**Web Development Roadmaps**

# Front-End Roadmap

# STEP 1 Learn the Basics

Every **Web Developer** must have a basic understanding of **HTML**, **CSS**, and **JavaScript**.

**Responsive Web Design** is used in all types of modern web development.

**ECMAScript 5** (JavaScript 5) is supported in all modern browsers. Take a good look at it, especially the new array functions.

# STEP 2 Dig Deeper

When you feel comfortable with **HTML** and **CSS**, it is time to dig deeper.

You should learn how to use **Maps**, **Fonts** and **Icons** in HTML.

On the **JavaScript** side, you should learn how to access the **HTML DOM**.

You should also learn how to use **AJAX** and **JSON** for making server requests.

# STEP 3 Choose Frameworks

Now it is time to look at some **Frameworks**.

On the **CSS** side you should choose a framework for reponsive web design:  
**Bootstrap**  /  **Material Design**  /  **W3.CSS**

On the **JavaScript** side you should learn at least one modern framework:  
**React.js**  /  **Angular.js**  /  **Vue.js**   /  **W3.JS**

Maybe the popularity of **jQuery** has passed the top, but it is still the most used JavaScript framework.

# STEP 4 Back-End Roadmaps

## Fullstack

## Fullstack JS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is HTTP?

**HTTP** stands for **H**yper **T**ext **T**ransfer **P**rotocol

**WWW** is about communication between web **clients** and **servers**

Communication between client computers and web servers is done by sending **HTTP Requests** and receiving **HTTP Responses**

## **World Wide Web Communication**

The World Wide Web is about communication between web **clients** and web **servers**.

**Clients** are often browsers (Chrome, Edge, Safari), but they can be any type of program or device.

**Servers** are most often computers in the cloud.

## **HTTP Request / Response**

Communication between clients and servers is done by **requests** and **responses**:

1. A client (a browser) sends an **HTTP request** to the web
2. A web server receives the request
3. The server runs an application to process the request
4. The server returns an **HTTP response** (output) to the browser
5. The client (the browser) receives the response

## **The HTTP Request Circle**

A typical HTTP request / response circle:

1. The browser requests an HTML page. The server returns an HTML file.
2. The browser requests a style sheet. The server returns a CSS file.
3. The browser requests an JPG image. The server returns a JPG file.
4. The browser requests JavaScript code. The server returns a JS file
5. The browser requests data. The server returns data (in XML or JSON).

## **XHR - XML Http Request**

All browsers have a built-in XMLHttpRequest Object (XHR).

XHR is a JavaScript object that is used to transfer data between a web browser and a web server.

XHR is often used to request and receive data for the purpose of modifying a web page.

Despite the XML and Http in the name, XHR is used with other protocols than HTTP, and the data can be of many different types like [HTML](https://www.w3schools.com/whatis/whatis_html.asp), [CSS](https://www.w3schools.com/whatis/whatis_css.asp), [XML](https://www.w3schools.com/whatis/whatis_xml.asp), [JSON](https://www.w3schools.com/whatis/whatis_json.asp), and plain text.

The XHR Object is a Web Developers Dream, because you can:

Update a web page without reloading the page

Request data from a server - after the page has loaded

Send data to a server - in the background

The XHR Object is the underlying concept of [AJAX](https://www.w3schools.com/whatis/whatis_ajax.asp) and [JSON](https://www.w3schools.com/whatis/whatis_json.asp):

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is HTML?

HTML stands for **H**yper **T**ext **M**arkup **L**anguage

HTML is the **standard markup** language for Web pages

HTML **elements** are the building blocks of HTML pages

HTML elements are represented by **<> tags**

## **HTML Elements**

An HTML element is a **start** tag and an **end** tag with content in between:

<h1>This is a Heading</h1>

|  |  |  |
| --- | --- | --- |
| Start tag | Element content | End tag |
| <h1> | This is a Heading | </h1> |
| <p> | This is paragraph. | </p> |

## **HTML Attributes**

* HTML elements can have **attributes**
* Attributes provide **additional information** about the element
* Attributes come in name/value pairs like **charset="utf-8"**

### **Example Explained**

**HTML elements are the building blocks of HTML pages.**

* The <!DOCTYPE html> declaration defines this document to be HTML5
* The <html> element is the root element of an HTML page
* The lang attribute  defines the language of the document
* The <meta> element contains meta information about the document
* The charset attribute defines the character set used in the document
* The <title> element specifies a title for the document
* The <body> element contains the visible page content
* The <h1> element defines a large heading
* The <p> element defines a paragraph

## **HTML Documents**

All HTML documents must start with a document type declaration: <!DOCTYPE html>.

The HTML document itself begins with <html> and ends with </html>.

The visible part of the HTML document is between <body> and </body>.

