JavaScript

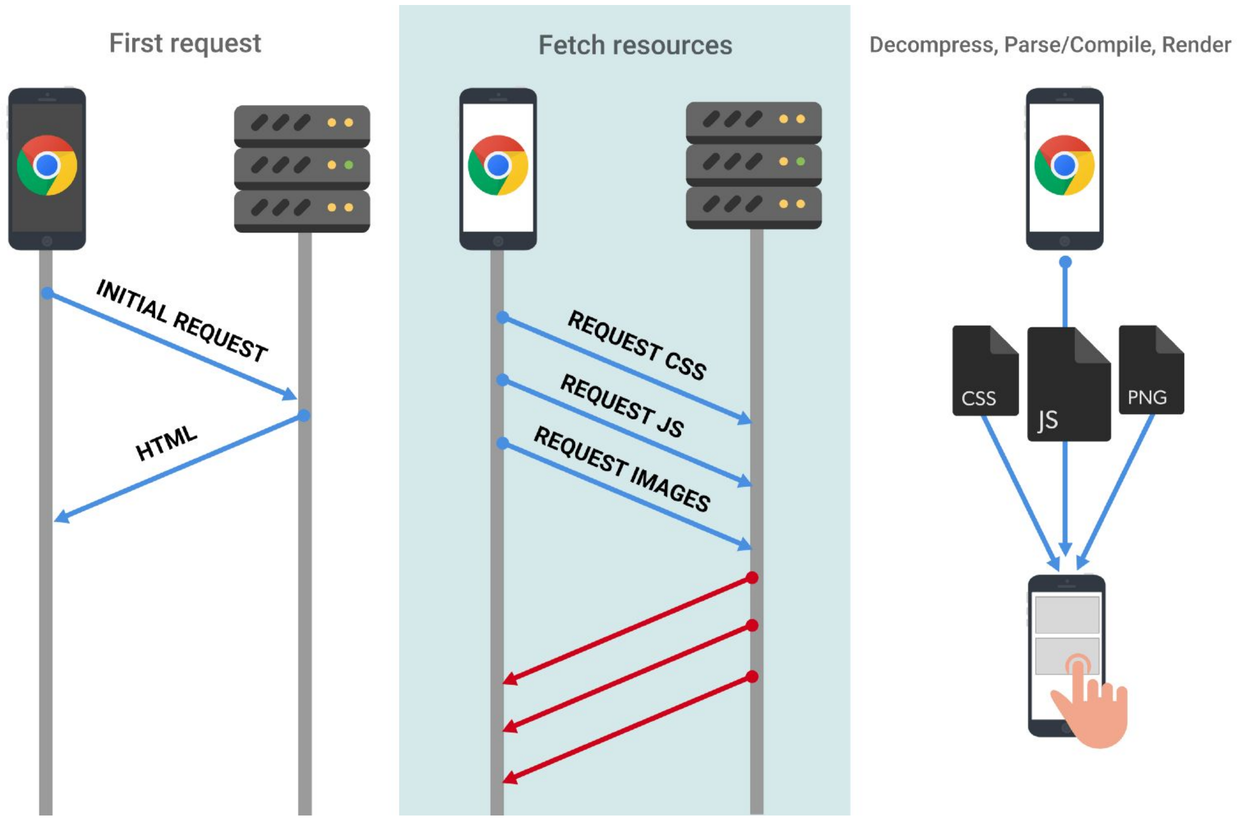
The native language of modern-web browsers

JS

JavaScript (/ˈdʒɑːvəˌskrɪpt/),[8] often abbreviated as JS, is a high-level, interpreted programming language that conforms to the ECMAScript specification. It is a language that is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web.[9] JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it,[10] and all major web browsers have a dedicated JavaScript engine to execute it.

