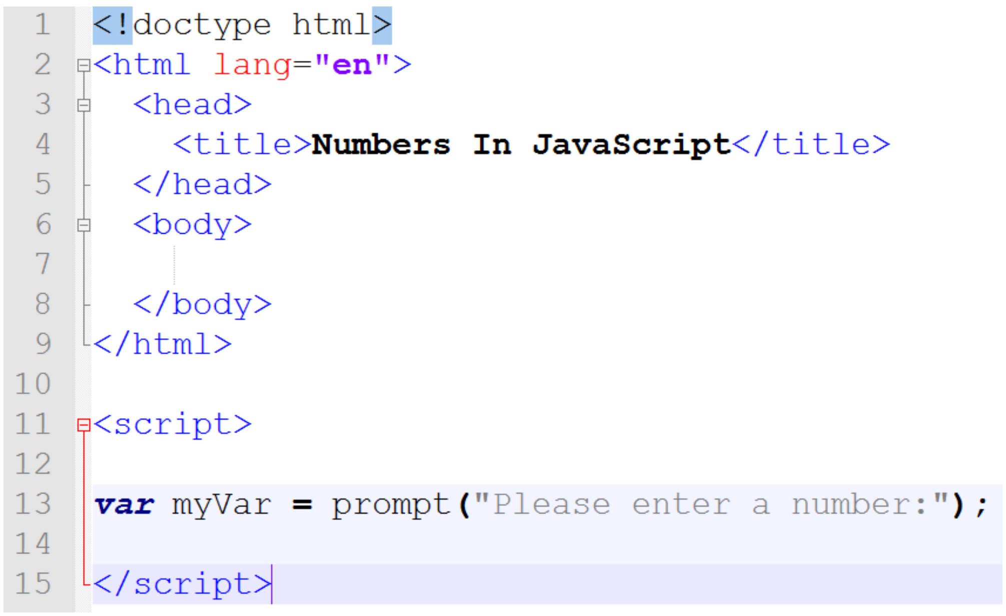
Converting from a string to a number in JavaScript

Following is a brief outline in how to convert strings to numbers, and how we can handle numbers in JavaScript.

Let's assume that we have taken an input from a user using a prompt, outlined with the page shown below:



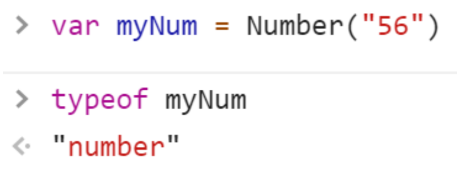
```
<!doctype html>
<html lang="en">
  <head>
    <title>Numbers In JavaScript</title>
  </head>
  <body>

  </body>
</html>

<script>

var myVar = prompt("Please enter a number:");

</script>
```

After this page has run and the user inputs a number, the variable will hold that value, but it is a string. If we want to handle this as a number rather than a string, we can achieve this in a variety of ways.

As we saw in previous videos, we can use the in-built JavaScript Number method to convert a string to a number. This is achieved with the following:

```
var myVar = Number(prompt("Please enter a number:"));
```

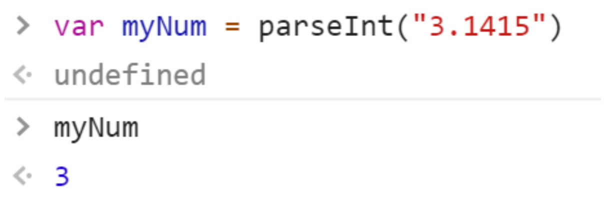
The Number method will convert whatever is inside the brackets to a number type. Consider the code from the console shown below:



```
> var myNum = Number("56")
> typeof myNum
< "number"
```

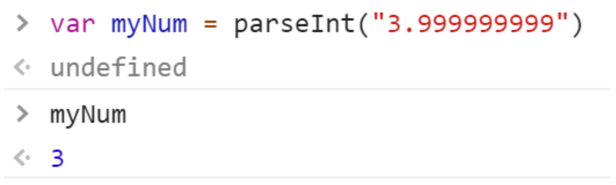
In the above example, a variable is created that is equals to a Number *conversion* of the string 56. (Note: trying to convert a string such as "hello" to a number will result in NaN).

There are other ways of performing this conversion. For example, the parseInt() method. Using parseInt will convert or parse to an integer from a string. It will essentially "chop off" any decimal component – eg., the string "3.1415" would become 3 using parseInt.



