ScriptEd >_ Advanced Reference Course Table 2017

Table of Contents

Comments

Comments	
Comments allow you to include information for other coders and is ignored by the computer.	
`<!-- These are comments in the code.-->`	Add a comment in HTML
`// One line of comments.`	Add one line comment in JavaScript
`/* Type a long section in the comments */`	Add a section of comments in JavaScript and CSS

Command Line

Terminal Commands	
`pwd`	pwd print working directory
`ls`	The ls command is used for viewing files and directories. The ls command - the list command - shows all of the major directories filed under a given file system

`cd <directory name>`
`cd ..`
`cd ~`
	The cd command - change directory - will allow the user to change between directories. .. represents the parent directory and ~ represents the root directory.

`mv`	The mv command - move - allows a user to move a file to another directory. Just like dragging a file located on a PC desktop to a folder stored within the "Documents"
`mkdir <directory name>`	The mkdir - make directory - command allows the user to make a new directory.
`touch <file name>`	The touch command - a.k.a. the make file command - allows users to make files. Just as the mkdir command makes directories, the touch command makes files.
`rm` `rmdir`	The rm command like the rmdir command is meant to remove files. The rmdir command will remove directories and files within, the rm command will delete created files.
`clear`	The clear command clears the screen and wipes the board clean.

Git and Github

Comments	
Comments allow you to include information for other coders and is ignored by the computer.	
	Forking creates a new copy of the project on your github profile.
`git clone <your repo link here>`	Cloning a repo to your local environment makes a local copy over your repo.

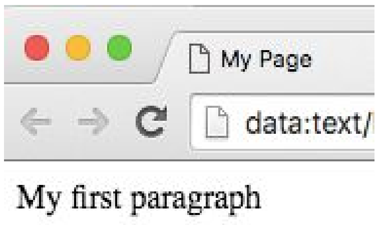
`git status`
`git add .`
`git commit -m "<your message here>"`
`git push`
	To put your local changes back into the cloud. First check the status of what changes you have made. Stage all of your changes by using the add command. Commit all of your changes. Push your changes to the repo in the cloud.

`git checkout -b gh-pages`
`git add .`
`git commit -m "<your message here>"`
`git push origin gh-pages`
	Create and checkout a new branch called gh-pages. Add and commit your changes. Push the changes to a new branch in the cloud called gh-pages.

HTML

Basic Structure of an HTML document (or webpage)

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Page</title>
  </head>
  <body>
    <p>My first paragraph</p>
  </body>
</html>
```



My first paragraph

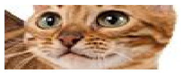
HTML Elements	Code Example	Output
paragraph	`<p>This is a paragraph.</p>`	This is a paragraph.

heading	`<h1>Heading level 1</h1>`
	`<h6>Heading level 6</h6>`
		Heading level 1

ordered list (w/ numbers)	`<ol>`
	`  <li>George Washington</li>`
	`  <li>John Adams</li>`
	`</ol>`
		1. George Washington
		2. John Adams

div	`<div>This is a div</div>`	This is a div
input	`<input>`	

Nesting in HTML		
In coding, **nesting** is when you put one tag completely inside another tag's content. It allows you to organize your page's content into multiple levels.	`<div>` `<h1>Weekday</h1>` `<p>Monday</h1>` `</div>`	On the left, the `<h1>` and `<p>` tags are nested within the `<div>` tags because the `<h1>` and `<p>` tags are completely within the opening `<div>` tag and the closing `</div>` tag.

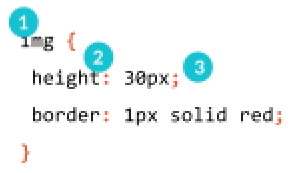
HTML elements w/ attributes	Code Example	Output
image **	`<img src="https://imgur/cats.png">`	
Link (anchor tag)	`<a href="https://www.google.com">This is a link to Google</a>`	This is a link to Google
Adding IDs*	`<p id="oneID">text</p>`	text
Adding Classes*	`<h1 class="aClass">text</h1>`	**text**
Input w/ placeholder**	`<input placeholder="type here">`	type here

*You can add an `id` and/or `class` to any HTML element (`<img>`, `<a>`, `<li>`, `<ul>`, etc.)
****Self-closing:** Does not have a closing tag.

id vs. class	
Class and **id's** are HTML attributes that you can add to HTML opening tags.	
id	class
• Each HTML element can only have one `id`. • Each page can only have one HTML element with that `id`. • In CSS and jQuery, the symbol that you use to select an `id` is a `#` (hashtag).	• You can use the same `class` on multiple HTML elements. • You can use more than one `class` on the same HTML element. • In CSS and jQuery, the symbol that you use to select a `class` is a `.` (dot).

CSS

CSS Syntax