## **HTML Document Structure**

Below is a visualization of an HTML document (an HTML Page):

<html>

<head>

<title>Page title</title>

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

</body>

</html>

**Note:** Only the content inside the <body> section (the white area above) is displayed in a browser.

## **HTML Headings**

HTML headings are defined with <h1> to <h6> tags.

<h1> defines the most important heading.

<h6> defines the least important heading:

### **Example**

<h1>This is heading 1</h1>  
<h2>This is heading 2</h2>  
<h3>This is heading 3</h3>

## **HTML Paragraphs**

HTML paragraphs are defined with <p> tags:

### **Example**

<p>This is a paragraph.</p>  
<p>This is another paragraph.</p>

## **HTML Links**

HTML links are defined with <a> tags:

### **Example**

<a href="https://www.w3schools.com">This is a link</a>

The link's destination is specified in the href attribute.

## **HTML Images**

HTML images are defined with <img> tags.

The source file (src), alternative text (alt), width, and height are provided as attributes:

### **Example**

<img src="img\_w3schools.jpg" alt="W3Schools" style="width:120px;height:150px"

## **HTML Buttons**

HTML buttons are defined with <button> tags:

**Example**

<button>Click me</button>

## **HTML Lists**

HTML lists are defined with <ul> (unordered/bullet list) or <ol> (ordered/numbered list) tags, followed by <li> tags (list items):

**Example**

<ul>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>

## **HTML Tables**

An HTML table is defined with a <table> tag.

Table rows are defined with <tr> tags.

Table headers are defined with <th> tags. (bold and centered by default).

Table cells (data) are defined with <td> tags.

### **Example**

<table>  
  <tr>  
    <th>Firstname</th>  
    <th>Lastname</th>   
    <th>Age</th>  
  </tr>  
  <tr>  
    <td>Jill</td>  
    <td>Smith</td>   
    <td>50</td>  
  </tr>  
  <tr>  
    <td>Eve</td>  
    <td>Jackson</td>   
    <td>94</td>  
  </tr>  
</table>

## **Programming HTML**

Every HTML element can have attributes.

For web development and programming, the most important attributes are id and class. These attributes are often used to address program based web page manipulations.

|  |  |
| --- | --- |
| **Attribute** | **Example** |
| id | <table **id**="table01"> |
| class | <p **class**="normal"> |
| style | <p **style**="font-size:16px"> |
| data- | <div **data**-id="500"> |
| onclick | <input **onclick**="myFunction()"> |
| onmouseover | <a **onmouseover**="this.setAttribute('style','color:red')"> |
|  |  |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is CSS?

CSS stands for Cascading Style Sheets

CSS describes how HTML elements are to be displayed

### **CSS Example**

<style>  
  
body {background-color:lightblue; text-align:center;}  
h1 {color:blue; font-size:40px;}  
p {font-family:verdana; font-size:20px;}  
  
</style>

## **CSS Syntax**

A CSS rule consists of a selector and a declaration block

The selector points to the HTML element to style (h1).

The declaration block (in curly braces) contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

In the following example all <p> elements will be center-aligned, red and have a font size of 32 pixels:

### **Example**

<style>  
p {font-size:32px; color:red; text-align:center;}  
</style>

**Same example can also be written like this:**

<style>  
p {  
    font-size: 32px;  
    color: red;  
    text-align: center;  
}  
</style>

## **External Style Sheet**

A CSS style sheet can be stored in an external file:

### **mystyle.css**

body {background-color: orange; font-family:verdana}  
h1 {color: white;}  
p {font-size: 20px;}

External style sheets are linked to HTML pages with **<link>** tags:

### **Example**

<!DOCTYPE html>  
<html>  
<link rel="stylesheet" href="mystyle.css">  
<body>  
  
<h1>My First CSS Example</h1>  
<p>This is a paragraph.</p>  
  
</body>  
</html>

## **Inline Style**

### **Example**

<!DOCTYPE html>  
<html>  
<link rel="stylesheet" href="mystyle.css">  
<body>  
  
<h1>My First CSS Example</h1>  
<p>This is a paragraph.</p>  
<p style="font-size:25px">This is a paragraph.</p>  
<p style="font-size:30px">This is a paragraph.</p>  
  
</body>  
</html>

## **Cascading Order**

If different styles are specified for HTML elements, the styles will **cascade** into new styles with the following priority:

* Priority 1: Inline styles
* Priority 2: External and internal style sheets
* Priority 3: Browser default
* If different styles are defined on the same priority level, the last one has the highest priority.