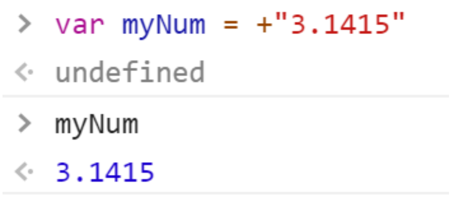
```
> var myNum = parseInt("3.1415")
<· undefined
> myNum
<· 3
```

Note also that parseInt does not round up, so if we had the string "3.999999", it will still convert to the number three. It simply uses what's on the left of the decimal point.



```
> var myNum = parseInt("3.999999999")
<· undefined
> myNum
<· 3
```

Using the unary operator (+) will convert to a number from a string. The unary operator is a plus symbol directly in front of the string you want to convert to a number. So in the same example, the string "3.1415" will become the number 3.1415 using the unary operator.



```
> var myNum = +"3.1415"
<· undefined
> myNum
<· 3.1415
```

Following are some examples of the differences between using parseInt versus unary:

Take for example a variable string that is "123.234abcdef". parseInt can parse this string even though it has characters at the end of it (this will not work if the characters are at the beginning of the string):

```
var a = "123.234abcdef";

var b = parseInt(a);
```

After the above code, the variable b will be equal to the number `123`

If we try the unary operator to convert variable “a” to a number, it will not work, as the unary operator will try to convert the entire string.

```
    var c = +a;
```

After the above code, the variable c will return NaN (this means Not a Number). The unary operator cannot handle the characters at the end of the string (123.234abcdef)

There are also some other possibilities we can use for converting strings to numbers:

parseFloat()

Continuing with the example above, we could do the following:

```
    var d = parseFloat(a);
```

After the above code, the variable d will be equal to the number 123.234

parseFloat() works in a similar manner to parseInt(), but it will include numbers after the decimal point (until it encounters non-numeric characters).

Number()

```
    var e = Number(a);
```

We have seen this method used to convert a string to a number in pervious videos from week 6. This will convert a string to a number (as long as the string does not contain any non-numeric characters). In this regard, it is very similar to the unary operator.

Double tilde

~~

This can also be used to convert a string to a number, and is similar is some respects to parseInt(), in that it will convert a string to an integer.

For example:

```
    var e = ~~”123.456”;
```

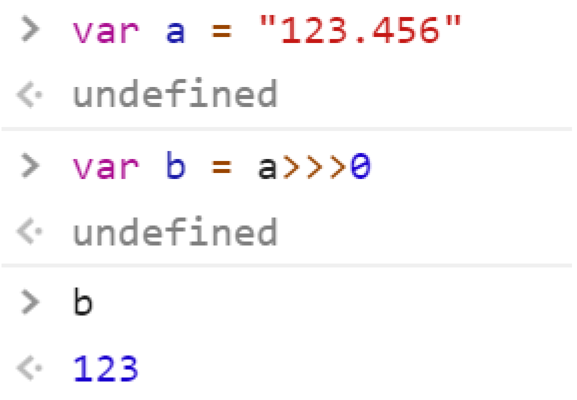
After the above code, the variable e will be equal to 123.
Unlike parseInt(), the double tilde will not work correctly if the string contains non-numeric characters.

Bitwise Operator

>>>0

This will convert to an integer.

For example:

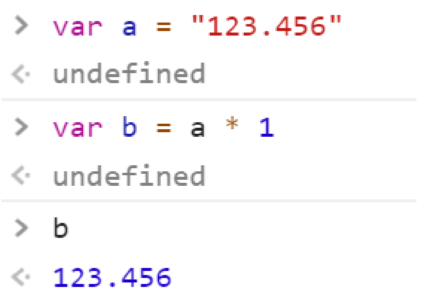


```
> var a = "123.456"
<· undefined
> var b = a>>>0
<· undefined
> b
<· 123
```

Unlike parseInt(), the >>> will not work correctly if the string contains non-numeric characters.

Using Coercion to convert to a number

Although it would not be the best way to achieve this result, and not a recommended way either, it is possible to convert to a number using coercion. By using multiple by 1, this will coerce the string to a number as shown below:



```
> var a = "123.456"
<· undefined
> var b = a * 1
<· undefined
> b
<· 123.456
```

Which should you use?

It really depends on the situation or the result you're trying to achieve. There is a test you can perform to determine which is faster – parseInt or unary. Results are likely to be mixed, and you may see big differences in each method with different browsers. You can perform this test in any browsers you use with this URL:

https://jsperf.com/parseint-vs-unary-operator

Generally, unary is easier to use as it is so simple. However, on the downside it is possibly less readable or more difficult to understand/find when troubleshooting code.