```
img {
  height: 30px;
  border: 1px solid red;
}
```

1. Selector: Identifies the parts of your page that will be affected by this CSS rule. You can select using the tag name, id, or class.
2. Property: The thing you want to change for the element(s) you've selected. Each property should be followed by a : (colon) .
3. Value: What you want to set this property to. Each value should be followed by a ; (semicolon) .

CSS Selectors

Selector	Symbol	Code Example	What it does

element	none	`div {`
		`    width:50px;`
		`}`
			Selects every `div` and gives them a width of 50 pixels. **Other options for HTML elements** (see above in HTML elements): `p`, `body`, `h1`, `ul`, `li`, `img`, etc.

id	# hashtag	`#myID{`
		`    color:blue;`
		`}`
			Selects the one HTML element with the id `myID` and changes the font color to blue.

class	. period	`.myClass{`
		`    text-align:right;`
		`}`
			Selects all the HTML element(s) with class `myClass` and changes the text so it's right-aligned.

element, element		div, p{
		position: absolute;
		}
			Selects all <div> elements and all <p> element

element element		div a{
		font-size: "12px";
		}
			Selects all <a> elements inside <div> elements

CSS Properties and Values

Change ...	Code Examples	What it does

text	`font-family: "Comic Sans";`
	`font-size: 12px;`
	`text-align: center;`
	`color: blue;`
		Changes the font to `Comic Sans`.
		Changes font size to `12` pixels.
		Aligns the text to the `center`.
		Changes the font color to `blue`.

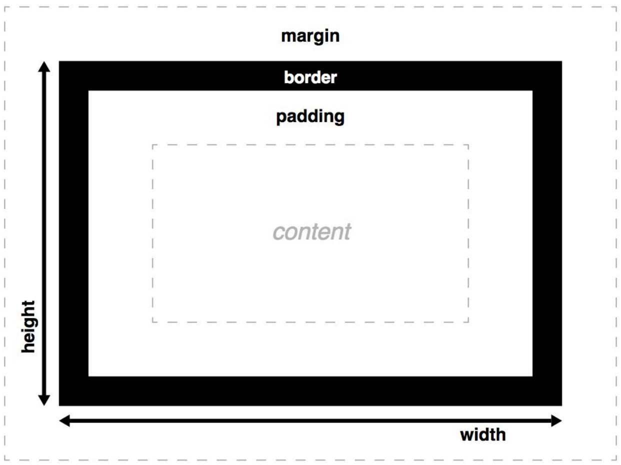
color	`background-color: #000000;`
	`color: yellow;`
		Changes the background color to that hex code, which is black.
		Changes the font color to a specific shade of `yellow`.

position	`position: fixed;`
	`position: absolute;`
		The element ...will not move while a page is scrolling.
		...will move with the page.

background	`background-color: pink;`
	`background: url("www.ex.png");`
		Changes the background color to `pink`.
		Changes the background to an image w/ URL `"www.ex.png"`

size	`width: 50px;`
	`font-size: 20px;`
		Changes the width to `50` pixels.
		Changes the font-size to `20` pixels.

CSS Layout

CSS Box Model	
All HTML elements are shaped like boxes. Each box has a content area (text, image, link, etc.) and optional surrounding padding, border, and margin areas.	
	**Box Model Properties** **Content** - (not a property) the HTML element i.e. paragraph, image, link, etc. **Padding**** - surrounds the content. (Example value: `10px`.) **Border**** - surrounds the padding. (Example value: `2px black`) Think of it like an outline around a picture. **Margin**** - surrounds the border and buffers the content from other content. (Example value: `30px`) Margin backgrounds are always transparent.

**Also has location-specific properties like `border-top`, `border-right`, `border-bottom`, `border-left`.

Bootstrap

https://getbootstrap.com

Bootstrap is a toolkit for developing with a responsive grid system and prebuilt components.

Topic	Code Examples	What it does
`CSS Link`	`<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css">`	Copy-paste the stylesheet `<link>` into your `<head>` before all other stylesheets to load CSS
`JS Script`	`<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/js/bootstrap.min.js"></script>`	Place the `<script>` right before the closing `</body>` tag. jQuery comes first, then JavaScript.

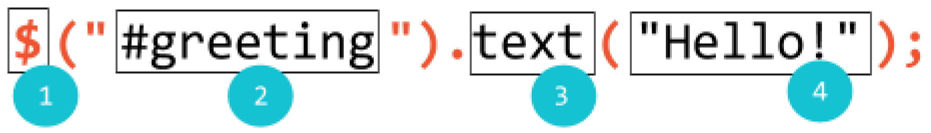
`The Grid`	`<div class=”container”>`
	`        <div class=”row”>`
	`                <div class=”col-md-12”>`
	`                        <p> Lorem Ipsum.....</p>`
	`                </div>`
	`        </div>`
	`</div>`
		container class adds predefined padding and margin to your div
		Rows are wrappers for columns
		column classes indicate the number of columns to use out of the possible 12 per row

