Website Design I

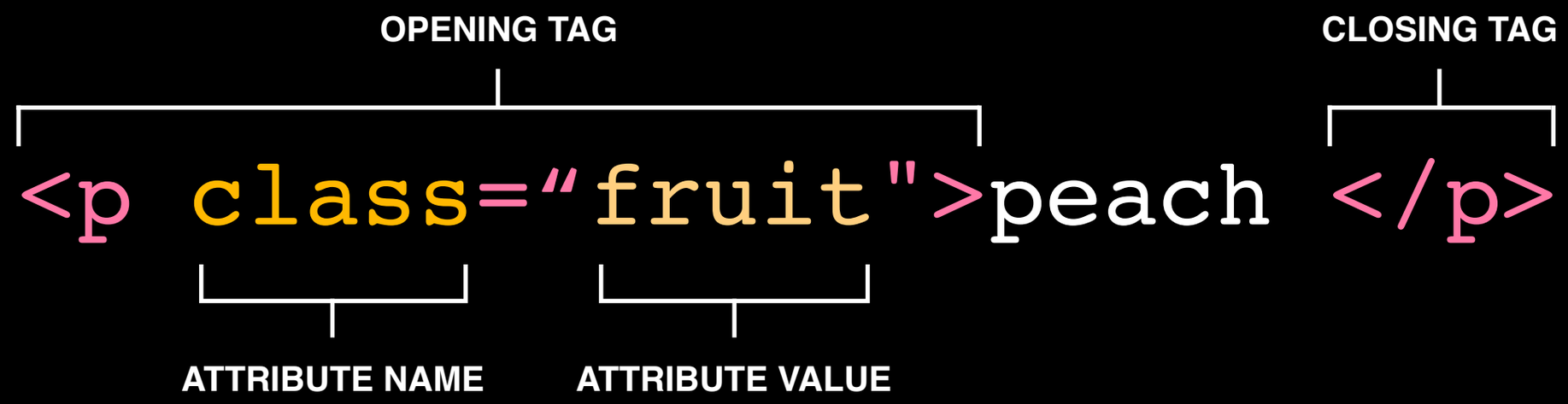
Queens College Art Department | Fall 2019
Professor Sofia Paraskeva | sofia.paraskeva@qc.cuny.edu
webdesign2019fall.copperbluemedia.com

Week 12: 11/22/2019

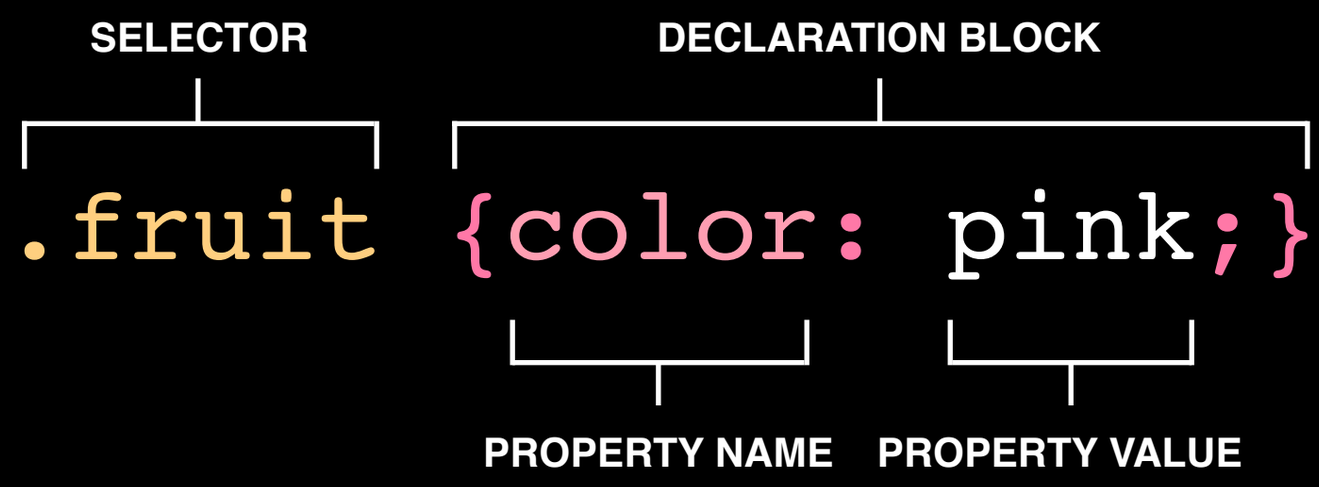
ARTS 214 – 01
Friday 10:00 am – 1:50 pm
I-Building 203

HTML & CSS Review

HTML



CSS



What is JavaScript?

- Object-Orientated Programming Language!
- Can manipulate DOM elements!
- Works on all modern browsers and devices!
- Not related to Java!

What are the possibilities?

- Add interactivity to HTML code!
- Can react to events (i.e. onLoad, onClick)!
- Browser detection!
- Complex algorithms!