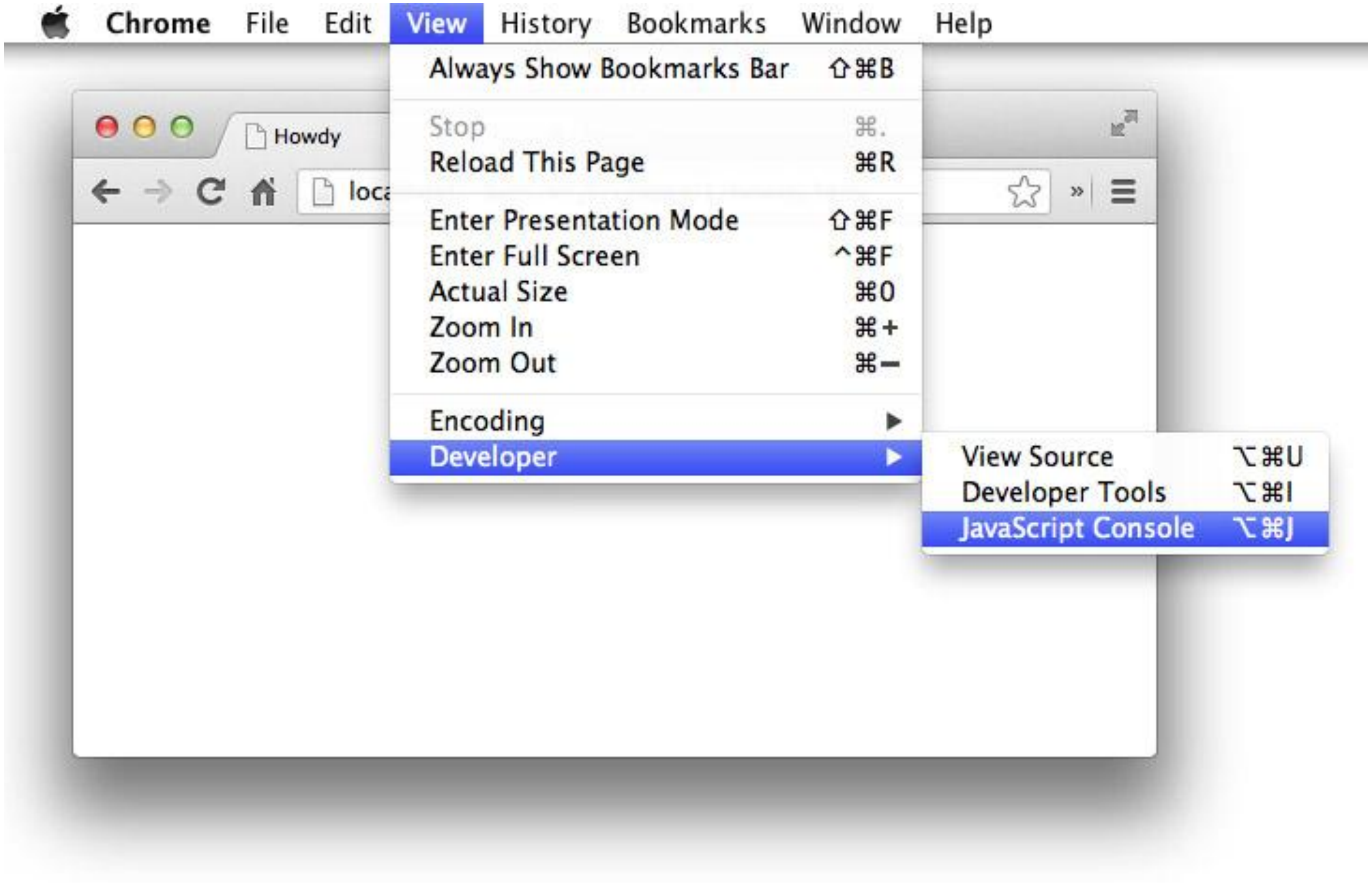
JavaScript is executed by the Browser

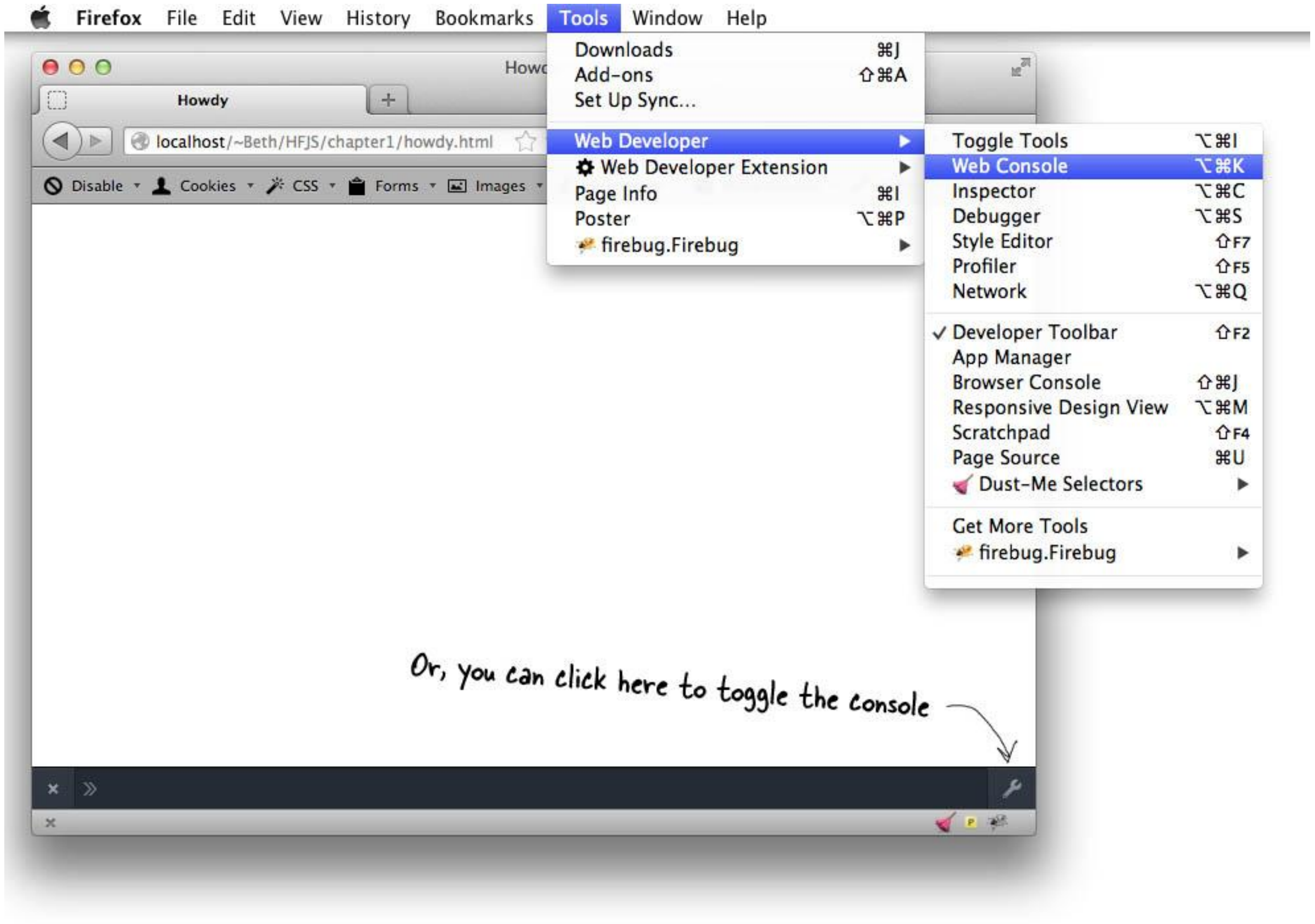


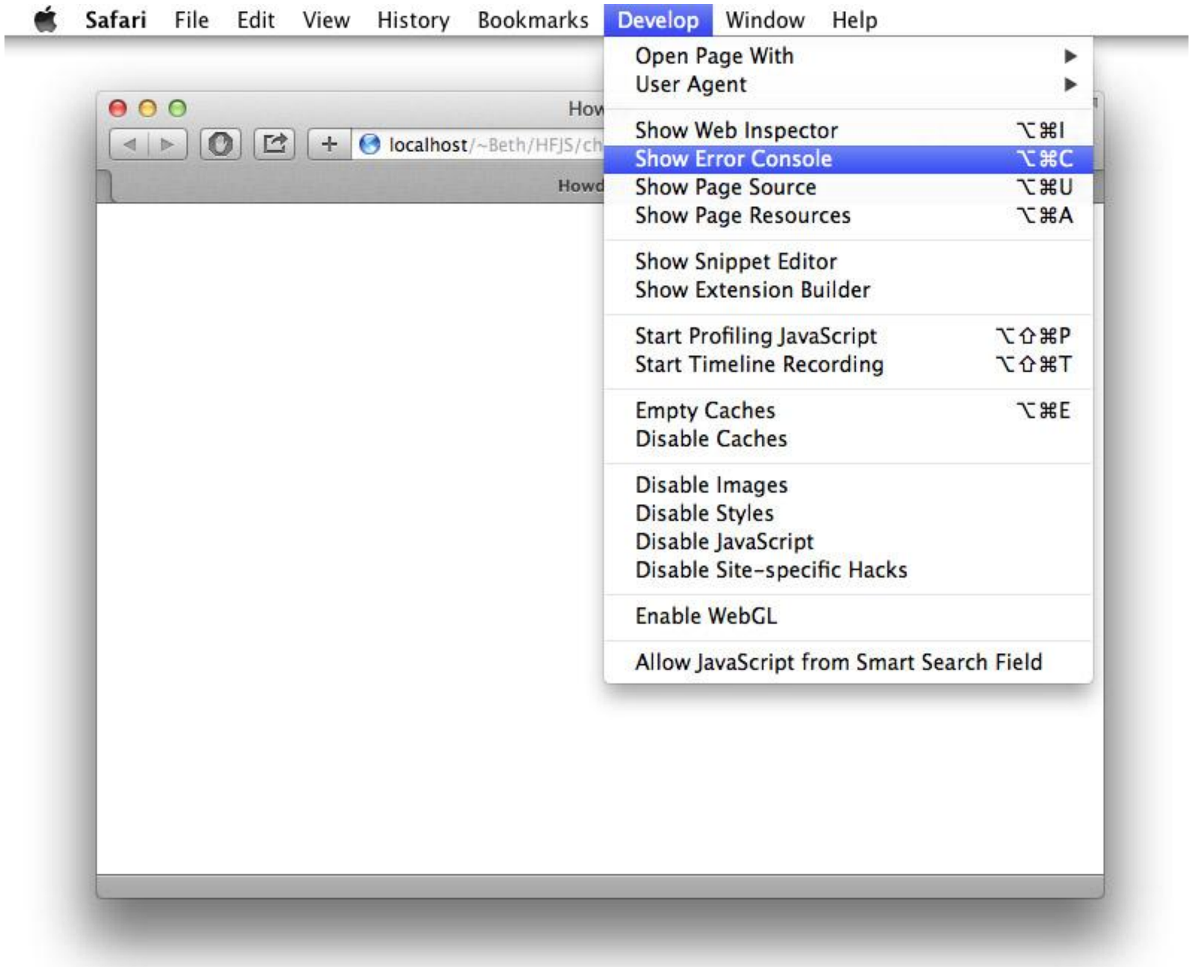
JavaScript Console

The console is a panel that displays important messages, like errors, for developers. Much of the work the computer does with our code is invisible to us by default. If we want to see things appear on our screen, we can print, or log, to our console directly.