### **Example**

<!DOCTYPE html>  
<html>  
<link rel="stylesheet" href="mystyle.css">  
  
<style>  
body {background-color: lightblue;}  
</style>  
  
<body style="background-color: olivedrab">  
<h1>Multiple Styles Cascades into One</h1>  
<p>Try experimenting by removing styles to see how the cascading stylesheets work.</p>  
<p>Try removing the inline first, then the internal, then the external.</p>  
</body>  
</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is Responsive Web Design?

Responsive Web Design is about using HTML and CSS to automatically resize a website.

Responsive Web Design is about making a website look good on all devices (desktops, tablets, and phones):

## **Setting The Viewport**

When making responsive web pages, add the following <meta> element to all your web pages:

### **Example**

<meta name="viewport" content="width=device-width, initial-scale=1.0">

## **Media Queries**

Media Queries plays an important role in responsive web pages.

With media queries you can define different styles for different browser sizes.

**Example:**  
Resize the browser window to see that the three elements below will display horizontally on large screens and vertically on small screens:

### **Example**

<style>  
.left, .right {  
  float: left;  
  width: 20%; /\* The width is 20%, by default \*/  
}  
  
.main {  
  float: left;  
  width: 60%; /\* The width is 60%, by default \*/  
}  
  
/\* Use Media Query to add a breakpoint at 800px: \*/  
@media screen and (max-width:800px) {  
  .left , .main, .right {width:100%;}  
}  
</style>

## **Responsive Images**

Responsive images are images that scale nicely to fit any browser size.

When the CSS width property is set to a percentage value, an image will scale up and down when resizing the browser window.

### **Example**

<img src="img\_girl.jpg" style="width:80%;height:auto;">

### If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

### **Example**

<img src="img\_girl.jpg" style="max-width:100%;height:auto;">

## **Image Depending on Browser Size**

The HTML <picture> element allows you to define different images for different browser window sizes.

### **Example**

<picture>  
  <source srcset="img\_smallflower.jpg" media="(max-width: 600px)">  
  <source srcset="img\_flowers.jpg" media="(max-width: 1500px)">  
  <source srcset="flowers.jpg">  
  <img src="img\_smallflower.jpg" alt="Flowers">  
</picture>

## **Responsive W3.CSS**

W3.CSS is a free CSS Framework that offers Responsive Design by default.

W3.CSS makes it easy to develop sites that look nice on any device; desktop, laptop, tablet, or phone:

### **Example**

<!DOCTYPE html>  
<html>  
<meta name="viewport" content="width=device-width, initial-scale=1">  
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">  
<body>  
  
<div class="w3-center w3-padding-64 w3-light-grey">  
  <h1>My W3.CSS Page</h1>  
  <p>Resize this page to see the responsive effect!</p>   
</div>  
  
<div class="w3-row-padding">  
  <div class="w3-third">  
    <h2>London</h2>  
    <p>London is the capital city of England.</p>  
    <p>It is the most populous city in the United Kingdom,  
    with a metropolitan area of over 13 million inhabitants.</p>  
  </div>  
  
  <div class="w3-third">  
    <h2>Paris</h2>  
    <p>Paris is the capital of France.</p>   
    <p>The Paris area is one of the largest population centers in Europe,  
    with more than 12 million inhabitants.</p>  
  </div>  
  
  <div class="w3-third">  
    <h2>Tokyo</h2>  
    <p>Tokyo is the capital of Japan.</p>  
    <p>It is the center of the Greater Tokyo Area,  
    and the most populous metropolitan area in the world.</p>  
  </div>  
</div>  
  
</body>  
</html>

## **Bootstrap**

Bootstrap is a very popular framework that uses HTML, CSS and jQuery to make responsive web pages.

### **Example**

<!DOCTYPE html>  
<html lang="en">  
<head>  
<title>Bootstrap Example</title>  
<meta charset="utf-8">  
<meta name="viewport" content="width=device-width, initial-scale=1">  
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">  
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>  
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>  
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>  
</head>  
<body>  
  
<div class="jumbotron text-center">  
  <h1>My First Bootstrap Page</h1>  
  <p>Resize this responsive page to see the effect!</p>   
</div>  
  
<div class="container-fluid">  
  <div class="row">  
    <div class="col-sm-4">  
      <h2>London</h2>  
      <p>London is the capital city of England.</p>  
      <p>It is the most populous city in the United Kingdom,  
      with a metropolitan area of over 13 million inhabitants.</p>  
    </div>  
    <div class="col-sm-4">  
      <h2>Paris</h2>  
      <p>Paris is the capital of France.</p>   
      <p>The Paris area is one of the largest population centers in Europe,  
      with more than 12 million inhabitants.</p>  
    </div>  
    <div class="col-sm-4">  
      <h2>Tokyo</h2>  
      <p>Tokyo is the capital of Japan.</p>  
      <p>It is the center of the Greater Tokyo Area,  
      and the most populous metropolitan area in the world.</p>  
    </div>  
  </div>  
</div>  
  
</body>  
</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is JavaScript?