1

The browser receives a page as HTML code

2

The browser creates a model of the page and stores it in memory

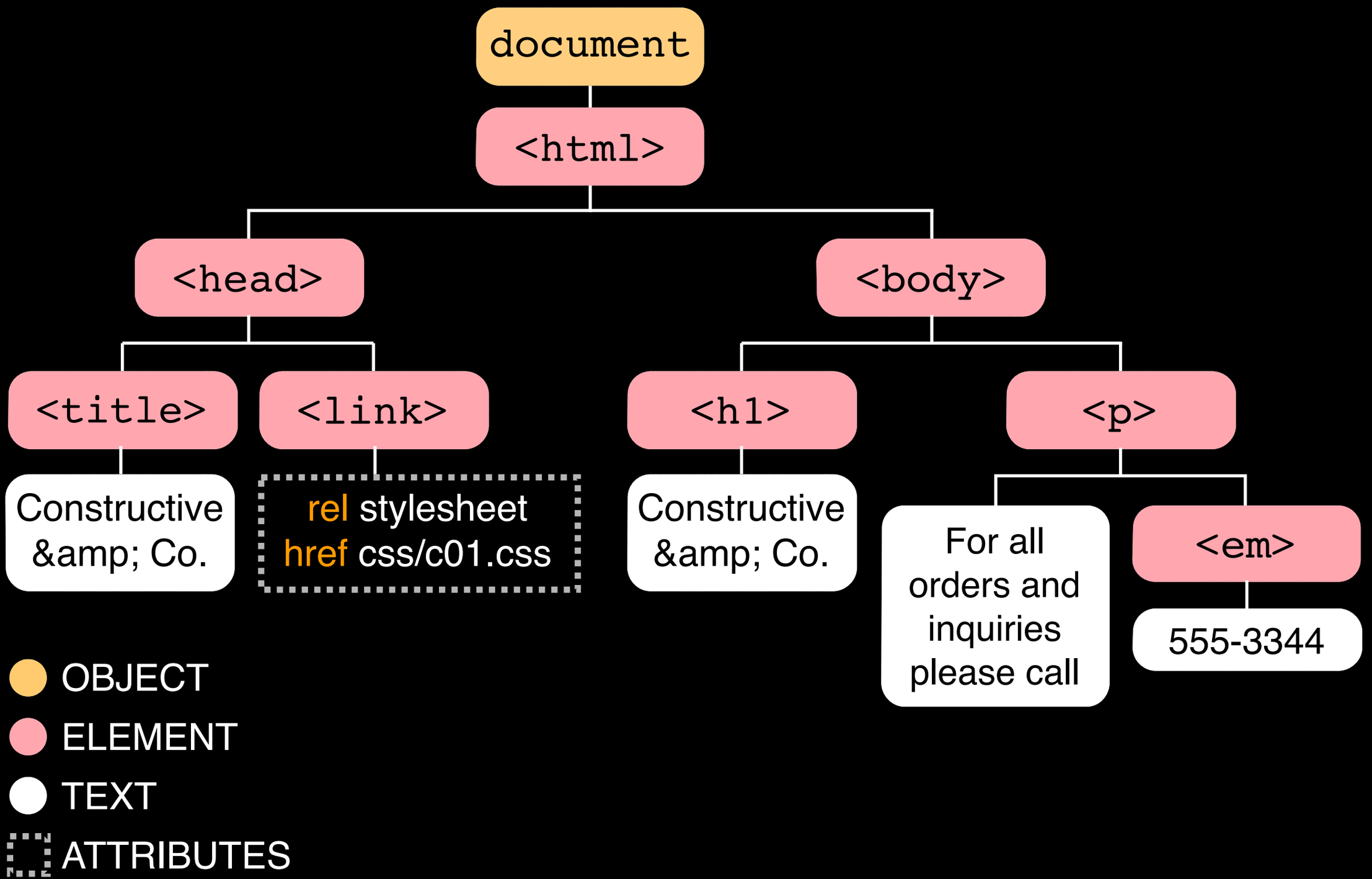
3

It shows the page on the screen using a rendering engine

The browser receives an HTML page

```
<!DOCTYPE html>
<html>
  <head>
    <title>Constructive &amp;amp; Co.</title>
    <link rel="stylesheet" href="css/c01.css" />
  </head>
  <body>
    <h1>Constructive &amp;amp; Co.</h1>
    <p>For all orders and inquiries please call
     <em>555-3344</em></p>
  </body>
</html>
```

The browser creates a model of the page and stores it in memory

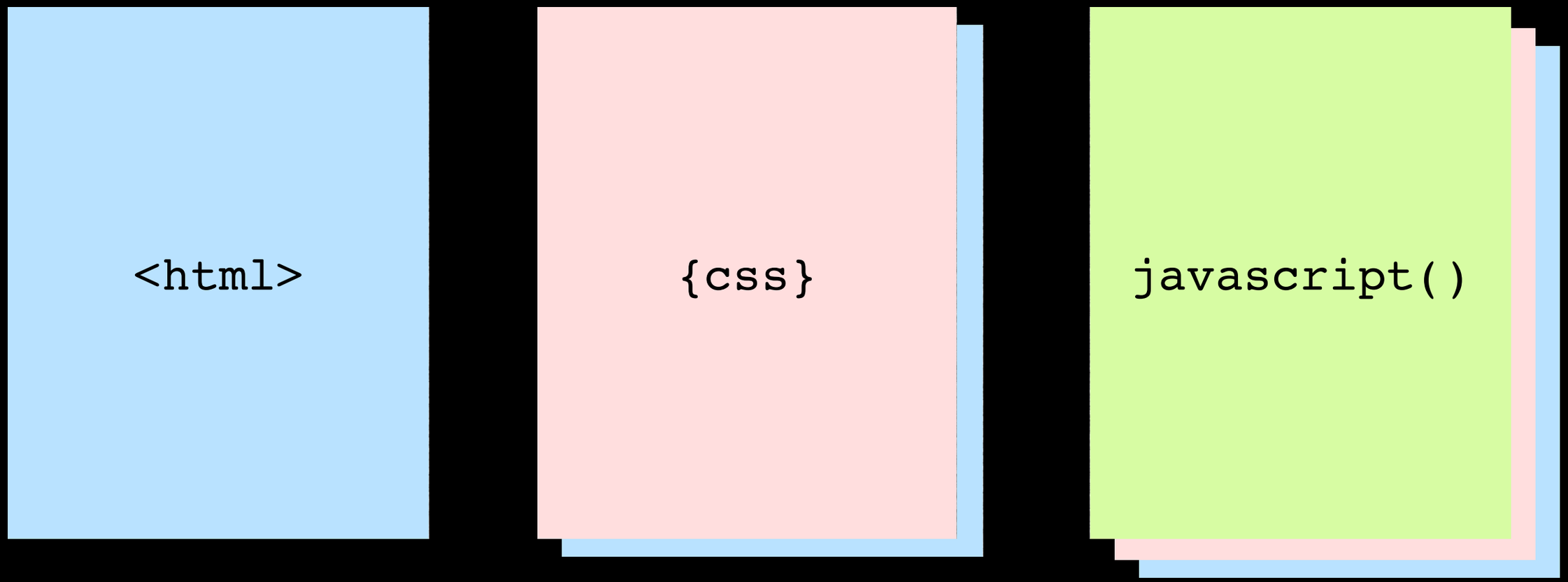


The browser shows the page on screen using a rendering engine

JavaScript

* Note how: Images were added in CSS as background images. The <h1> tag contains the name of the company. The text-indent property is used to take it off screen . Instead, in the place of the <h1> we see the logo but the information is still there for accessibility reasons.

JavaScript



CONTENT LAYER

.html files

This is where the content of the page lives. The HTML gives the page structure and adds semantics

PRESENTATION LAYER

.css files

The CSS enhances the HTML page with rules that state how the HTML content is presented (backgrounds, borders, box dimensions, colors, fonts, etc).

BEHAVIOR LAYER

.js files

Change how the page behaves adding interactivity using JAVASCRIPT.

JavaScript: Progressive Enhancement

HTML ONLY

Focuses on content and works on all devices, is accessible to all users and loads quickly on slow connections.