It's very useful for us to be able to print values to the console, so we can see the work that we're doing and debug.


Chrome
File
Edit
View
History
Bookmarks
Window
Help
Always Show Bookmarks Bar ⇧⌘B
Stop ⌘.
Reload This Page ⌘R
Enter Presentation Mode ⇧⌘F
Enter Full Screen ^⌘F
Actual Size ⌘0
Zoom In ⌘+
Zoom Out ⌘−
Encoding
Developer
View Source ⌥⌘U
Developer Tools ⌥⌘I
JavaScript Console ⌥⌘J
Howdy
loca


Firefox File Edit View History Bookmarks Tools Window Help
Howdy
localhost/~Beth/HFJS/chapter1/howdy.html
Disable Cookies CSS Forms Images
Downloads ⌘J
Add-ons ⇧⌘A
Set Up Sync...
Web Developer
Web Developer Extension
Page Info ⌘I
Poster ⌥⌘P
firebug.Firebug
Toggle Tools ⌥⌘I
Web Console ⌥⌘K
Inspector ⌥⌘C
Debugger ⌥⌘S
Style Editor ⇧F7
Profiler ⇧F5
Network ⌥⌘Q
✓ Developer Toolbar ⇧F2
App Manager
Browser Console ⇧⌘J
Responsive Design View ⌥⌘M
Scratchpad ⇧F4
Page Source ⌘U
Dust-Me Selectors
Get More Tools
firebug.Firebug
Or, you can click here to toggle the console


Safari
File
Edit
View
History
Bookmarks
Develop
Window
Help
localhost/~Beth/HFJS/ch
Open Page With
User Agent
Show Web Inspector ⌥⌘I
Show Error Console ⌥⌘C
Show Page Source ⌥⌘U
Show Page Resources ⌥⌘A
Show Snippet Editor
Show Extension Builder
Start Profiling JavaScript ⌥⇧⌘P
Start Timeline Recording ⌥⇧⌘T
Empty Caches ⌥⌘E
Disable Caches
Disable Images
Disable Styles
Disable JavaScript
Disable Site-specific Hacks
Enable WebGL
Allow JavaScript from Smart Search Field

Printing to the console

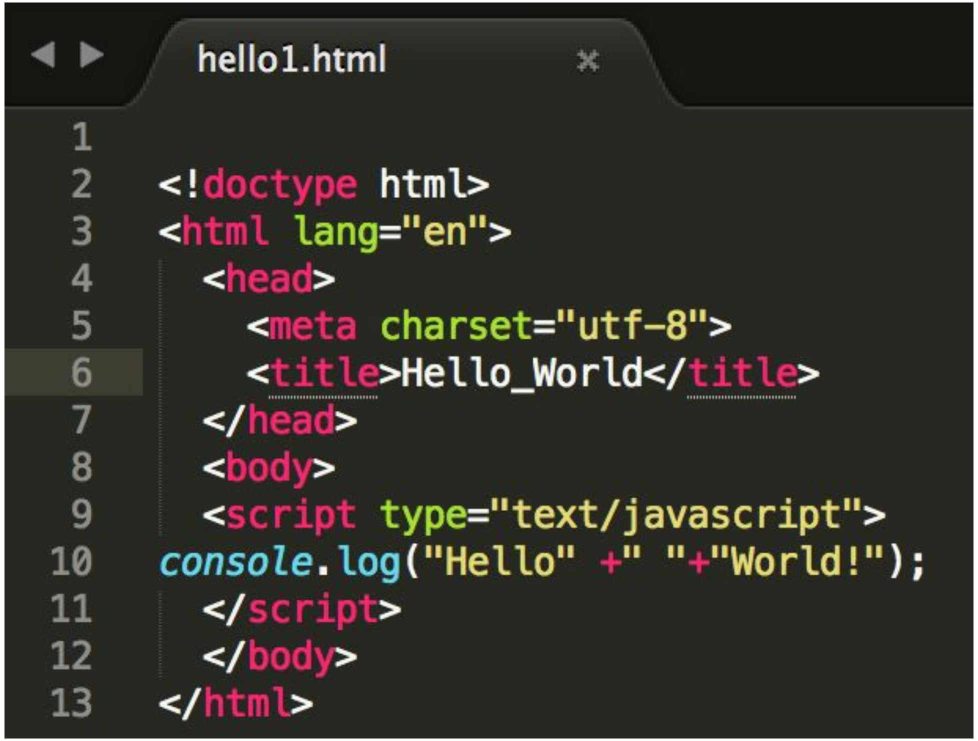
The Console method log() outputs a message to the web console. The message may be a single string (with optional substitution values), or it may be any one or more JavaScript objects.

Syntax:

```
console.log(obj1 [, obj2, ..., objN]);
console.log(msg [, subst1, ..., substN]);
```

https://developer.mozilla.org/en-US/docs/Web/API/Console/log

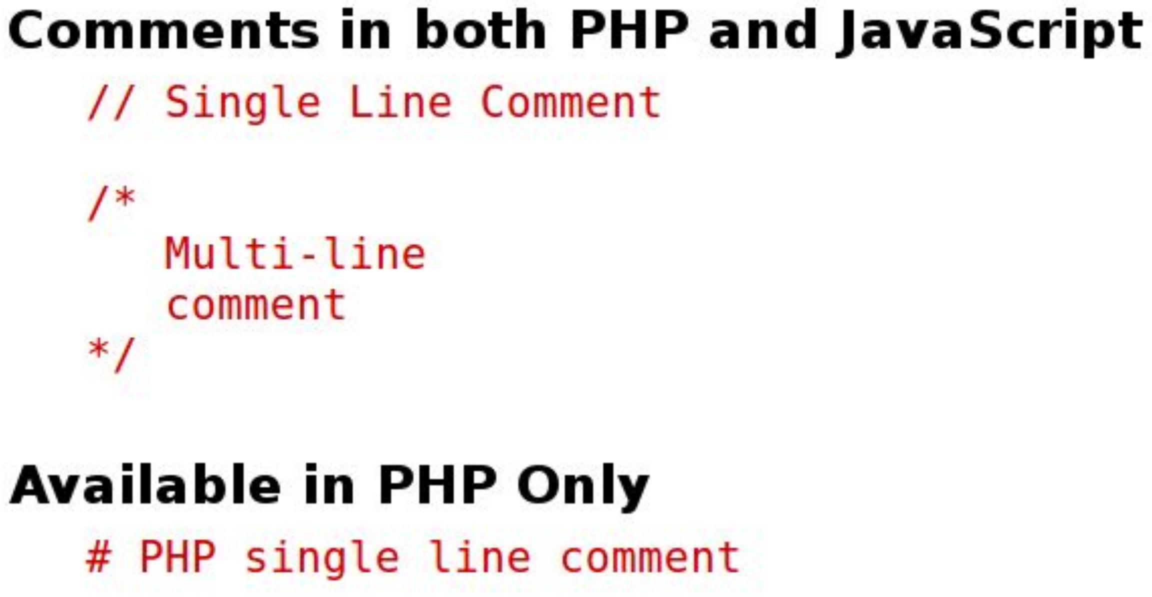
Let's write the following code and preview it in Chrome:


hello1.html

```

<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello_World</title>
  </head>
  <body>
  <script type="text/javascript">
console.log("Hello" +" "+"World!");
  </script>
  </body>
</html>
```