`Components`	`https://getbootstrap.com/docs/4.0/components/`	Bootstrap comes with custom components like buttons and navbars

jQuery

jQuery Syntax

jQuery is a JavaScript library with different actions that make it easier to make your page interactive with JavaScript.



1. The $ symbol lets you know you are using jQuery, the JavaScript library.
2. The selector is exactly like a CSS selector. It selects or identifies the element on the page. You can use the name of an **HTML element** (`<p>`, `<h1>`, `<body>`), **id** (`#results`, `#div1`) or **class** (`.results`, `.div1`).
3. The jQuery action() to be performed on the element. See more options below.
4. The argument tells more information about how to change the element. Sometimes, there is no argument, i.e. `.show()`, and sometimes, there are several arguments, i.e. `.css()`.

Click Handler

1 `$("#yourID").click(function(){`	1 **When the user clicks the HTML element with an id** `yourId` **...**
2 `  //insert code here`	2 This is a comment. The computer does not read this as code.
3 `  $("h1").hide();`	3 Use jQuery to hide every `<h1>` tag.
4 `});`	4 **End of the click handler.**

Action	Code Example	What it does

Show an element.
Hide an element.
	`$(".yourClass").show();`
	`$("#yourID").hide();`
		Show all HTML elements w/ the class `yourClass`.
		Hide all HTML elements with the id `yourID`.

Replaces the content of an HTML element.	`$("body").html("<p>Hi!</p>");`	In the HTML, replace the content inside the `<body>` with `<p>Hi!</p>`.
Add/change the CSS, or style, of an element. (Change the property and/or value)	`$(".yourclass").css("color", "red");`	Add/change the CSS property `color` to `red` for all HTML elements with a class of `container`.
Add/change the text in an element.	`$("#yourID").text("You won!");`	Add/change the text to `"You won!"` for the HTML element with the id `results`.
Add/change an HTML attribute. (See page 4 for info about attributes.)	`$("img").attr("src", "http://pics.com/blah.jpg");`	Add/change the HTML attribute `src`, or source, to that URL for all `<img>` tags.
Append (add) content to an element.	`$("div").append("Bye!");`	Append, or add, the text `"Bye!"` to the end of the all the `<div>` tags.
Retrieve a value from an <input>	`var firstName = $("input").val();`	Retrieve a value from the input tag and store it in a variable named `firstName`.

Example: Retrieve a value from an input	

`<input id="myID">`
`<button id="yourID"> Go! </button>`
	Creates an input field in HTML with an id `myID`.
	Creates a button that says Go! with an id `yourID`.

`$("#yourID").click(function(){`
`  var message = $("#myID").val();`
`});`
	When the user clicks the HTML with an id `yourID` (which is the button), retrieve the value from the input field.

JavaScript

Mathematical Operators**		
Symbol	**Definition**	**Code Example**
+	Addition****	`return a + b;`
-	Subtraction	`return a - b;`
*	Multiplication	`return a * b;`
/	Division	`return a / b;`

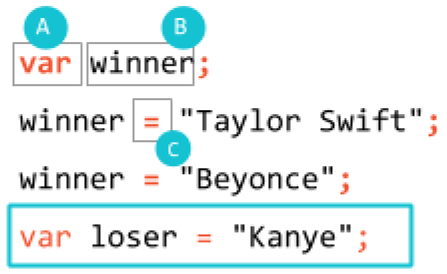
** Follow the order of operations rule **PEMDAS**: 1) Parenthesis, 2) Exponents, 3) Multiply/Divide, 4) Addition/Subtraction
****Can *ALSO* be used to concatenate, or combine, strings, not just add numbers.

Comparison Operators		
Symbol	**Definition**	**Code Example**
<	Less than	`if (number < 10)`
>	Greater than	`else if (grade > 70)`
<=	Less than or equal to	`if (points <= 100)`
>=	Greater than or equal to	`else if (age >= 16)`
===	Equal to	`if (username === "scripted1")`
!==	NOT equal to	`else if (password !== "p@$sw0rd")`

Logical Operators		
Symbol	**Definition**	**Code Example**
&&	And	`if (number > 10 && number < 20)`
||	Or	`if (grade > 65 || passedRegents)`
!	Not	`if (!(number < 10))`

Variable Syntax

Variables are containers for storing data values.