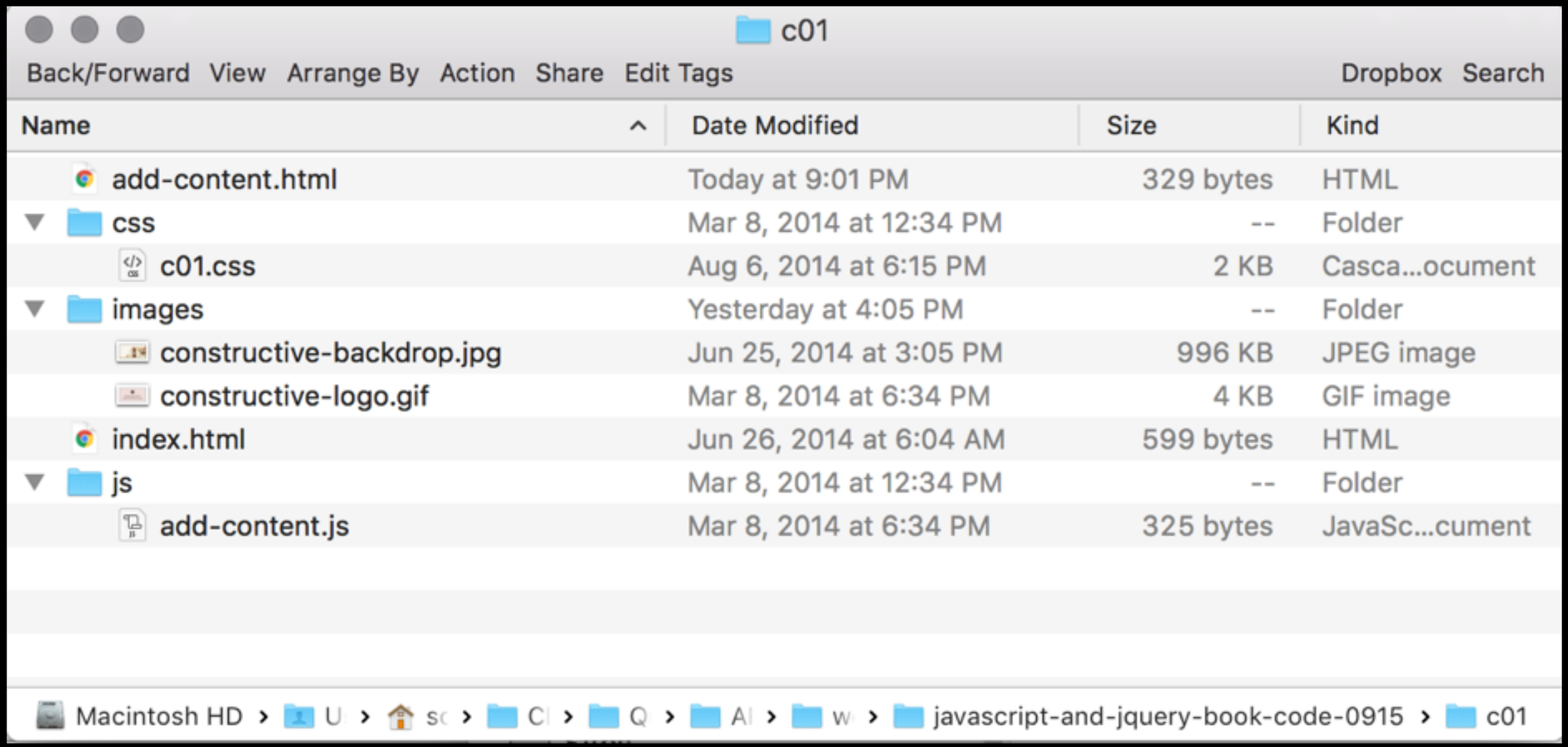
HTML + CSS

You can use different style sheets with the same content to create different views of the same data.

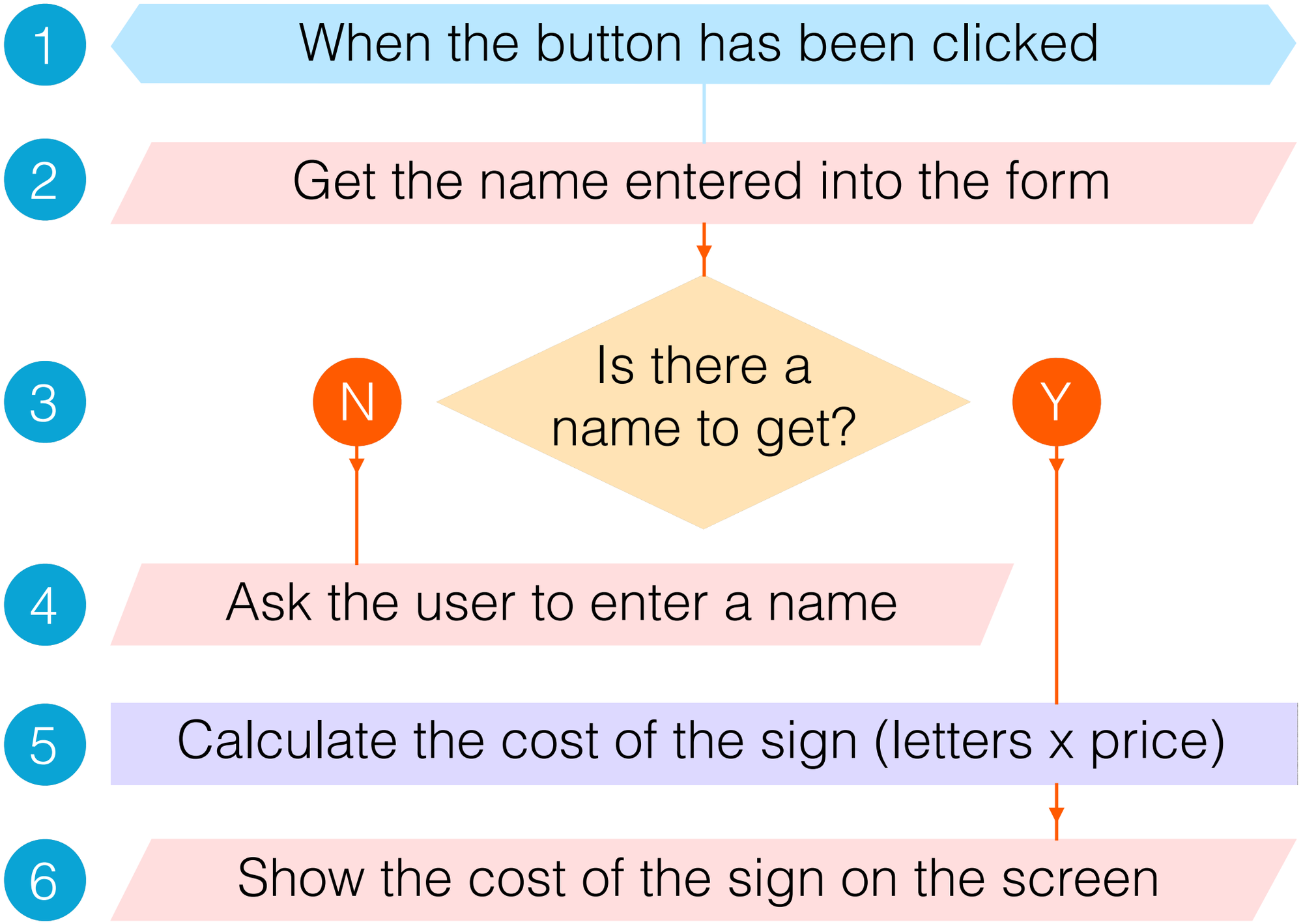
HTML + CSS + JAVASCRIPT

Enhances the usability of the page or the experience of interacting with the site.

JavaScript


c01
Back/Forward View Arrange By Action Share Edit Tags Dropbox Search
Name
Date Modified
Size
Kind
add-content.html Today at 9:01 PM 329 bytes HTML
css Mar 8, 2014 at 12:34 PM -- Folder
c01.css Aug 6, 2014 at 6:15 PM 2 KB Casca...ocument
images Yesterday at 4:05 PM -- Folder
constructive-backdrop.jpg Jun 25, 2014 at 3:05 PM 996 KB JPEG image
constructive-logo.gif Mar 8, 2014 at 6:34 PM 4 KB GIF image
index.html Jun 26, 2014 at 6:04 AM 599 bytes HTML
js Mar 8, 2014 at 12:34 PM -- Folder
add-content.js Mar 8, 2014 at 6:34 PM 325 bytes JavaSc...cument
Macintosh HD
javascript-and-jquery-book-code-0915
c01

JavaScript: Sketching out Tasks in a Flowchart



JavaScript: Statements

A statement is an individual instruction the computer must follow. Each statement starts on a new line and ends with a semicolon. The semicolon also tells the javaScript interpreter when a step is over indicating that it could move to the next step. Statements can be organized into code blocks. A code block { } can contain multiple statements.

JavaScript: Statements

Each of the lines of code in green is a statement

```
var today = new Date();
var hourNow = today.getHours();
var greeting;

if (hourNow > 18) {
    greeting = 'Good evening!';
} else if (hourNow > 12) {
    greeting = 'Good afternoon!';
} else if (hourNow > 0) {
    greeting = 'Good morning!';
} else {
    greeting = 'Welcome!';
}
document.write('<h3>' + greeting + '</h3>');
```

A script temporarily stores data in variables. For example, to calculate the area on a rectangle we multiply:

width x height = area

For the computer to carry out the same task, it needs to follow a number of steps in a certain order:

1. Remember the value for width
2. Remember the value for height
3. Multiply width by height
4. Return the result to the user

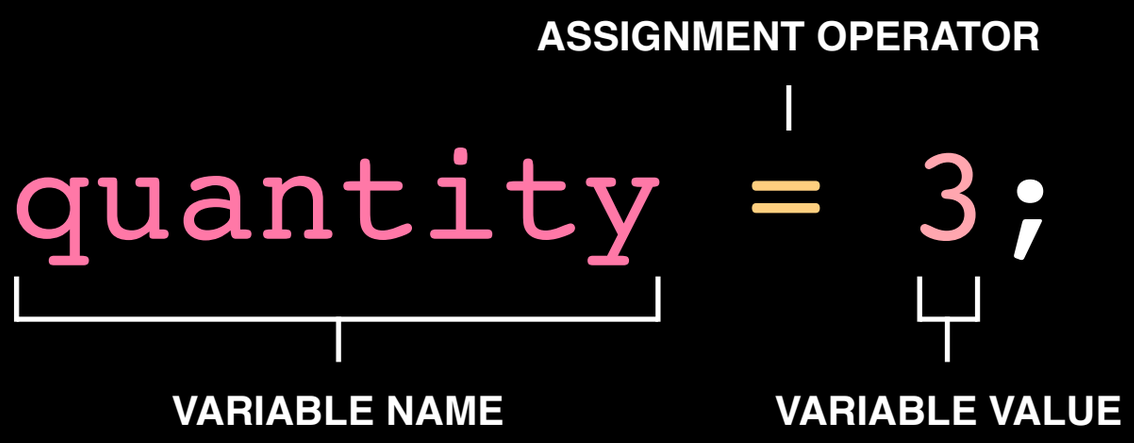
Part of these steps are to create **variables** for *width* and *height* to store their value before calculating the area.