*Comments

In JavaScript (like other C-based languages) comments are designated using the double backslash:



Comments in both PHP and JavaScript

```
// Single Line Comment

/*
   Multi-line
   comment
*/
```

Available in PHP Only

```
# PHP single line comment
```

Data Types

JS uses 7 fundamental data types

1. *Number*: Any number, including numbers with decimals: `4`, `8`, `1516`, `23.42`.
2. *String*: Any grouping of characters on your keyboard (letters, numbers, spaces, symbols, etc.) surrounded by single quotes: `' ... '` or double quotes `" ... "`. Though we prefer single quotes. Some people like to think of string as a fancy word for text.
3. *Boolean*: This data type only has two possible values— either `true` or `false`(without quotes). It's helpful to think of booleans as on and off switches or as the answers to a "yes" or "no" question.
4. *Null*: This data type represents the intentional absence of a value, and is represented by the keyword `null` (without quotes).
5. *Undefined*: This data type is denoted by the keyword `undefined` (without quotes). It also represents the absence of a value though it has a different use than `null`.
6. *Symbol*: A newer feature to the language, symbols are unique identifiers, useful in more complex coding. No need to worry about these for now.
7. *Object*: Collections of related data.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures

Data Primitives

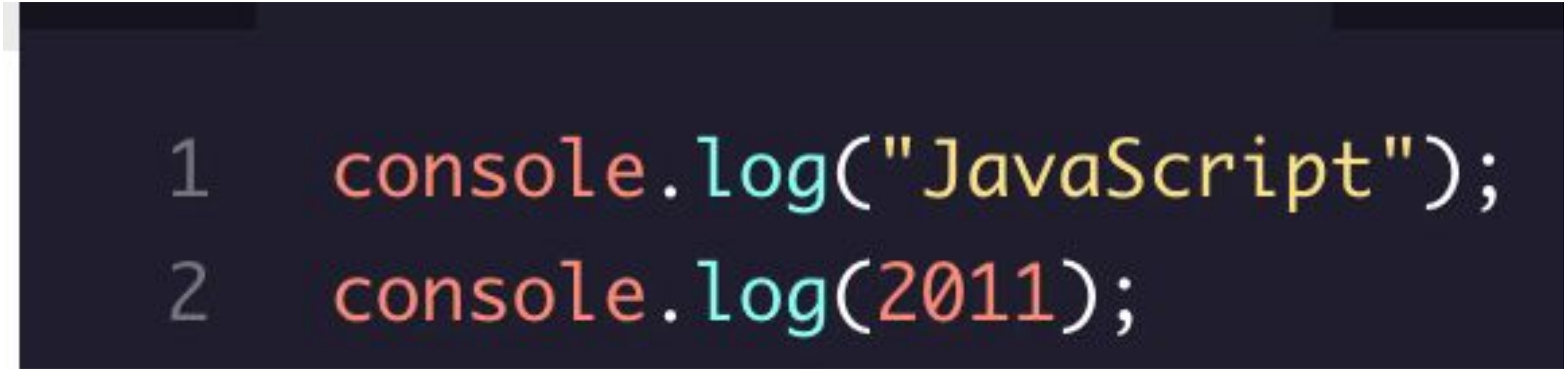
The first 6 of those types are considered primitive data types. They are the most basic data types in the language. Objects are more complicated and we'll look at them as we progress.

Two of the most common data types are Strings and Numbers.

Strings are expressed in quotes and have no numerical value.

Numbers are expressed without quotes and hold a mathematical value.

Try inputting the following into your code



```
console.log("JavaScript");
console.log(2011);
```

The first statement outputs a string. The second, a number.

Class Exercise - Strings & Numbers

Modify your previous hello_world code. Save as a new file and add the following:

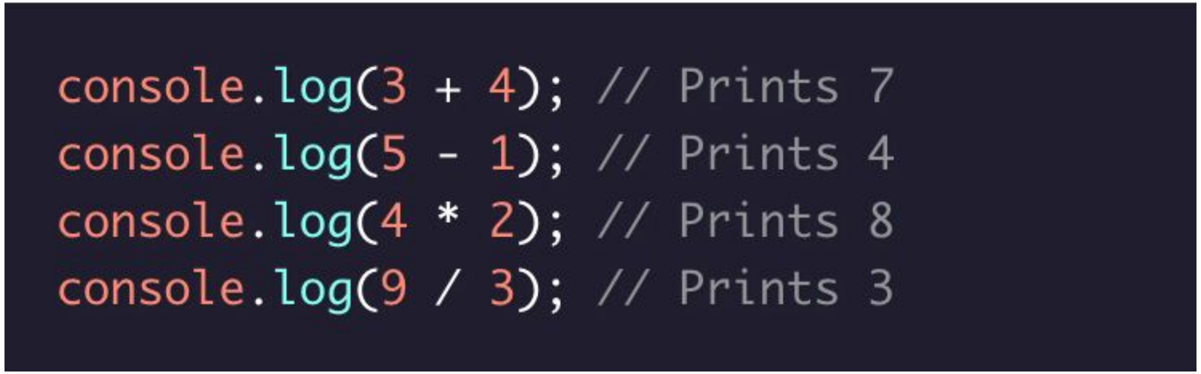
1) Output a string to the console
2) Output an integer to the console
3) Output the string: `Woohoo! I love to code! #javascript` to the console.
4) Output a float value to the console

Arithmetic Operators

Operator Statements

An operator is a character that performs a task in our code. JavaScript has several built-in in arithmetic operators, that allow us to perform mathematical calculations on numbers. These include the following operators and their corresponding symbols:

1. Add: +
2. Subtract: -
3. Multiply: *
4. Divide: /
5. Remainder: %



```
console.log(3 + 4); // Prints 7
console.log(5 - 1); // Prints 4
console.log(4 * 2); // Prints 8
console.log(9 / 3); // Prints 3
```

Your Turn

To your code, add the following console statements:

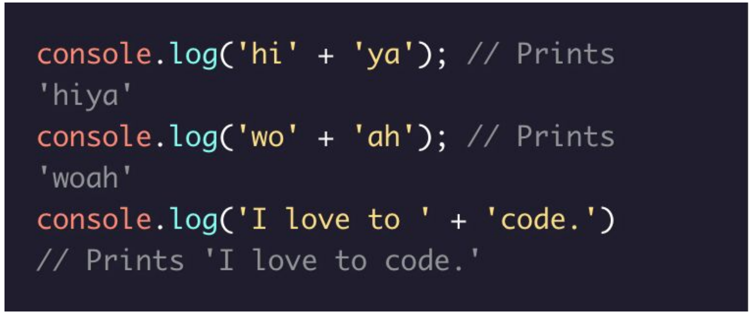
1. Inside of a `console.log()`, add `3.5` to your age. This is the age you'll be when we start sending people to live on Mars.
2. On a new line write another `console.log()`. Inside the parentheses, take the current year and subtract `1969`.