JavaScript is the **Programming Language** for the Web.

JavaScript can update and change both **HTML** and **CSS.**

JavaScript can **calculate**, **manipulate** and **validate** data.

## **JavaScript Quickstart Tutorial**

This tutorial will take a quick look at the most important JavaScript data types.

JavaScript variables can be:

* Numbers
* Strings
* Objects
* Arrays
* Functions

## **JavaScript Variables**

* JavaScript variables are containers for storing data values.
* In this example, x, y, and z, are variables:
* Example
* var x = 5;  
  var y = 6;  
  var z = x + y;
* From the example above, you can expect:
* x stores the value 5
* y stores the value 6
* z stores the value 11

## **JavaScript Numbers**

JavaScript has only one type of number. Numbers can be written with or without decimals.

### **Example**

var x = 3.14;    // A number with decimals  
var y = 3;       // A number without decimals

All numbers are stored as double precision floating point numbers.

The maximum number of decimals is 17, but floating point is not always 100% accurate:

### **Example**

### var x = 0.2 + 0.1;         // x will be 0.30000000000000004

## **JavaScript Strings**

Strings store text. Strings are written inside quotes. You can use single or double quotes:

### **Example**

var carname = "Volvo XC60";  // Double quotes  
var carname = 'Volvo XC60';  // Single quotes

The length of a string is found in the built in property length:

### **Example**

var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
var sln = txt.length;

## **JavaScript Objects**

You have already learned that JavaScript variables are containers for data values.

This code assigns a **simple value** (Fiat) to a **variable** named car:

var car = "Fiat";

Objects are variables too. But objects can contain many values.

This code assigns many values (Fiat, 500, white) to a variable named car:

var car = {type:"Fiat", model:"500", color:"white"};

## **JavaScript Arrays**

JavaScript arrays are used to store multiple values in a single variable.

### Example

var cars = ["Saab", "Volvo", "BMW"];

## **JavaScript Functions**

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

### **Example**

function myFunction(p1, p2) {  
    return p1 \* p2;              // The function returns the product of p1 and p2  
}

## **What can JavaScript Do?**

This section contains some examples of what JavaScript can do:

* JavaScript Can Change HTML Content
* JavaScript Can Change HTML Attribute Values
* JavaScript Can Change HTML Styles (CSS)
* JavaScript Can Hide HTML Elements
* JavaScript Can Show HTML Elements

## **JavaScript Can Change HTML Content**

One of many JavaScript HTML methods is getElementById().

This example uses the method to "find" an HTML element (with id="demo") and changes the element content (innerHTML) to "Hello JavaScript":

### **Example**

document.getElementById("demo").innerHTML = "Hello JavaScript";

## **JavaScript Can Change HTML Attribute Values**

In this example JavaScript changes the value of the src (source) attribute of an <img> tag:

<!DOCTYPE html>

<html>

<body>

<h2>What Can JavaScript Do?</h2>

<p>JavaScript can change HTML attribute values.</p>

<p>In this case JavaScript changes the value of the src (source) attribute of an image.</p>

<button onclick="document.getElementById('myImage').src='img\_bulbon.gif'">Turn on the light</button>

<img id="myImage" src="img\_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='img\_bulboff.gif'">Turn off the light</button>

</body>

</html>

## **JavaScript Can Change HTML Styles (CSS)**

Changing the style of an HTML element, is a variant of changing an HTML attribute:

### **Example**

document.getElementById("demo").style.fontSize = "35px";  
or  
document.getElementById('demo').style.fontSize = '35px';

## **JavaScript Can Hide HTML Elements**

Hiding HTML elements can be done by changing the display style:

### **Example**

document.getElementById("demo").style.display = "none";  
or  
document.getElementById('demo').style.display = 'none';

## **JavaScript Can Show HTML Elements**

Showing hidden HTML elements can also be done by changing the display style:

### **Example**

document.getElementById("demo").style.display = "block";  
or  
document.getElementById('demo').style.display = 'block';

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is ES5?

ES5 is a shortcut for ECMAScript 5

ECMAScript 5 is also known as JavaScript 5

ECMAScript 5 is also known as ECMAScript 2009

# ECMAScript 5

ECMAScript 5 was released in 2009.