JavaScript: Variables

Before using them **variables** must be **declared**. This involves creating a variable and giving it a name (*identifier*).

JavaScript: Variables

After a variable has been declared, a **value** can be **assigned** to it.
If a variable is not assigned a value it is **undefined**.



* Where a variable is declared can have an effect on the variable's **scope**, i.e. whether the rest of the script can use it.

JavaScript: Expressions

An **expression** *evaluates* (results in) a single value.

```
var color = 'beige';
```

The value of color is now beige.

Expressions can use two ore more values to return a single value eg. **operators**

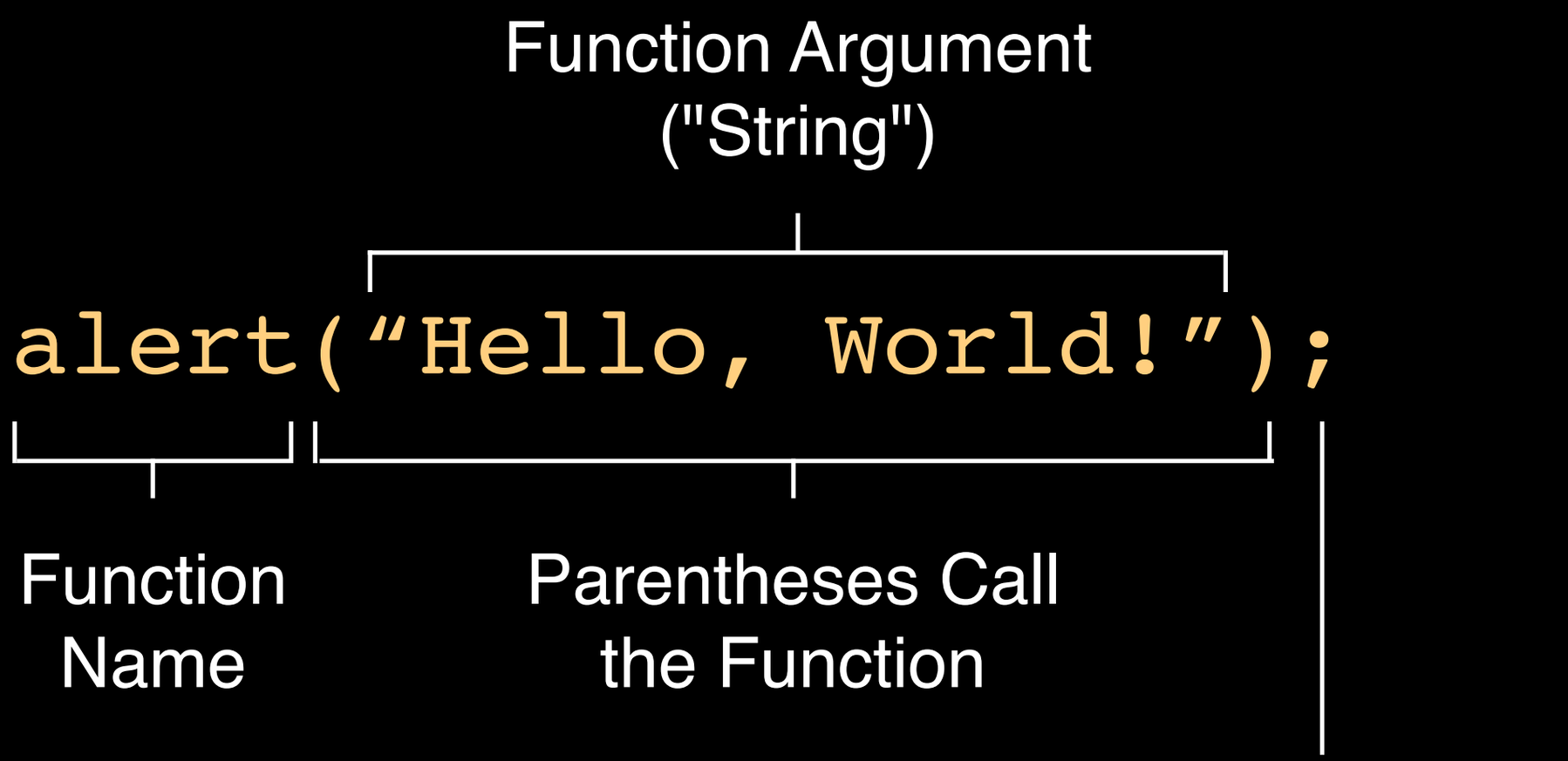
```
var area = 3 * 2;
```

The value of area is now 6.

What we will cover

- Types!
- Operators!
- Conditional Code!
- Loops!
- Arrays!
- Objects!
- Functions!
- Scope!

Simple Line of JS



External JS

HTML & JS

```
<html>
  <head>
    <title>Using External JS</title>
    <script src="/path/to/example.js">
    </script>
  </head>
  <body>
  </body>
</html>
```

External JS

HTML & JS

```
<!DOCTYPE html>
<html>
  <head>
    <title>Constructive &amp;amp; Co.</title>
    <link rel="stylesheet" href="css/c01.css" />
  </head>
  <body>
    <h1>Constructive &amp;amp; Co.</h1>
    <script src="js/add-content.js"></script>
    <p>For all orders and inquiries please call
     <em>555-3344</em></p>
  </body>
</html>
```

External JS

JS

```
var today = new Date();
var hourNow = today.getHours();
var greeting;

if (hourNow > 18) {
    greeting = 'Good evening!';
} else if (hourNow > 12) {
    greeting = 'Good afternoon!';
} else if (hourNow > 0) {
    greeting = 'Good morning!';
} else {
    greeting = 'Welcome!';
}

document.write('<h3>' + greeting + '</h3>');
```

External JS

RESULT

External JS

HTML & JS

```
<!DOCTYPE html>
<html>
  <head>
    <title>Constructive &amp;amp; Co.</title>
    <link rel="stylesheet" href="css/c01.css" />
  </head>
  <body>
    <h1>Constructive &amp;amp; Co.</h1>
    <p>For all orders and inquiries please call
     <em>555-3344</em></p>
    <script src="js/add-content.js"></script>
  </body>
</html>
```

* Javascript runs where it is found in the HTML. It's good practice to add scripts just before the </body> tag in the HTML document.