   The answer is how many years it's been since the 1969 moon landing.
3. Create another `console.log()`. Inside the parentheses divide `65` by `240`.
4. Create one last `console.log()`. Inside the parentheses, multiply `0.2708` by `100`.

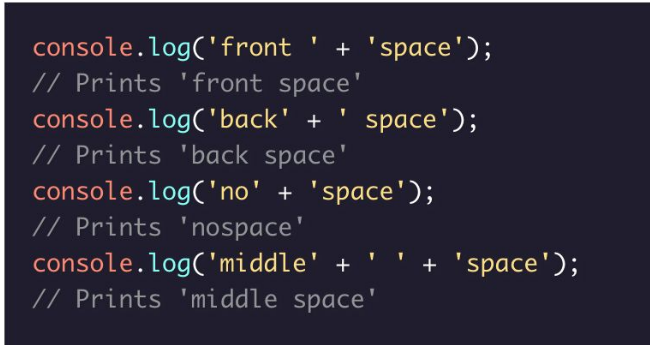
   That's the percent of the sun that is made up of helium. Assuming we could stand on the sun, we'd all sound like chipmunks!

String Concatenation

Operators aren't just for numbers! When a + operator is used on two strings, it appends the right string to the left string:



```
console.log('hi' + 'ya'); // Prints
'hiya'
console.log('wo' + 'ah'); // Prints
'woah'
console.log('I love to ' + 'code.')
// Prints 'I love to code.'
```

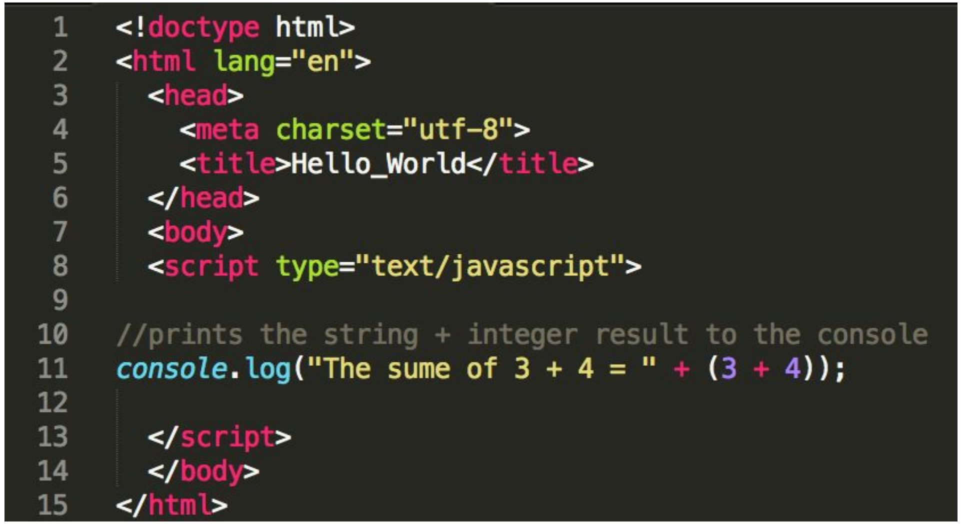


```
console.log('front ' + 'space');
// Prints 'front space'
console.log('back' + ' space');
// Prints 'back space'
console.log('no' + 'space');
// Prints 'nospace'
console.log('middle' + ' ' + 'space');
// Prints 'middle space'
```

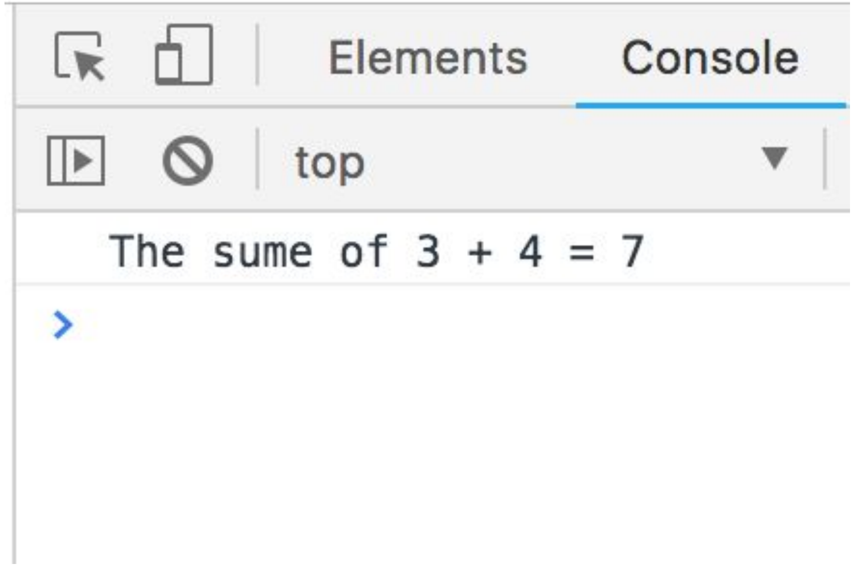
This process of appending one string to another is called concatenation. Notice in the third example we had to make sure to include a space at the end of the first string. The computer will join the strings exactly, so we needed to make sure to include the space we wanted between the two strings.

Mixing Numbers + Strings

We can concatenate output by paying attention to expressing strings + numbers correctly:



```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Hello_World</title>
  </head>
  <body>
  <script type="text/javascript">

//prints the string + integer result to the console
console.log("The sume of 3 + 4 = " + (3 + 4));

  </script>
  </body>
</html>
```



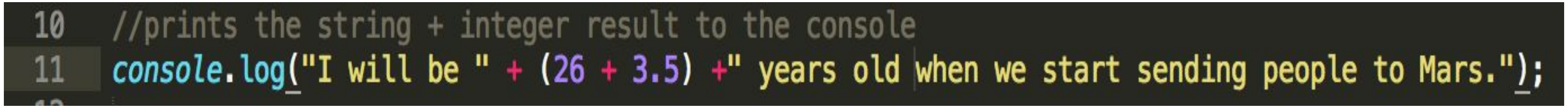
```
The sume of 3 + 4 = 7
>
```

Class Exercise

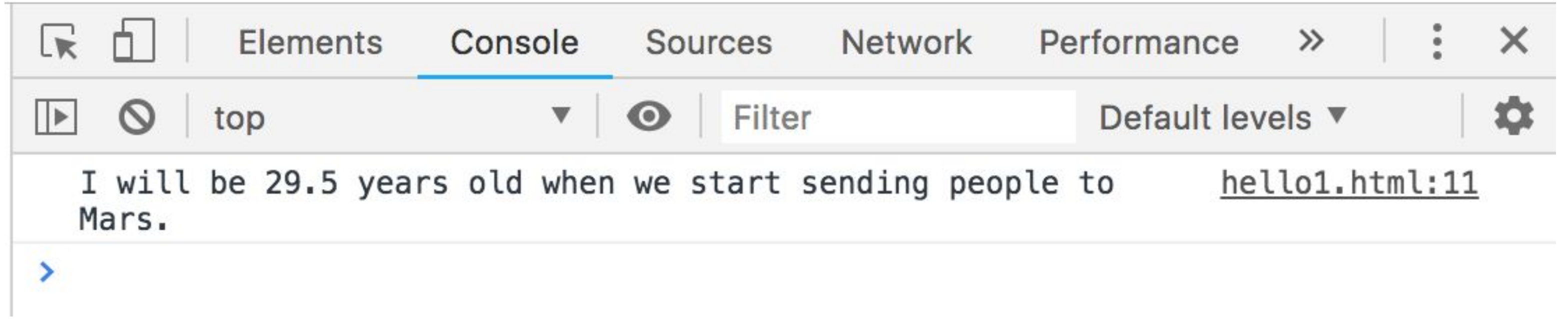
Rewrite your arithmetic operators code to include the textual context.