# ECMAScript 5 Features

These were the new features released in 2009:

* The "use strict" Directive
* String.trim()
* Array.isArray()
* Array.forEach()
* Array.map()
* Array.filter()
* Array.reduce()
* Array.reduceRight()
* Array.every()
* Array.some()
* Array.indexOf()
* Array.lastIndexOf()
* JSON.parse()
* JSON.stringify()
* Date.now()
* Property Getters and Setters
* New Object Property Methods

# ECMAScript 5 Syntactical Changes

Property access [ ] on strings

Trailing commas in array and object literals

Multiline string literals

Reserved words as property names

## **Browser Support for ES5 (2009)**

Chrome 23, IE 10, and Safari 6 were the first browsers to fully support ECMAScript 5:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Chrome 23 | IE10 / Edge | Firefox 21 | Safari 6 | Opera 15 |
| Sep 2012 | Sep 2012 | Apr 2013 | Jul 2012 | Jul 2013 |

Internet Explorer 9 (March 2011) supports ES 5 except for "use strict".

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is the HTML DOM?

The HTML DOM is an Object Model for HTML. It defines:

HTML elements as objects

Properties for all HTML elements

Methods for all HTML elements

Events for all HTML elements

The HTML DOM is an API (Programming Interface) for JavaScript:

* JavaScript can add/change/remove HTML elements
* JavaScript can add/change/remove HTML attributes
* JavaScript can add/change/remove CSS styles
* JavaScript can react to HTML events
* JavaScript can add/change/remove HTML events

## **The HTML DOM (Document Object Model)**

* When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
* The **HTML DOM** model is constructed as a tree of **Objects**:

## **Finding HTML Elements**

When you want to access HTML elements with JavaScript, you have to find the elements first.

There are a couple of ways to do this:

* Finding HTML elements by id
* Finding HTML elements by tag name
* Finding HTML elements by class name
* Finding HTML elements by CSS selectors
* Finding HTML elements by HTML object collections

## **Finding HTML Element by Id**

The easiest way to find an HTML element in the DOM, is by using the element id.

* This example finds the element with id="intro":

### **Example**

var myElement = document.getElementById("intro");

If the element is found, the method will return the element as an object (in myElement).

If the element is not found, myElement will contain null.

## **Finding HTML Elements by Tag Name**

This example finds all <p> elements:

### **Example**

var x = document.getElementsByTagName("p");

This example finds the element with id="main", and then finds all <p> elements inside "main":

### **Example**

var x = document.getElementById("main");  
var y = x.getElementsByTagName("p");

## **Finding HTML Elements by Class Name**

If you want to find all HTML elements with the same class name, use getElementsByClassName().

This example returns a list of all elements with class="intro".

### **Example**

var x = document.getElementsByClassName("intro");

Finding elements by class name does not work in Internet Explorer 8 and earlier versions.

## **Finding HTML Elements by CSS Selectors**

If you want to find all HTML elements that matches a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method.

This example returns a list of all <p> elements with class="intro".

### **Example**

var x = document.querySelectorAll("p.intro");

## The querySelectorAll() method does not work in Internet Explorer 8 and earlier versions. Finding HTML Elements by HTML Object Collections

**HTML object collections are also accessible:**

* [document.anchors](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_anchors)
* [document.forms](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_forms)
* [document.images](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_images)
* [document.links](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_links)
* [document.scripts](https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_scripts)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is Google Maps?

Google Maps is a Google API

Google Fonts is a Google API

Google Charts is a Google API

## **My First Google Map**

Start with a simple web page.

Add a <div> element where you want the map to display, and set the size of the map:

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Google Map</h1>  
  
<div id="map" style="width:100%;height:400px;">My map will go here</div>  
  
</body>  
<html>

**Add a JavaScript function to display the map:**

### **Example**

function myMap() {  
    var mapCanvas = document.getElementById("map");  
    var mapOptions = {  
        center: new google.maps.LatLng(51.5, -0.2),  
        zoom: 10  
    };  
    var map = new google.maps.Map(mapCanvas, mapOptions);  
}

The **mapCanvas** variable is the map's HTML element.

The **mapOptions** variable defines the properties for the map.

The **center** property specifies where to center the map (using latitude and longitude coordinates).

The **zoom** property specifies the zoom level for the map (try to experiment with the zoom level).

The **google.maps.Map** object is created with mapCanvas and mapOptions as parameters.

## **Finally Add the Google API**

The functionality of the map is provided by a JavaScript library located at Google:

### **Example**

<scriptsrc="https://maps.googleapis.com/maps/api/js?key=YOUR\_KEY&callback=myMap"></script>

# What is Google Fonts?

Google Maps is a Google API

Google Fonts is a Google API

Google Charts is a Google API

Currently there are 1043 fonts available from Google:

# What is Google Charts?

Google Maps is a Google API

Google Fonts is a Google API

Google Charts is a Google API

## **Google Pie Chart**

Start with a simple basic web page.