```
var winner;
winner = "Taylor Swift";
winner = "Beyonce";
var loser = "Kanye";
```

Parts:

A. The keyword `var` indicates declaring a variable
B. The variable name `winner`
C. The equal = sign assigns a value.

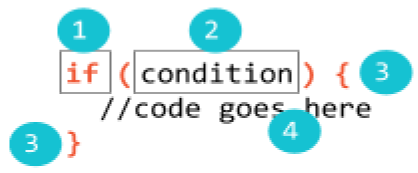
Line 1: Declares a variable and gives it a name `winner`.
Line 2: Assigns a value to the variable `winner`.
Line 3: Re-assigns a different value to the variable `winner`. The value of `winner` is no longer `"Taylor Swift"`. It is now `"Beyonce"`.
Line 4: A shortcut! Declares a variable named `loser` and assigns it a value `"Kanye"` all in one line of code.

Control Flow

Conditional Syntax

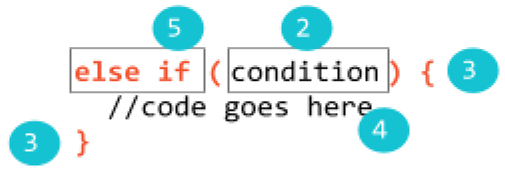
Conditional statements are used to perform different actions based on conditions.

1) if statement



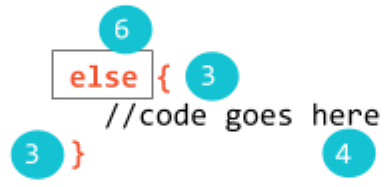
```
if (condition) {
  //code goes here
}
```

2) else-if statement