For example, the console output to question #1 should read:

I will be 29.5 years old when we start sending people to Mars.



```
//prints the string + integer result to the console
console.log("I will be " + (26 + 3.5) +" years old when we start sending people to Mars.");
```



```
I will be 29.5 years old when we start sending people to     hello1.html:11
Mars.
```

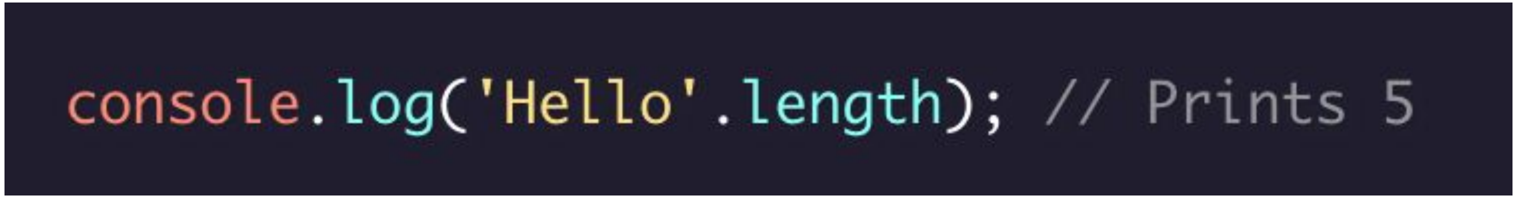
Properties

The “.” operator

When you introduce a new piece of data into a JavaScript program, the browser saves it as an instance of the data type.

Every data type has certain properties associated with it.

Every string instance has a property called length that stores the number of characters in that string. You can retrieve property information by appending the string with a period and the property name:



```
console.log('Hello'.length); // Prints 5
```

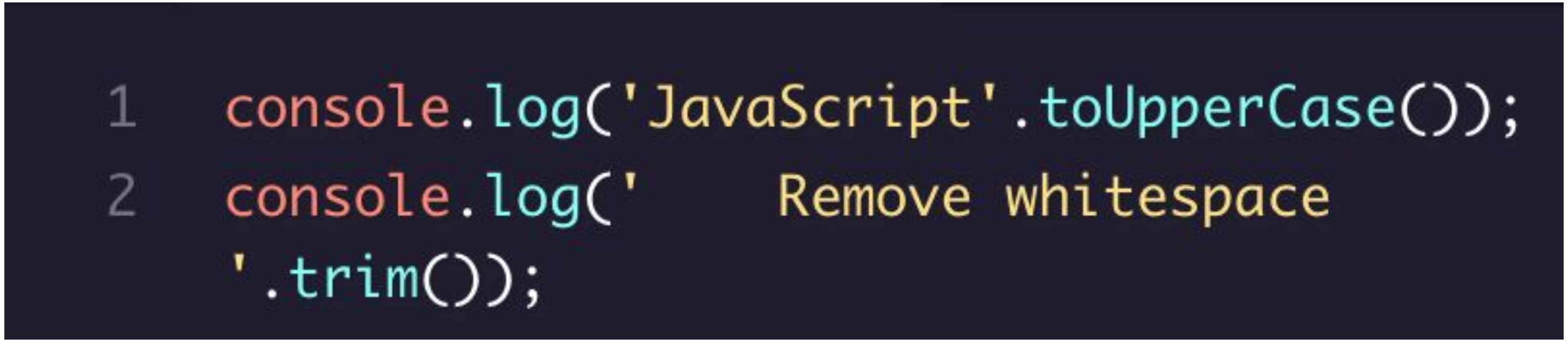
Methods

Remember that methods are actions we can perform. JavaScript provides a number of string methods.

We call, or use, these methods by appending an instance with a period (the dot operator), the name of the method, and opening and closing parentheses: ie. 'example string'.methodName().

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/prototype

Add the following to your code



```
console.log('JavaScript'.toUpperCase());
console.log('    Remove whitespace
'.trim());
```

Built-in objects

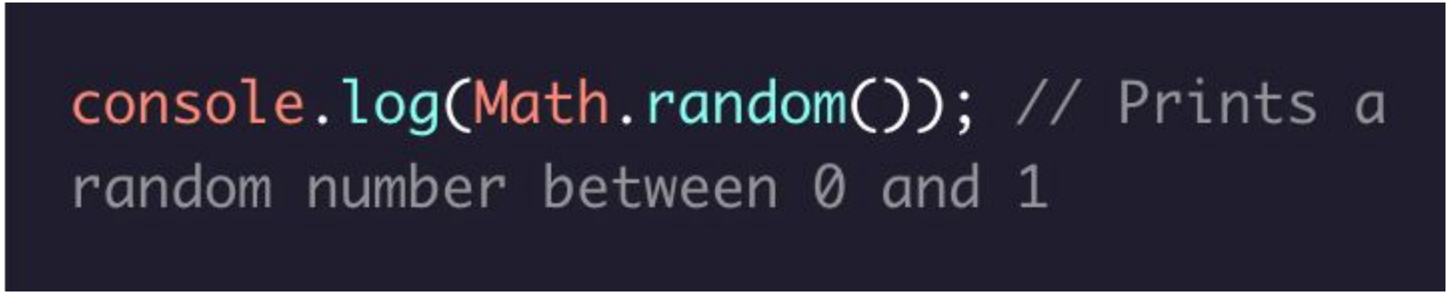
In addition to console, there are other objects built into JavaScript. Down the line, you’ll build your own objects, but for now these “built-in" objects are full of useful functionality.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

Practice with global objects

For example, if you wanted to perform more complex mathematical operations than arithmetic, JavaScript has the built-in Mathobject.

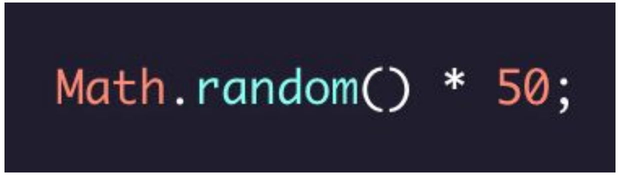
The great thing about objects is that they have methods! Let's call the .random() method from the built-in Math object:



```
console.log(Math.random()); // Prints a
random number between 0 and 1
```

Arithmetic operators with objects

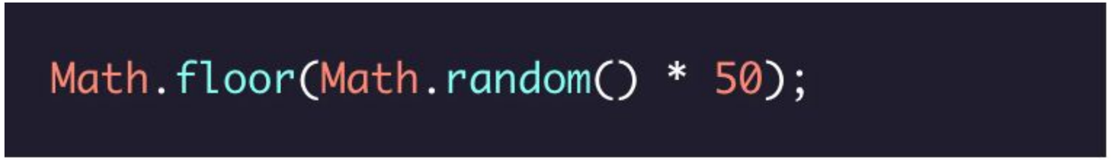
To generate a random number between 0 and 50, we could multiply this result by 50, like so:



```
Math.random() * 50;
```

The example above will likely evaluate to a decimal. To ensure the answer is a whole number, we can take advantage of another useful `Math` method called `Math.floor()`.