Add a <div> element with the id "piechart":

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My Web Page</h1>  
  
<div id="piechart"></div>  
  
</body>  
<html>

**Add a reference to the Chart API at google.com:**

### **Example**

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

**And add a JavaScript function:**

### **Example**

<script type="text/javascript">  
// Load google charts  
google.charts.load('current', {'packages':['corechart']});  
google.charts.setOnLoadCallback(drawChart);  
  
// Draw the chart and set the chart values  
function drawChart() {  
  var data = google.visualization.arrayToDataTable([  
  ['Task', 'Hours per Day'],  
  ['Work', 8],  
  ['Friends', 2],  
  ['Eat', 2],  
  ['TV', 2],  
  ['Gym', 2],  
  ['Sleep', 8]  
]);  
  
  // Optional; add a title and set the width and height of the chart  
  var options = {'title':'My Average Day', 'width':550, 'height':400};  
  
  // Display the chart inside the <div> element with id="piechart"  
  var chart = new google.visualization.PieChart(document.getElementById('piechart'));  
  chart.draw(data, options);  
}  
</script>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is XML?

XML stands for eXtensible Markup Language

XML plays an important role in many different IT systems

XML is often used for distributing data over the Internet

It is important for all web developers to have a good understanding of XML

### **XML Example 1**

<?xml version="1.0" encoding="UTF-8"?>  
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>

### **XML Example 2**

<?xml version="1.0" encoding="UTF-8"?>  
<breakfast\_menu>  
<food>  
    <name>Belgian Waffles</name>  
    <price>$5.95</price>  
    <description>  
   Two of our famous Belgian Waffles with plenty of real maple syrup  
   </description>  
    <calories>650</calories>  
</food>  
<food>  
    <name>Strawberry Belgian Waffles</name>  
    <price>$7.95</price>  
    <description>  
    Light Belgian waffles covered with strawberries and whipped cream  
    </description>  
    <calories>900</calories>  
</food>  
<food>  
    <name>Berry-Berry Belgian Waffles</name>  
    <price>$8.95</price>  
    <description>  
    Belgian waffles covered with assorted fresh berries and whipped cream  
    </description>  
    <calories>900</calories>  
</food>  
<food>  
    <name>French Toast</name>  
    <price>$4.50</price>  
    <description>  
    Thick slices made from our homemade sourdough bread  
    </description>  
    <calories>600</calories>  
</food>  
<food>  
    <name>Homestyle Breakfast</name>  
    <price>$6.95</price>  
    <description>  
    Two eggs, bacon or sausage, toast, and our ever-popular hash browns  
    </description>  
    <calories>950</calories>  
</food>  
</breakfast\_menu>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is AJAX?

AJAX is a developer's dream, because you can:

* Read data from a web server - after a web page has loaded
* Update a web page without reloading the page
* Send data to a web server - in the background

## **AJAX Example Explained**

### **HTML Page**

* <!DOCTYPE html>  
  <html>  
  <body>  
    
  <div id="demo">  
    <h2>Let AJAX change this text</h2>  
    <button type="button" onclick="loadDoc()">Change Content</button>  
  </div>  
    
  </body>  
  </html>

The HTML page contains a <div> section and a <button>.

The <div> section is used to display information from a server.

The <button> calls a function (if it is clicked).

The function requests data from a web server and displays it:

### **Function loadDoc()**

function loadDoc() {  
  var xhttp = new XMLHttpRequest();  
  xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
     document.getElementById("demo").innerHTML = this.responseText;  
    }  
  };  
  xhttp.open("GET", "ajax\_info.txt", true);  
  xhttp.send();  
}

## **What is AJAX?**

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

AJAX is not a programming language.

AJAX just uses a combination of:

* A browser built-in XMLHttpRequest object (to request data from a web server)
* JavaScript and HTML DOM (to display or use the data)

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

## **How AJAX Works**



* 1. An event occurs in a web page (the page is loaded, a button is clicked)
* 2. An XMLHttpRequest object is created by JavaScript
* 3. The XMLHttpRequest object sends a request to a web server
* 4. The server processes the request
* 5. The server sends a response back to the web page
* 6. The response is read by JavaScript
* 7. Proper action (like page update) is performed by JavaScript

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is JSON?

JSON stands for JavaScript Object Notation

JSON is a lightweight format for storing and transporting data

JSON is often used when data is sent from a server to a web page

JSON is "self-describing" and easy to understand

### **JSON Example**

* This example defines an employees object: an array of 3 employee records (objects):
* {  
  "employees":[  
      {"firstName":"John", "lastName":"Doe"},   
      {"firstName":"Anna", "lastName":"Smith"},  
      {"firstName":"Peter", "lastName":"Jones"}  
  ]  
  }

## **JSON Syntax Rules**

* Data is in name/value pairs
* Data is separated by commas
* Curly braces hold objects
* Square brackets hold arrays

## **JavaScript Object Notation**

The JSON format is syntactically identical to the code for creating JavaScript objects.