```
else if (condition) {
  //code goes here
}
```

3) else statement



```
else {
  //code goes here
}
```

Conditional Statements can be created using a combination of the three statements on the left.

1. The keyword `if` indicates this is an if statement
2. The condition goes between the ( ); the result should be true or false. If you need multiple conditions, you will need an else-if statement.
3. Curly brackets indicate the body of the condition statement.
4. Body - This is the code that executes if the condition is true. If the condition is false, then the code will NOT execute.
5. The keyword `else if` indicates an else-if statement.
6. The keyword `else` indicates an else statement.

An if statement is required to create a conditional statement, while else-if statements and else statements may or may not be used. You can also use more than one else-if statement.

Basic Conditional Statement Example

	Code		Explanation
1	`var number = 3;`	1	Declare variable named number and assign it a value of 3.
2	`if (number < 5) {`	2	If the variable number is less than 5...
3	`  $("#buttonID").hide();`	3	Hide the HTML element with the id `buttonID`.
4	`} else {`	4	Or else...
5	`  $("#buttonID").show();`	5	Show the HTML element with the id `buttonID`
6	`}`	6	End of conditional statement.

Data Types and Structures

Value Types		

Number	Duh... you know what a number is... No quotation marks, may start with a + or -, may include a decimal.	`var temperature = -1;`
		`var price = 5.99;`

String	Always inside single (' ') or double (" ") quotes. Can be an empty string " ". Can include letters, spaces, symbols, numbers... as long as it's in quotes.	`var greeting = "Kevin is here!";`
		`var space = ' ';`
		`var price = "$5.99";`

Boolean	true or false has no quotation marks	`var scriptedIsAmazing = true;`
		`var brunoMarsOverrated = false;`

Array	A list of multiple values separated by commas inside square brackets [ ]	`var oddNumbers = [1,3,5,7,9];`
		`var airport = ["JFK", "LGA", "SFO"];`

Object	A collection of properties separated by commas inside curly brackets { }. A property is an association between a name (or key) and a value separated by a colon :	`var student = {`
		`    name : "Erica",`
		`    school : "Columbia HS",`
		`};`

String Method and Properties

Action	Code Example	What it does

. length property returns the length of a string	var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
	var sln = txt.length;
		Returns the length of the array. `sln` will evaluate to `26`.

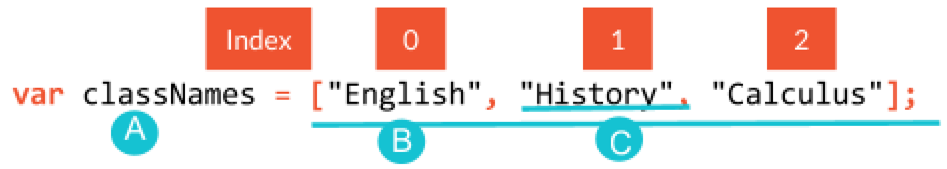
slice() extracts a part of a string and returns the extracted part in a new string.	var str = "Apple, Banana, Kiwi";
	var res = str.slice(7, 13);
		The method takes 2 parameters: the starting index (position), and the ending index (position). This example slices out a portion of a string from position 7 to position 13. The result of `res` is `"Banana"`.

A string is converted to uppercase with toUpperCase() or to lower case with toLowerCase():	var text1 = "Hello World!";
	var text2 = text1.toUpperCase();
	var text3 = text1.toLowerCase();
		The result of text1 is `"Hello World!"`.The result of text2 is `"HELLO WORLD!".` The result of text1 is `"hello world!"`

A string can be converted to an array with the split() method:	var txt = "a b c d e"; txt.split(" ");	Converts `txt` from a string into an array splitting on each space. The result of `txt` is the array `["a","b","c","d","e"]`.

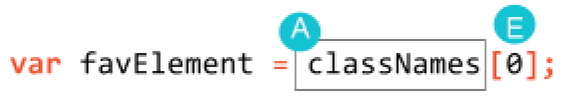
Array Syntax

An **array** is a list-like way to store data.



```
var classNames = ["English", "History", "Calculus"];
```

A. Declare a variable called `classNames`.
B. An array is a list of values — they can be any JavaScript value including numbers, strings, objects, functions, and even arrays. Square brackets start and end an array.
C. Each array element, or individual item (i.e. `"History"`) in the array, is separated by a comma.



```
var favElement = classNames[0];
```

E. To use a specific array element, use the array index. It (see above) represents the location of an array element and always begins with 0. The array index uses the name of the array + [the index surrounded by square brackets]. The value of `favElement` is `"English"`.

Array Methods and Properties

Action	**Code Example**	**What it does**

.length tells us how many items there are in the array	var fruits = ["Banana", "Orange", "Apple", "Mango"];
	var x = fruits.length;
		Returns the number of the elements in the array. `x` will evaluate to `4`.

The pop() method removes the last element from an array:	var fruits = ["Banana", "Orange", "Apple", "Mango"];
	var x = fruits.pop();
		Removes the last element ("Mango") from fruits. The result of `fruits` is `["Banana", "Orange", "Apple"]`

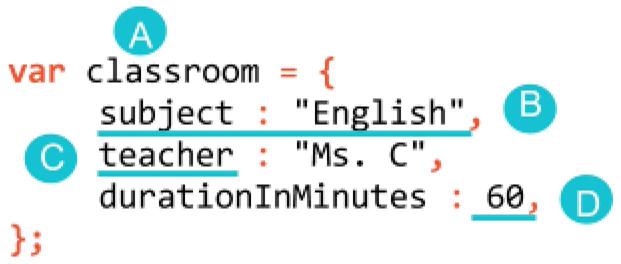
The push() method adds a new element to an array (at the end):	var fruits = ["Banana", "Orange", "Apple", "Mango"];
	fruits.push("Kiwi");
		Adds a new element ("Kiwi") to fruits. The result of `fruits` is `["Banana", "Orange", "Apple", “Kiwi”]`

Array elements are accessed using their index number:	var fruits = ["Banana", "Orange", "Apple", "Mango"];
	fruits[0] = "Kiwi";
		Changes the first element of fruits to `"Kiwi"`. The result of `fruits` is `["Kiwi", "Orange", "Apple", “Kiwi”]`

The join() method also joins all array elements into a string.	var fruits = ["Banana", "Orange","Apple", "Mango"];
	var x = fruits.join(" * ");
		Joins all elements into a string separated by `" * "`. The result of `x` is `"Banana * Orange * Apple * Mango"`.

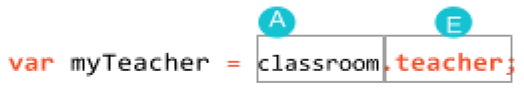
Object Syntax

An **object** is a way to store data as properties with keys and values.



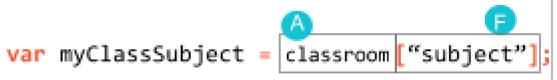
```
var classroom = {
    subject : "English",
    teacher : "Ms. C",
    durationInMinutes : 60,
};
```

A. Declare a variable called `classroom`.
B. An **object** is a collection of properties separated by a commas
C. Each property, in the array has a unique name or key used to identify it
D. Each key has and corresponding value separated by a colon `:` .