`Math.floor()` takes a decimal number, and rounds down to the nearest whole number. You can use `Math.floor()` to round down a random number like this:

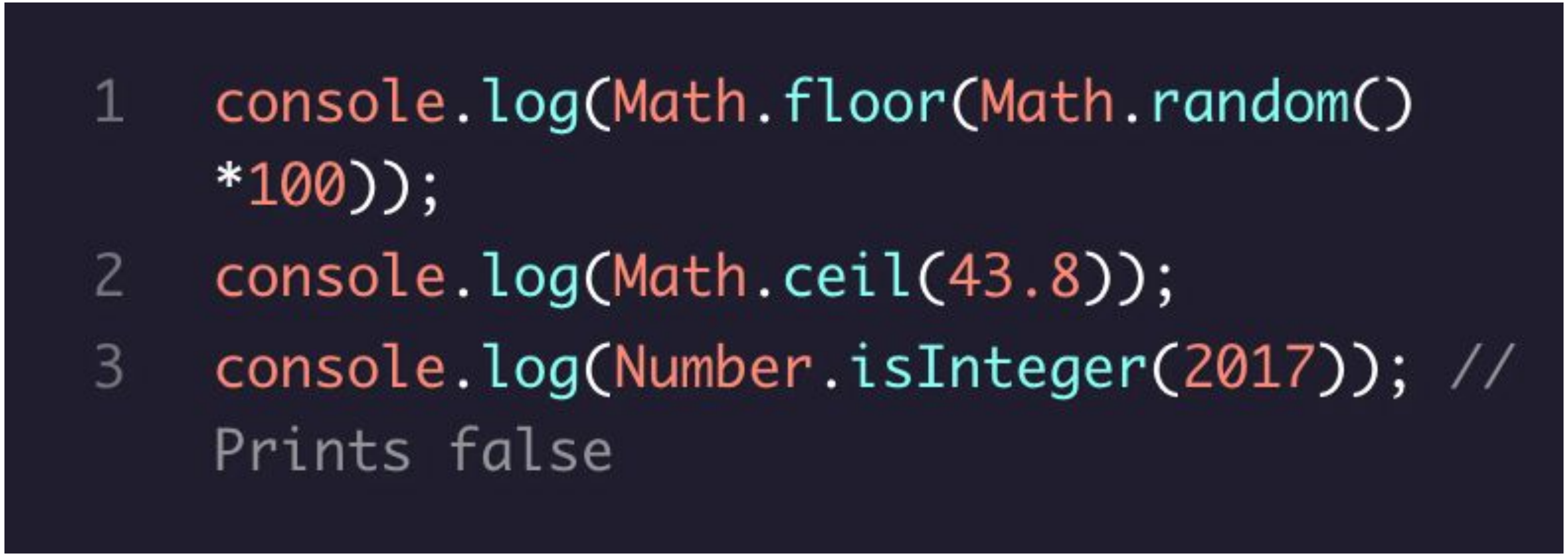


```
Math.floor(Math.random() * 50);
```

Your turn

1) Inside of a `console.log()`, create a random number with `Math.random()`, then multiply it by `100`.
2) Now, use `Math.floor()` to make the output a whole number. Inside the `console.log` you wrote in the last step, put `Math.random() * 100` inside the parentheses of `Math.floor()`.
3) Find a method on the JavaScript `Math`object that returns the smallest integer greater than or equal to a decimal number. Use this method with the number `you generated`. Log the answer to the console.
4) Use the JavaScript documentation to find a method on the built-in `Number` object that checks if a number is an integer.

   Put the number `you generated from #3` in the parentheses of the method and use `console.log()` to print the result.



```javascript
console.log(Math.floor(Math.random()*100));
console.log(Math.ceil(43.8));
console.log(Number.isInteger(2017)); // Prints false
```

Output

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

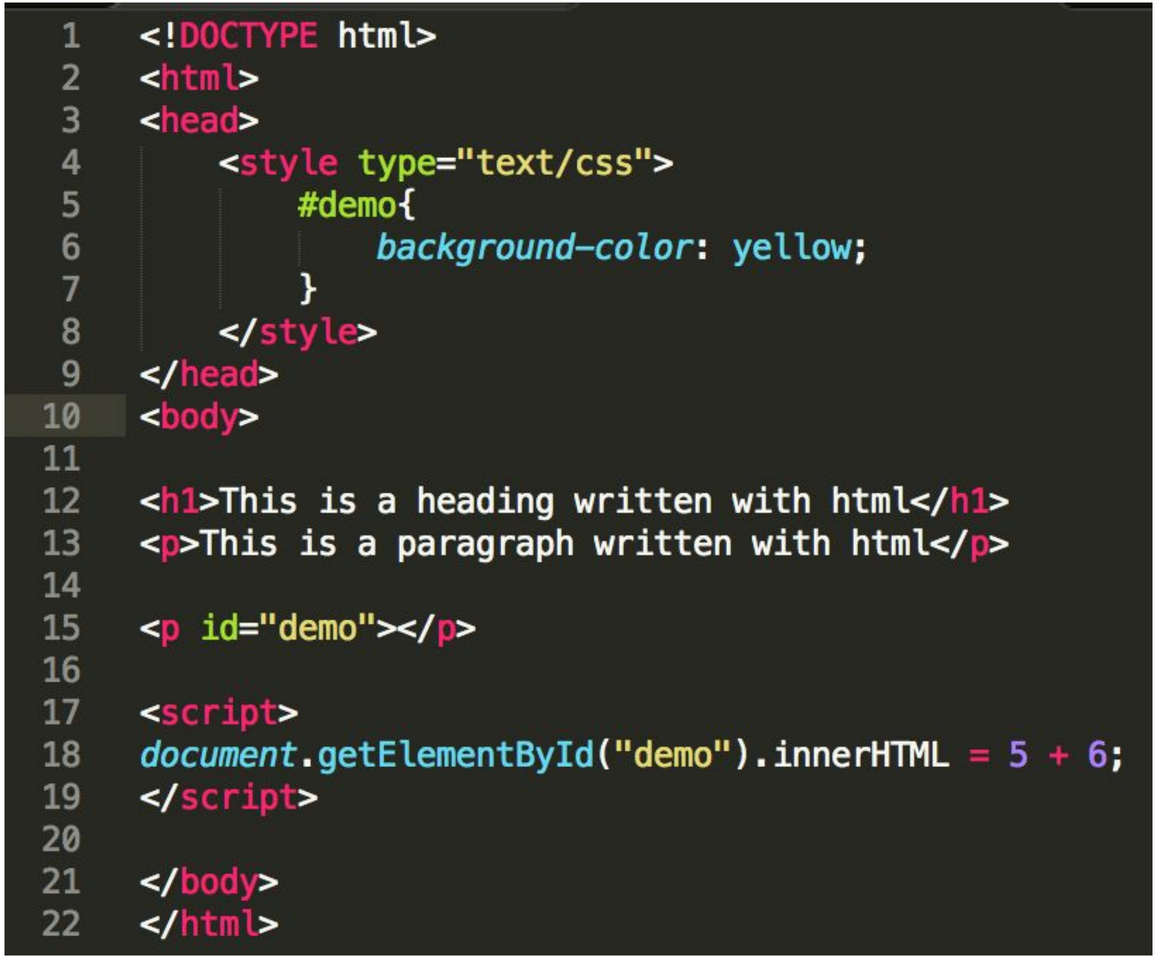
Using innerHTML

To access an HTML element, JavaScript can use the document.getElementById(id) method:

https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById

The id attribute defines the HTML element. The innerHTML property defines the HTML content:

```
document.getElementById("id_name").innerHTML = “your output”;
```



```
<!DOCTYPE html>
<html>
<head>
    <style type="text/css">
        #demo{
            background-color: yellow;
        }
    </style>
</head>
<body>

<h1>This is a heading written with html</h1>
<p>This is a paragraph written with html</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

Try this code on your own.

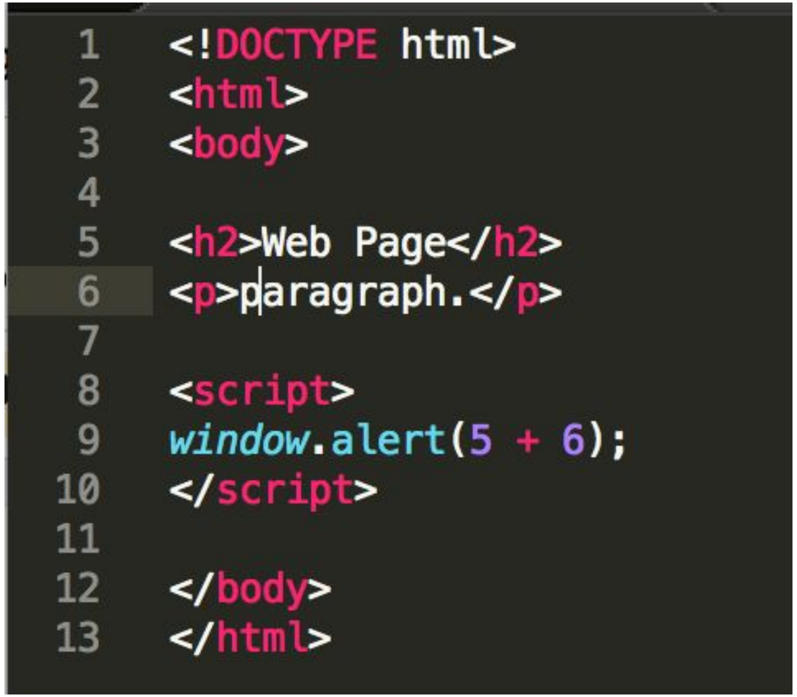
Then, create another div, assign it an id, and add your own string message to display within that div using the innerHTML method with JavaScipt.

Try it with an image:

.innerHTML="<img src=' ' >" ;

window.alert()

You can use an alert box to display data:



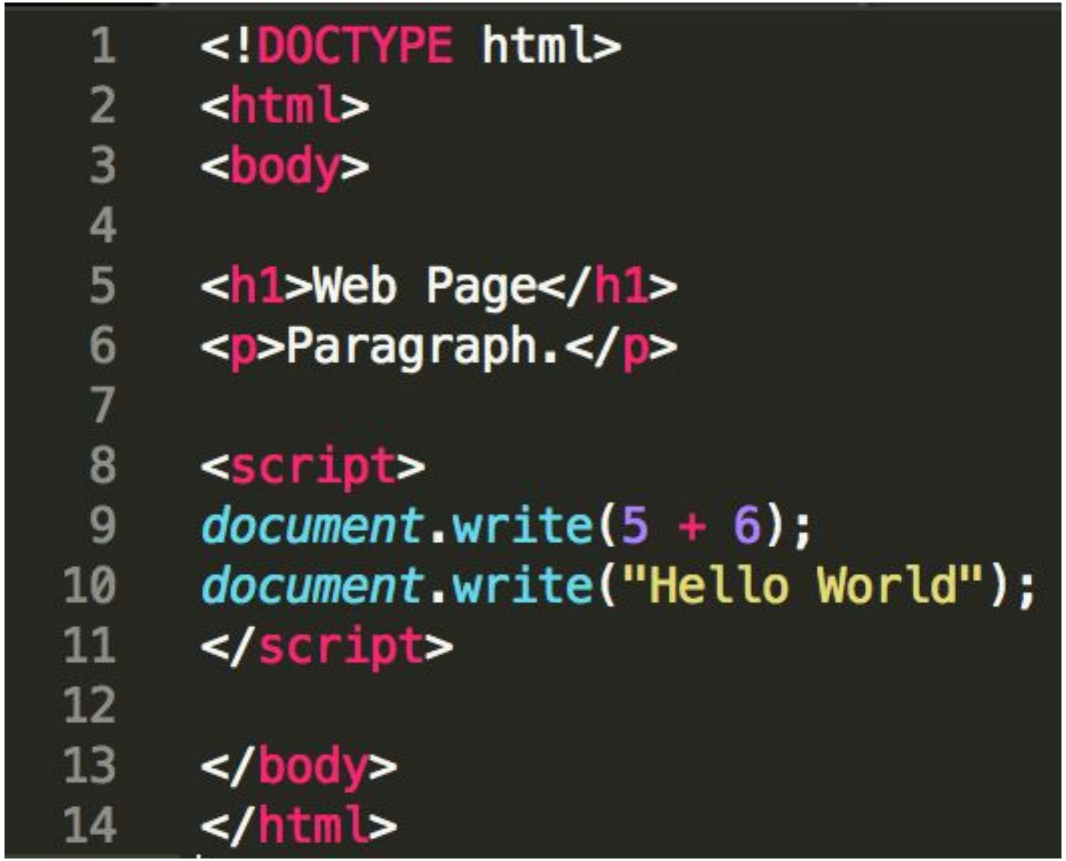
```
<!DOCTYPE html>
<html>
<body>

<h2>Web Page</h2>
<p>paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

Try this code and then add a second window alert displaying a string message.

document.write()

The document.write() method writes a string of text to a document stream. For testing purposes, it is convenient to use document.write() but please keep in mind this function deletes existing html after the document is loaded.

We will practice a bit with it, but it’s real utility beyond testing & learning lies with **AJAX (Asynchronous Javascript and XML)** programming. A technique for dynamic web programming we will look at later in class.



```
<!DOCTYPE html>
<html>
<body>

<h1>Web Page</h1>
<p>Paragraph.</p>

<script>
document.write(5 + 6);
document.write("Hello World");
</script>

</body>
</html>
```

Test this code out.

Use document.write to output the results of your own arithmetic operation and then a string message.

LAST CLASS 4 EXERCISE

1) Take your console output from the exercises on slide 15 and use the inner.HTML method to output each answer into it's own div
2) Do the same with your answers from slide 21
3) And the same with your answers from slide 29
4) Use CSS to position your divs as a horizontal and vertical grid and give each background a different color:

   https://gridbyexample.com/examples/