Because of this similarity, a JavaScript program can easily convert JSON data into native JavaScript objects.

The JSON syntax is derived from JavaScript object notation syntax, but the JSON format is text only. Code for reading and generating JSON data can be written in any programming language.

## **JSON Data - A Name and a Value**

JSON data is written as name/value pairs, just like JavaScript object properties.

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

"firstName":"John"

JSON names require double quotes. JavaScript names do not.

## **JSON Objects**

JSON objects are written inside curly braces.

Just like in JavaScript, objects can contain multiple name/value pairs:

{"firstName":"John", "lastName":"Doe"}

## **JSON Arrays**

JSON arrays are written inside square brackets.

Just like in JavaScript, an array can contain objects:

"employees":[  
    {"firstName":"John", "lastName":"Doe"},   
    {"firstName":"Anna", "lastName":"Smith"},   
    {"firstName":"Peter", "lastName":"Jones"}  
]

In the example above, the object "employees" is an array. It contains three objects.

Each object is a record of a person (with a first name and a last name).

## **Converting a JSON Text to a JavaScript Object**

A common use of JSON is to read data from a web server, and display the data in a web page.

For simplicity, this can be demonstrated using a string as input.

First, create a JavaScript string containing JSON syntax:

var text = '{ "employees" : [' +  
'{ "firstName":"John" , "lastName":"Doe" },' +  
'{ "firstName":"Anna" , "lastName":"Smith" },' +  
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';

Then, use the JavaScript built-in function JSON.parse() to convert the string into a JavaScript object:

var obj = JSON.parse(text);

Finally, use the new JavaScript object in your page:

### **Example**

<p id="demo"></p>  
  
<script>  
document.getElementById("demo").innerHTML =  
obj.employees[1].firstName + " " + obj.employees[1].lastName;  
</script>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is CSS Icons?

Icons come in scalable vector libraries that can be customized with CSS

Common libraries are:

* Font Awesome Icons
* Bootstrap Icons
* Google Icons

## **How To?**

To use icons, just add a link to the icon library the <head> section of your HTML page:

No downloads or installations required!

To insert an icon, add the name of the icon class to any inline HTML element like <i> or <span>.

### **Font Awesome Example**

<!DOCTYPE html>  
<html>  
<head>  
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">  
</head>  
<body>  
  
<i class="fa fa-cloud"></i>  
<i class="fa fa-heart"></i>  
<i class="fa fa-car"></i>  
<i class="fa fa-file"></i>  
<i class="fa fa-bars"></i>  
  
</body>  
</html>

### **Bootstrap Example**

<!DOCTYPE html>  
<html>  
<head>  
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">  
</head>  
<body>  
  
<i class="glyphicon glyphicon-cloud"></i>  
<i class="glyphicon glyphicon-remove"></i>  
<i class="glyphicon glyphicon-user"></i>  
<i class="glyphicon glyphicon-envelope"></i>  
<i class="glyphicon glyphicon-thumbs-up"></i>  
  
</body>  
</html>

### **Google Example**

<!DOCTYPE html>  
<html>  
<head>  
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">  
</head>  
<body>  
  
<i class="material-icons">cloud</i>  
<i class="material-icons">favorite</i>  
<i class="material-icons">attachment</i>  
<i class="material-icons">computer</i>  
<i class="material-icons">traffic</i>  
  
</body>  
</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is Bootstrap?

Bootstrap is the most popular CSS Framework for developing responsive and mobile-first websites.

Bootstrap 4 is the newest version of Bootstrap

### **Example**

<div class="jumbotron text-center">  
  <h1>My First Bootstrap Page</h1>  
  <p>Resize this page to see the responsive effect!</p>   
</div>  
  
<div class="container-fluid">  
  <div class="row">  
    <div class="col-sm-4">  
      <h2>London</h2>  
      <p>London is the capital city of England.</p>  
      <p>It is the most populous city in the United Kingdom,  
      with a metropolitan area of over 13 million inhabitants.</p>  
    </div>  
    <div class="col-sm-4">  
      <h2>Paris</h2>  
      <p>Paris is the capital of France.</p>   
      <p>The Paris area is one of the largest population centers in Europe,  
      with more than 12 million inhabitants.</p>  
    </div>  
    <div class="col-sm-4">  
      <h2>Tokyo</h2>  
      <p>Tokyo is the capital of Japan.</p>  
      <p>It is the center of the Greater Tokyo Area,  
      and the most populous metropolitan area in the world.</p>  
    </div>  
  </div>  
</div>

## **Browser Support**

Bootstrap 4 is the newest version of Bootstrap.

Bootstrap 4 supports all major browsers except Internet Explorer 9.