External JS

RESULT

Internal JS

HTML & JS

```
<html>
   <head>
      <title>Using Internal JS</title>
      <script>
      alert( "Hello World!" );
      </script>
   </head>
   <body>
   </body>
</html>
```

Internal JS

HTML & JS

```
<!DOCTYPE html>
<html>
  <head>
    <title>Constructive &amp;amp; Co.</title>
    <link rel="stylesheet" href="css/c01.css" />
  </head>
  <body>
    <h1>Constructive &amp;amp; Co.</h1>
    <script>document.write('<h3>Welcome!</h3>');
    </script>
    <p>For all orders and inquiries please call
     <em>555-3344</em></p>
  </body>
</html>
```

Internal JS

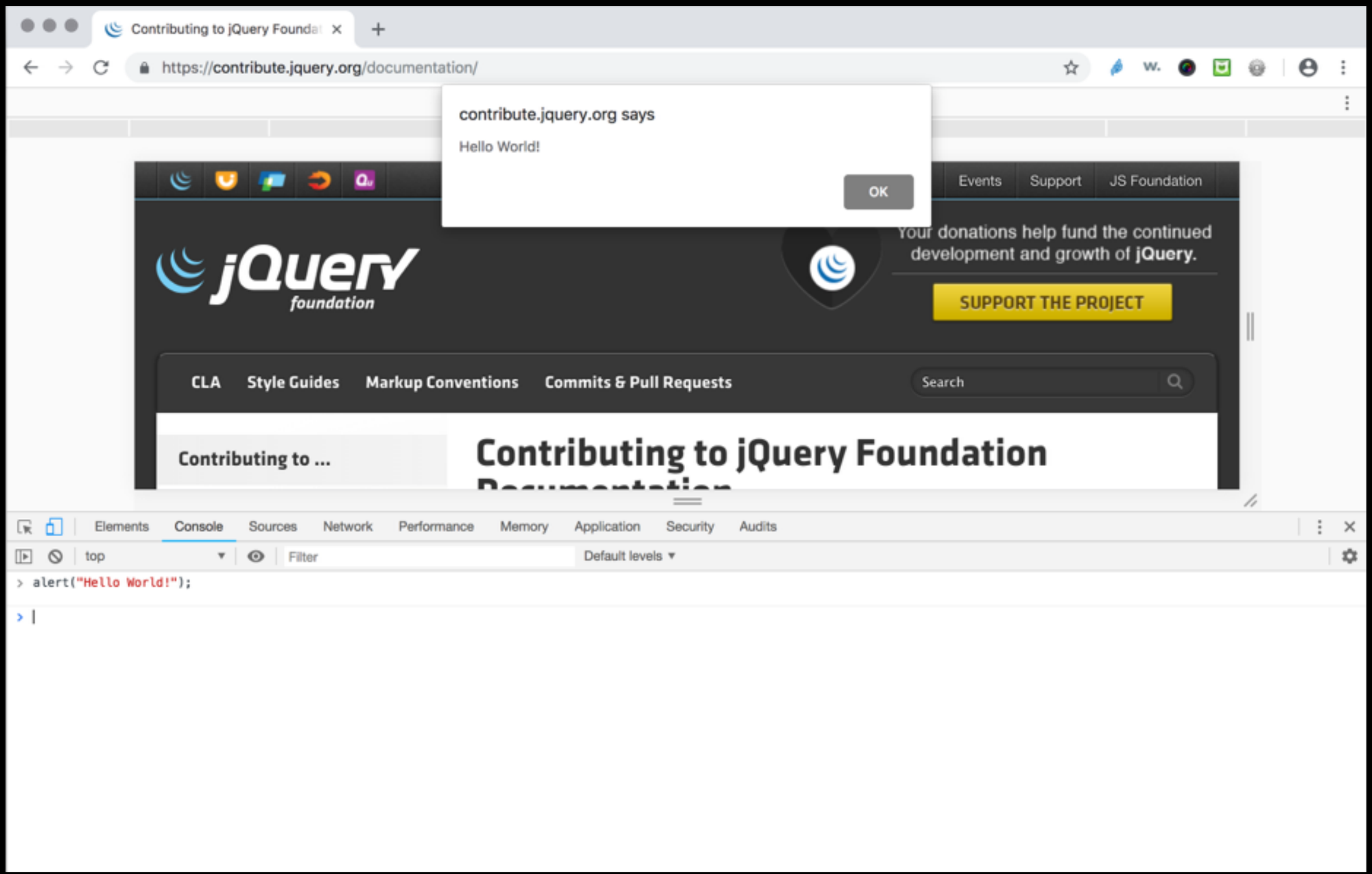
RESULT

Inline JS

HTML & JS

```
<html>
   <head>
      <title>Using Inline JS</title>
   </head>
   <body>
      <button onClick="alert('Hello');">
      Click me</button>
      <a href="javascript:alert('Hello');">
      Click me too</a>
   </body>
</html>
```

Writing code in the Console



Syntax Basics

- Comments // or /* … */!
- Whitespace is also ignored!
- Reserved words (i.e. true, false, if, else, for)!
- Can create variables !
- var someNumber = 4;!
- var someString = “abc”;!
- var someFunction = function() {…};!

Primitive Types

- Strings (i.e. “Hello World!”)!
- Numbers (i.e. 1.618 or 10)!
- Booleans (true or false)!
- null (the absence of a value)!
- undefined (no value assigned)!

Object Types

- Object!
- Array!
- Functions!

OBJECT TYPE: HOTEL

EVENT	happens when:	method called:
book	reservation is made	makeBooking()
cancel	reservation is cancelled	cancelBooking()

EVENT	what it does
makeBooking()	increases value of *bookings* property
cancelBooking()	decreases value or *bookings* property
checkAvailability()	subtracts value of *bookings* property from value of *rooms* property and returns number of rooms available

PROPERTIES	
name	Quay
rating	4
rooms	42
bookings	22
gym	false
pool	true

OBJECT TYPE: WINDOW

PROPERTIES

location	http://www.javascriptbook.com/

OBJECT TYPE: DOCUMENT

PROPERTIES

URL	http://www.javascriptbook.com/
lastModified	09/04/2018 15:33:37
title	Learn Javascript & jQuery - A book that teaches you in a nicer way

JavaScript: The Document Object

OBJECT TYPE: DOCUMENT

PROPERTIES

URL	http://www.javascriptbook.com/
lastModified	09/04/2018 15:33:37
title	Learn Javascript & jQuery - A book that teaches you in a nicer way

EVENT	happens when:
load	page and assets have finished loading
click	user clicks the mouse over the page
keypress	user presses down on a key

METHOD	what it does:
write()	adds new content to the document
getElementById()	accesses an element when you state its id attribute

Basic Operators

JS

```
var foo = "Hello";
var bar = "world”;
console.log(foo + " " + bar);

// Hello world
2 * 3 // 6
2 / 3 // 0.666666666

var i = 1;
console.log( i++ ); // 1
console.log( i ); // 2
```

Comparison Operators JS

```
var foo = 1, bar = 0, baz = '1';

foo == bar; // does equal-false
foo != bar; // doesn't equal-true

foo > bar; // greater than-true
foo < bar; // less than-false
foo <= bar; // less than or equal to-true

foo == baz; // does equal-true
foo === baz; // absolutely equals-false
```

JavaScript: Operators

ASSIGNMENT OPERATORS

```
var color = 'beige';
```

The value of color is now beige.

COMPARISON OPERATORS

```
buy = 3 > 5;
```

The value of buy is false.

ARITHMETIC OPERATORS

```
var area = 3 * 2;
```

The value of area is now 6.

LOGICAL OPERATORS

```
buy = (5 > 3) && (2 < 4);
```

The value of buy is now true.

STRING OPERATORS

```
greeting = ‘Hi ‘ + ‘Molly’;
```

The value of greeting is now `Hi Molly`.