```
var myTeacher = classroom.teacher;
```

E. To access a specific value from an object, use the corresponding key in dot notation . The value of `myTeacher` is `"Ms. C"`.



```
var myClassSubject = classroom["subject"];
```

F. To access a specific value from an object, you can also use the corresponding key in bracket notation. The value of `myClassSubject` is `"English"`.

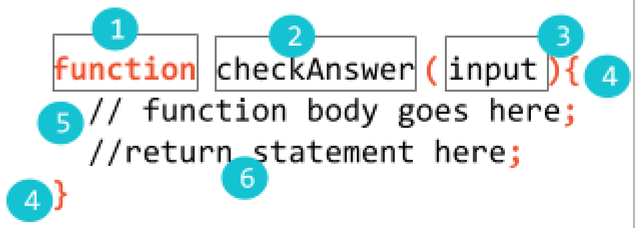
```
classroom["durationInMinutes"] = 45;
classroom.durationInMinutes = 45;
```

To change the value of a property you can use either dot notation or bracket notation and assign the property a new value.

Functions

Function Syntax

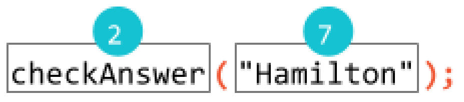
A **function** is a set of instructions-- the basic building block of a program.
A **function declaration** creates the set of instructions.



```
function checkAnswer(input){
  // function body goes here;
  //return statement here;
}
```

1. The keyword `function` is used in a **function declaration**.
2. The **name** of this function is `checkAnswer`.
3. Some functions use **parameters.** The name of this parameter is `input`. You may also accept *multiple* parameters, separated by commas.
4. **Curly brackets { }** surround the body of the function.
5. The **body** of the function is the list of instructions, enclosed in the curly brackets.
6. The **return statement** stops the function and returns a value to the caller of the function. But, not every function has a return statement.

To use the list of instructions, you must make a **function call.**



```
checkAnswer("Hamilton");
```

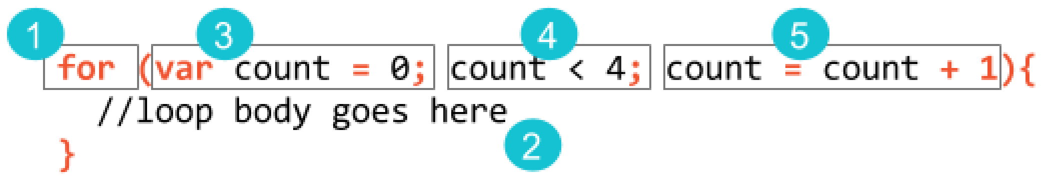
2. To **call the function**, use the function name `checkAnswer`.
7. In a function call, you should pass an **argument** for every parameter in the function declaration. The parentheses **( )** are *always* included, even if there isn't an **argument**. (see above).

Function Example with Return Statement	
1 `function compoundWord(a,b) {`	1 Declare function `compoundWord` that takes 2 parameters.
2 `      return a + b;`	2 Body: Return parameter `a` + parameter `b`.
3 `}`	3 End of function `compoundWord`.
4 `var word1 = compoundWord("can","not");`	4 Call function `compoundWord`, w/ arguments `"can"` & `"not"`.
5	5 Assign it to the variable `word1`. The value is `"cannot"`.
6 `var word2 = compoundWord("fire","work");`	6 Call function `compoundWord`, with arguments "fire" and "work". The value of variable `word2` is `"firework"`.

Iteration

For Loop Syntax

Loops repeat an action some # of times. A **for loop** repeats until a specified condition is false.