JavaScript: Arithmetic Operators

NAME	OPERATOR	PURPOSE & NOTES	EXAMPLE	RESULT
ADDITION	+	Adds one value to another	`10 + 5`	15
SUBTRACTOIN	–	Subtracts one value from another	`10 - 5`	5
DIVISION	/	Divides two values	`10 / 5`	2
MULTIPLICATION	*	Multiplies two values	`10 * 5`	50
INCREMENT	++	Adds one to the current number	`i=10;` `i++;`	11
DECREMENT	--	Subtracts one from the current number	`i=10;` `i--;`	9
MODULUS	%	Divides two values and returns the remainder	`10 % 3`	1

Arithmetic Operators JS

```
// Create a variable for the subtotal and make a calculation
var subtotal = (13 + 1) * 5; // Subtotal is 70

// Create a variable for the shipping and make a calculation
var shipping = 0.5 * (13 + 1); // Shipping is 7

// Create the total by combining the subtotal and shipping values
var total = subtotal + shipping; // Total is 77

// Write the results to the screen
var elSub = document.getElementById('subtotal');
elSub.textContent = subtotal;

var elShip = document.getElementById('shipping');
elShip.textContent = shipping;

var elTotal = document.getElementById('total');
elTotal.textContent = total;
```

Arithmetic Operators

RESULT

String Operators

There is only one string operator the + symbol. It is used to oin the strings on either side of it.

Concatenation is the process of joining together two or more strings to create one new string.

```
var firstName = 'Ivy';

var lastName = 'Stone';

var fullName = firstName + lastName;
```

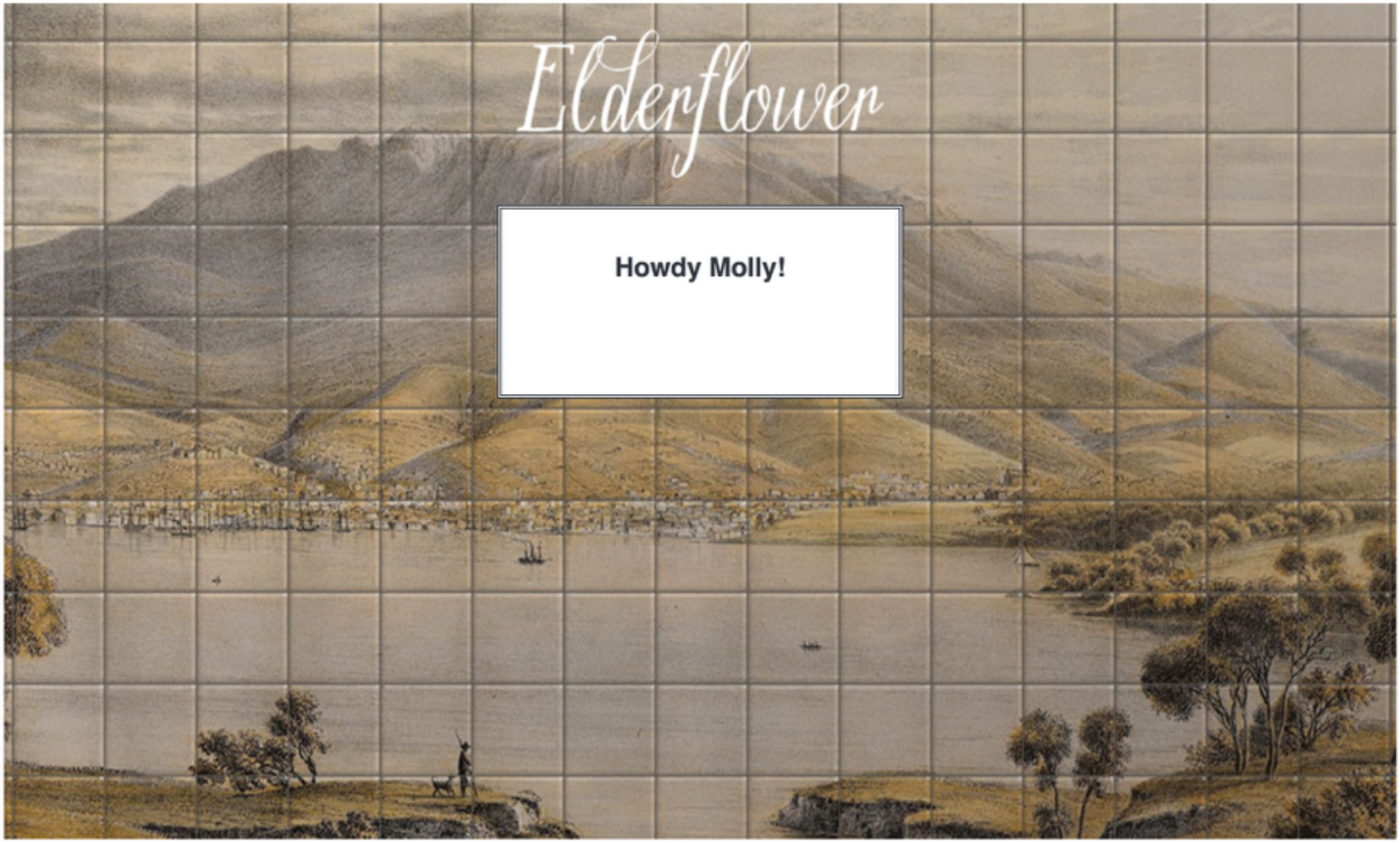
String Operators

JS

```
// Store the greeting in a variable
var greeting = 'Howdy ';

// Store the users name in a variable
var name = 'Molly';

/* Create the welcome message by concatenating the strings in the
two variables */
var welcomeMessage = greeting + name + '!';

// Get the element that has an id of greeting
var el = document.getElementById('greeting');

// Replace the content of this element with the personal message
el.textContent = welcomeMessage;
```

String Operators

RESULT



Conditional Code JS

```
// AND (&&) OR (||)
var foo = 10;
var bar = 15;

if (foo == 10 && bar >= 20) {
alert("First"); // Will never run
} else if (foo == 10 && bar <= 20) {
alert("Second"); // Will run
}
```

Loops

JS

```
for (var i = 0; i < 5; i++) {
   // Logs “try 0”, “try 1”, … “try 4”
   console.log("try " + i);
}

var i = 0;
while (i < 100) {
   // Will execute 100 times
   console.log(”Currently at " + i);
   i++; // Increases by 1 every time
}
```

Object using a Constructor

JS

```
var person1 = new Object;

person1.firstName = "John";
person1.lastName = "Doe”;

alert( person1.firstName + " " +
person1.lastName ); // John Doe
```

Object using Object Literal Syntax JS

```
var person1 = {
   firstName: "Jane",
   lastName: "Doe"
};

alert( person1.firstName + " " +
person1.lastName ); // John Doe
```

Arrays

JS

```
var foo = [100, 80, 38];
alert( foo[ 0 ] ); // 100
alert( foo[ 2 ] ); // 38
alert( foo.length ); // 3

var bar = new Array(100);
alert( bar[ 0 ] ); // undefined
alert( bar.length ); // 100
```

Arrays

An array is a special type of variable that stores a list of values. Values in an array are separated by commas. Different **data types** (*Boolean*, *String*, *Number*), can be stored in the same array. Each array has a **length** property which holds the number of items in the array.

Arrays JS

In Javascript, an **array literal** is a list of expressions, each of which represents an array element, enclosed in a pair of square brackets ' [ ] ' . When an array is created using an array literal, it is initialized with the specified values as its elements, and its length is set to the number of arguments specified

Arrays

JS

```
// Create an array and assign it values.

var colors;
colors = ['white', 'black', 'custom'];

// Show the second item from the array.

var el = document.getElementById('colors');
el.textContent = colors[1];
```

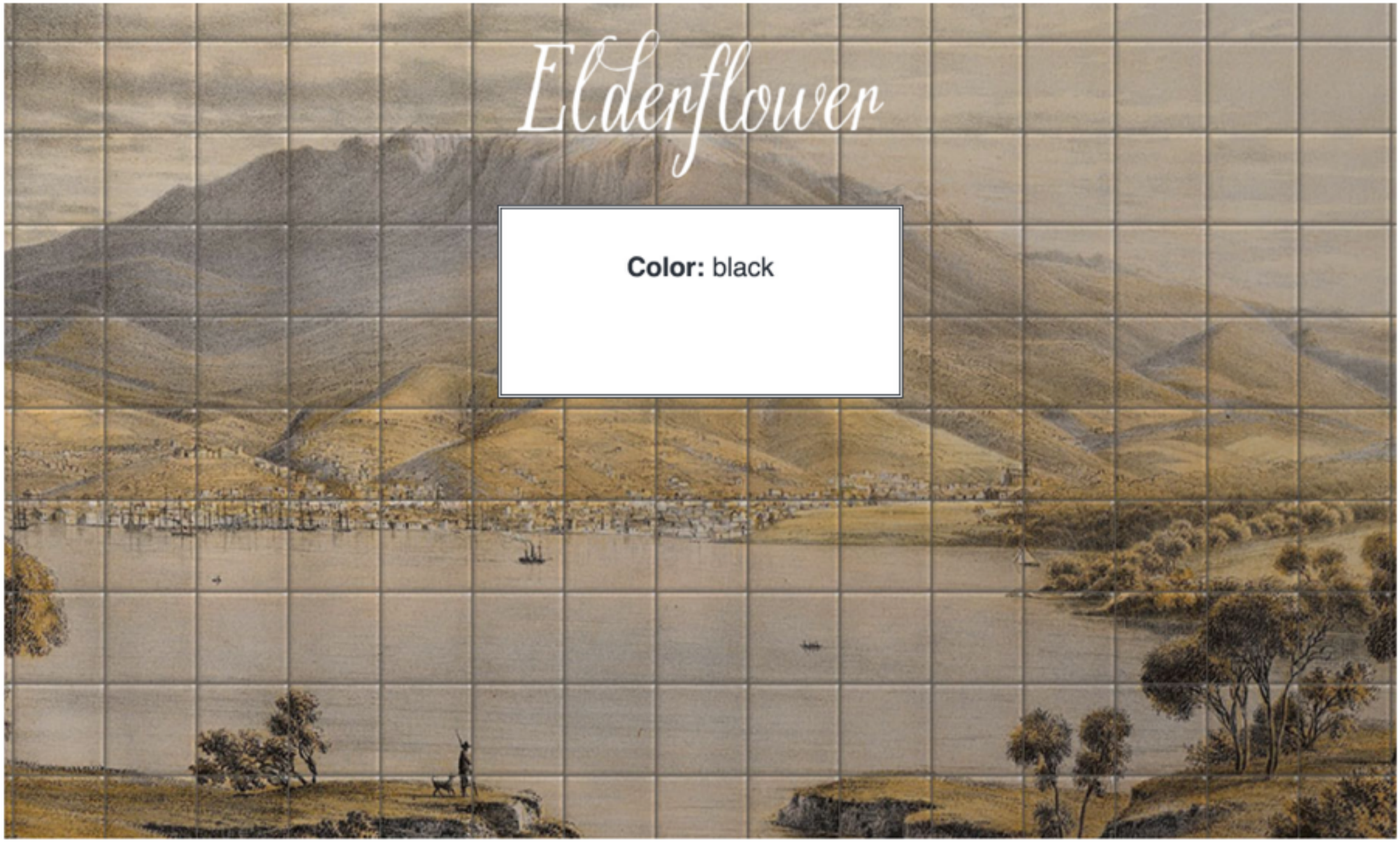
Arrays JS

Each item in an array is automatically given a number called an **index**. The numbering of an array list starts at **0**.

VALUE	INDEX
`'white'`	0
`‘black'`	1
`'custom'`	2

Arrays

RESULT



Arrays JS

```
// Create and name the variable.
// Tell the interpreter it is an array.
// Assign values inside the parentheses.

var colors = new Array('white',
                       'black',
                       'custom');

// Show the third item from the array.

var el = document.getElementById('colors');
el.textContent = colors[2];
```

Arrays

RESULT

Arrays JS

```
// Create the array and assign it values
var colors = ['white', 'black', 'custom'];

// Update the third item in the array
colors[2] = 'beige';

// Get the element with an id of colors
var el = document.getElementById('colors');
// Replace element with third item from the array
el.textContent = colors[2];
```

Arrays

RESULT

Functions, Methods & Objects

Functions consist of a series of statements that have been grouped together because they perform a specific task. A **method** is the same as a function except is created inside an object.

Objects create models of the world using data and they are made of **properties** ad **methods**.

Functions

The steps that the function needs to perform in order to perform its task are packaged in a **code block**. Functions have **parameters**, pieces of information passed to a function.

A **return keyword** is used to return a value to the code that called the function.

Functions

JS

```
function foo() {
   // Do something.
};

var foo = function() {
   // Do something.
};
```

Using Functions

JS

```
var greet = function( person, greeting ) {
  var text = greeting + ", " + person;
  return text;
};

console.log(greet("Sofia", "Hi class"));
// “Hi class, Sofia”
```

Function Returning Another Function JS

```
var greet = function( person, greeting ) {
var text = greeting + ", " + person;
return text;
};

var greeting = greet("Sofia", "Hi class”);

greeting(); // “Hi class, Sofia”
```

Functions

HTML & JS

```
<!DOCTYPE html>
<html>
  <head>
    <title>Basic Function</title>
    <link rel="stylesheet" href="css/c03.css" />
  </head>
  <body>
    <h1>TravelWorthy</h1>
    <div id="message">Welcome to our site!</div>
    <script src="js/basic-function.js"></script>
  </body>
</html>
```

Functions

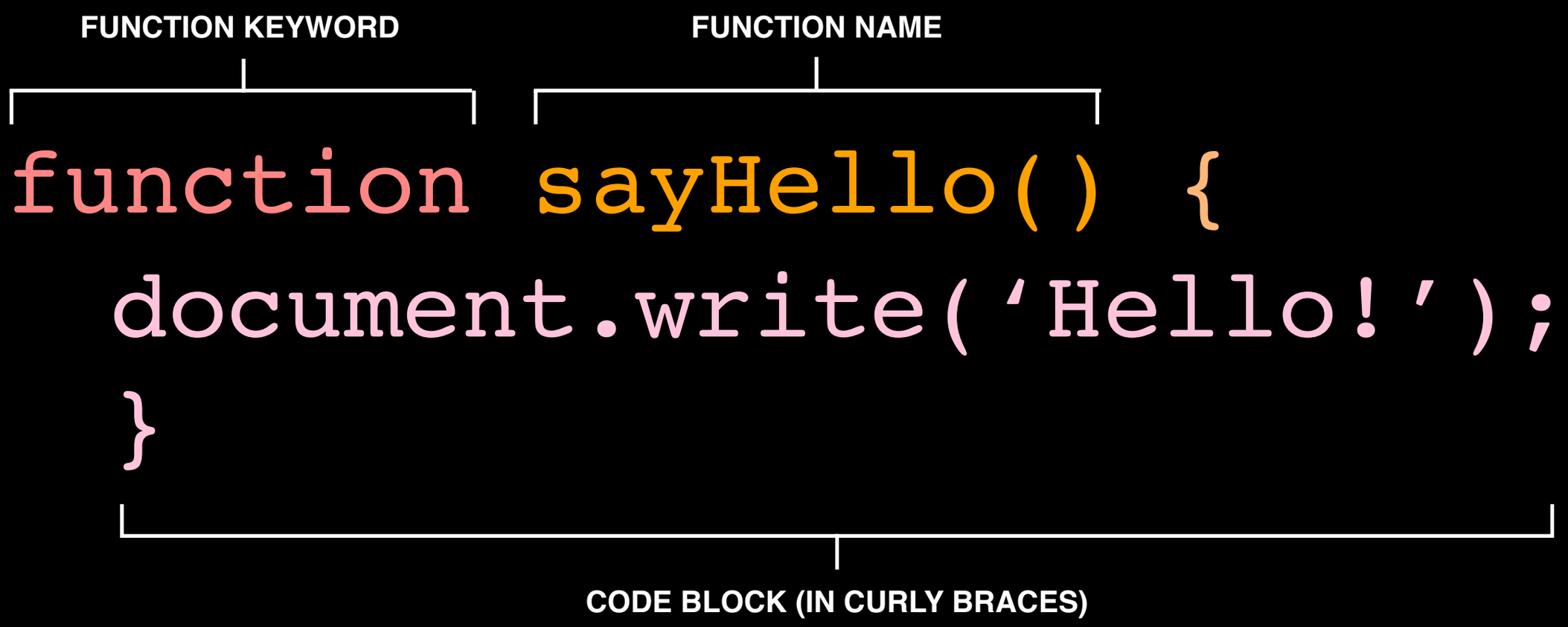
HTML & JS

```
// Create a variable called msg to hold a new message
var msg = 'Sign up to receive our newsletter
for 10% off!';

// Create a function to update the content of the
// element whose id attribute has a value of message
function updateMessage() {
  var el = document.getElementById('message');
  el.textContent = msg;
}

// Call the function
updateMessage();
```