```
for (var count = 0; count < 4; count = count + 1){
  //loop body goes here
}
```

1. Always begin the for loop with the keyword `for`.
2. The loop body goes between the curly brackets. This block of code executes while the condition is true.

The Three Parts of a 'For' Loop:

3. The 1st statement, called the Initial Expression, declares a variable and value of where the loop starts. In this case, it declares a variable `count` and begins at `0`.
4. The 2nd statement, called the Condition, tells the loop how many times to run. In this case, the loop will execute code as long as `count` is less than 4. In other words, the last time the loop will run is when `count` is 3.
5. The 3rd statement, called the Increment Expression, changes the variable value incrementally. A lot of times and in this case, the loop will increment, or increase, by 1. However, it could increment by 2 or 5 or 10, etc.

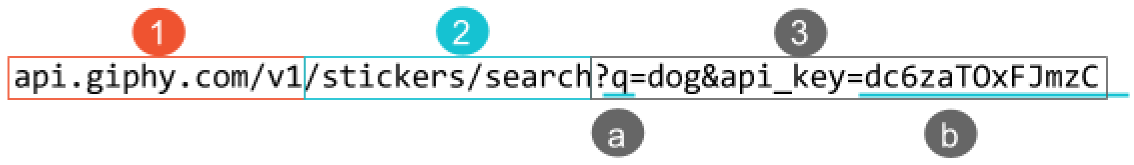
For Loop Example	
1 `for(var i=0; i=<5; i=i+1){`	1 Create a for loop that starts at 0, stops at 5, and increases by 1.
2 `  $("#yourid").append(i);`	2 Append the value of variable `i` (0, 1, 2, 3, 4, 5) to element with id `yourid`.
3 `}`	3 Exit the loop when the variable `i` is no longer less than or equal to 5.

Iterating Over an Array	
1 `var arr = ["a","b","c"];`	1 Define your array
2 `for(var i=0; i=<arr.length; i=i+1){`	2 Create a for loop that starts at 0, stops at array.length, and increases by 1
3 `  console.log(arr[i]);`	3 Use i as the index to access each item in the array
`}`	

APIs

API Request URL

API or Application Programming Interface Request URL

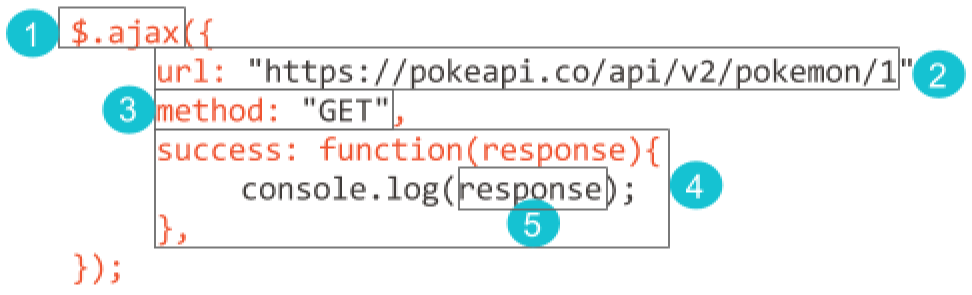


```
api.giphy.com/v1/stickers/search?q=dog&api_key=dc6zaTOxFJmzC
```

1. Base Url is the consistent part of your url. This will not change.
2. End Point refers to some object or set of objects that are exposed at an API endpoint.
3. Query String comes after the endpoint. This starts after the ? and includes the query parameters and their associated values. separated by & signs.
    a. Query Parameter
    b. Value is the data that is associated with a query parameter

AJAX Syntax

AJAX is used to retrieve data from an **API**



```
$.ajax({
    url: "https://pokeapi.co/api/v2/pokemon/1",
    method: "GET",
    success: function(response){
        console.log(response);
    },
});
```

1. Always begin the **AJAX request** with the query `$.ajax()`. The **AJAX request object** goes between the parentheses.
   Three basic properties of an AJAX request object are (there are others not listed):
2. url: Indicates where you are making the request to.
3. method: what type of request you are making. Ie. GET, POST, PUT, DELETE
4. success: the function to run upon a successful response from the API. This function takes a response as a parameter.
5. The success function uses the response object which contains all the data returned from the API