Functions

RESULT

Declaring a Function



Calling a Function

Functions can be called multiple times in the same script.

FUNCTION NAME

```
sayHello();
```

Declaring Functions that need information

PARAMETERS

```
function getArea(width, height) {
  return width * height;
}
```

THE PARAMETERS ARE USED LIKE
VARIABLES WITHIN THE FUNCTION

Calling a Method of an object

The **document object** represents the entire web page. All browsers implement this object. You can use it by giving its name.

The **write()** method of the document object allows new content to be written into the page where the **<script>** element sits.

The document object has several **methods** and **properties**. They are known as **members** of that object.
You can access the members of an object using a *dot* between the object name and the member you want to access. It is called a **member operator**.

Whenever a method requires some information in order to work, the data is given inside the parenthesis.
Each piece of information is called a **parameter** of the method. In this case the **write()** method needs to know what to write into the page.

Global Scope

JS

```
var x;

function myFunc() {
   x = 5;
};

myFunc();

console.log(x); // 5
```

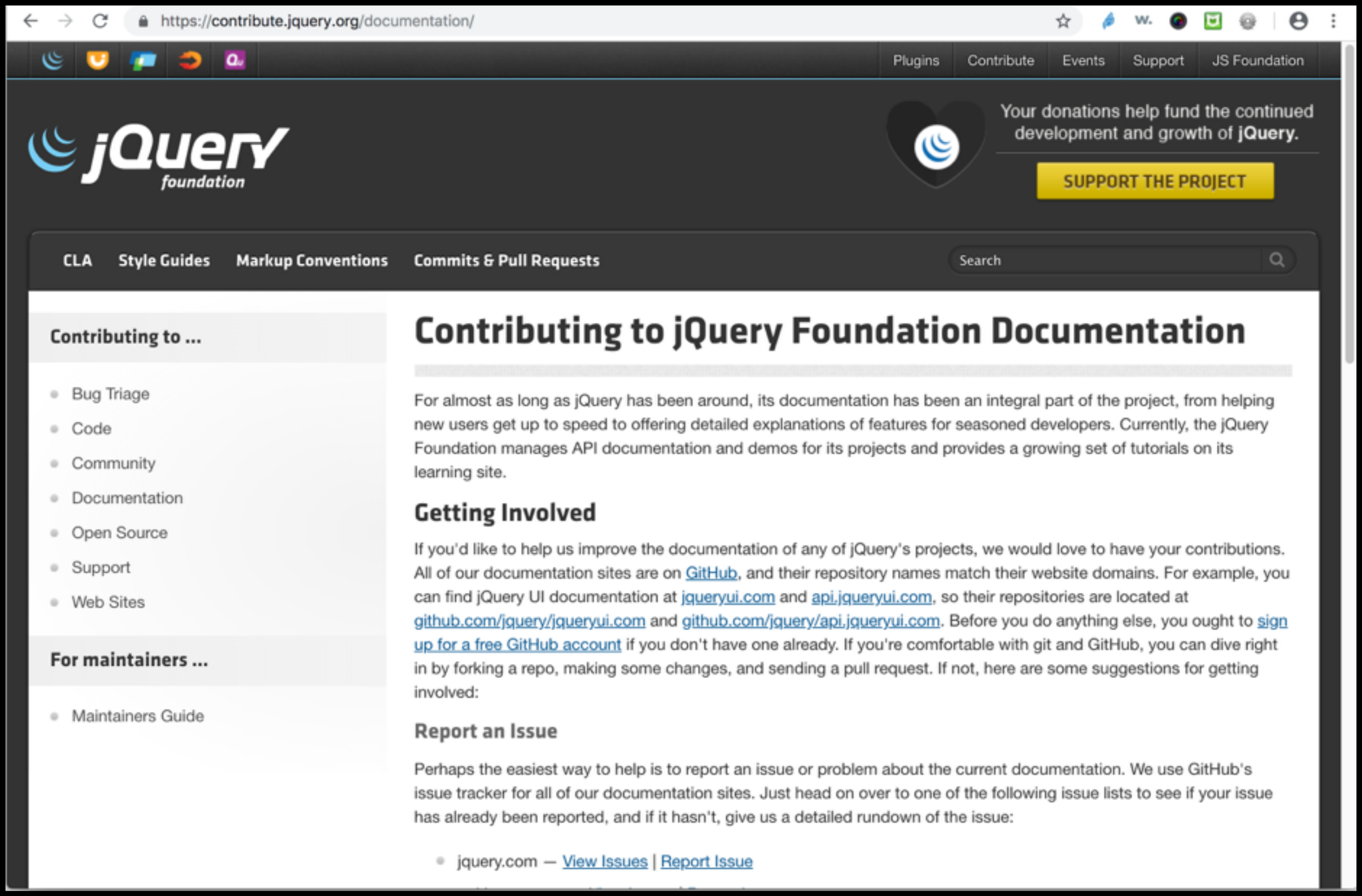
Local Scope

JS

```
function myFunc() {
   var x = 5;
};

myFunc();

console.log(x);
// ReferenceError: x is not defined
```

JavaScript Frameworks

jQuery


https://contribute.jquery.org/documentation/
Plugins
Contribute
Events
Support
JS Foundation
jQuery foundation
Your donations help fund the continued development and growth of jQuery.
SUPPORT THE PROJECT
CLA
Style Guides
Markup Conventions
Commits & Pull Requests
Search
Contributing to ...
Bug Triage
Code
Community
Documentation
Open Source
Support
Web Sites
For maintainers ...
Maintainers Guide
Contributing to jQuery Foundation Documentation
For almost as long as jQuery has been around, its documentation has been an integral part of the project, from helping new users get up to speed to offering detailed explanations of features for seasoned developers. Currently, the jQuery Foundation manages API documentation and demos for its projects and provides a growing set of tutorials on its learning site.
Getting Involved
If you'd like to help us improve the documentation of any of jQuery's projects, we would love to have your contributions. All of our documentation sites are on GitHub, and their repository names match their website domains. For example, you can find jQuery UI documentation at jqueryui.com and api.jqueryui.com, so their repositories are located at github.com/jquery/jqueryui.com and github.com/jquery/api.jqueryui.com. Before you do anything else, you ought to sign up for a free GitHub account if you don't have one already. If you're comfortable with git and GitHub, you can dive right in by forking a repo, making some changes, and sending a pull request. If not, here are some suggestions for getting involved:
Report an Issue
Perhaps the easiest way to help is to report an issue or problem about the current documentation. We use GitHub's issue tracker for all of our documentation sites. Just head on over to one of the following issue lists to see if your issue has already been reported, and if it hasn't, give us a detailed rundown of the issue:
jquery.com — View Issues | Report Issue