If you require support for IE9 or IE8, you must use Bootstrap 3.

## **Bootstrap Containers**

The container class is one of the most important Bootstrap classes.

It provides margins, padding, alignments, and more, to HTML elements.

### **Example**

<div class="container">  
    <h1>This is a paragraph</h1>  
    <p>This is a paragraph</p>  
    <p>This is a paragraph</p>  
    <p>This is a paragraph</p>  
    <p>This is a paragraph</p>  
</div>

## **Bootstrap Colors**

### **Example**

<div class="container bg-primary text-white py-4">  
<p>London is the most populous city in the United Kingdom, with a metropolitan area of over 9 million inhabitants.</p>  
</div>  
  
<div class="container bg-success text-white py-4">  
<p>London is the most populous city in the United Kingdom, with a metropolitan area of over 9 million inhabitants.</p>  
</div>

## **Bootstrap Text Colors**

<div class="container">  
    <p class="text-muted">This text is muted.</p>  
    <p class="text-primary">This text is important.</p>  
    <p class="text-success">This text indicates success.</p>  
    <p class="text-info">This text represents some information.</p>  
    <p class="text-warning">This text represents a warning.</p>  
    <p class="text-danger">This text represents danger.</p>  
</div>

## **Bootstrap Columns**

Three equal-width columns, on all devices and screen widths:

### **Example**

<div class="row">  
  <div class="col">.col</div>  
  <div class="col">.col</div>  
  <div class="col">.col</div>  
</div>

## **Responsive Columns**

Three equal-width columns scaling to stack on top of each other on small screens:

### **Example**

<div class="row">  
  <div class="col-sm-4">.col-sm-4</div>  
  <div class="col-sm-4">.col-sm-4</div>  
  <div class="col-sm-4">.col-sm-4</div>  
</div>

## **Bootstrap Tables**

A boredered zebra-striped table:

| **Firstname** | **Lastname** | **Email** |
| --- | --- | --- |
| John | Doe | john@example.com |
| Mary | Moe | mary@example.com |
| July | Dooley | july@example.com |

### **Example**

 <table class="table table-striped table-bordered">  
  <thead>  
    <tr>  
      <th>Firstname</th>  
      <th>Lastname</th>  
      <th>Email</th>  
    </tr>  
  </thead>  
  <tbody>  
    <tr>  
      <td>John</td>  
      <td>Doe</td>  
      <td>john@example.com</td>  
    </tr>  
    <tr>  
      <td>Mary</td>  
      <td>Moe</td>  
      <td>mary@example.com</td>  
    </tr>  
    <tr>  
      <td>July</td>  
      <td>Dooley</td>  
      <td>july@example.com</td>  
    </tr>  
  </tbody>  
</table>

## **Bootstrap Alerts**

Bootstrap provides an easy way to create predefined alert messages:

**Success!**  This alert box indicates a successful or positive action.

**Warning!**  This alert box indicates a warning that might need attention.

**Danger!**  This alert box indicates a dangerous or potentially negative action.

**Primary!**  This alert box indicates an important action.

### **Example**

<div class="alert alert-success">  
  <strong>Success!</strong> Indicates a successful or positive action.  
</div>

## **Bootstrap Buttons**

Bootstrap provides different styles of buttons:

### **Example**

<button type="button" class="btn">Basic</button>  
<button type="button" class="btn btn-primary">Primary</button>  
<button type="button" class="btn btn-secondary">Secondary</button>  
<button type="button" class="btn btn-success">Success</button>  
<button type="button" class="btn btn-info">Info</button>  
<button type="button" class="btn btn-warning">Warning</button>  
<button type="button" class="btn btn-danger">Danger</button>  
<button type="button" class="btn btn-dark">Dark</button>

## **Bootstrap Cards**

### **Example**

<div class="card" style="width:400px">  
  <img src="img\_avatar1.png" alt="Card image">  
  <div class="card-body">

    <h4 class="card-title">John Doe</h4>  
    <p class="card-text">Some example text.</p>  
    <a href="#" class="btn btn-primary">See Profile</a>  
   </div>  
 </div>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is Command Line Interface (CLI)?

C:\>npm install mysoftware

**CLI stands for:**

* Command Line Interface
* Command Line Interpreter
* Command Line Input

## **Command Line Input**

* CLI is a command line program that accepts text input to execute operating system functions.
* In the 1960s, using only computer terminals, this was the only way to interact with computers.
* In the 1970s an 1980s, command line input was commonly used by Unix systems and PC systems like MS-DOS and Apple DOS.
* Today, with graphical user interfaces (GUI), most users never use command-line interfaces (CLI).
* However, CLI is still used by software developers and system administrators to configure computers, install software, and access features that are not available in the graphical interface.

## **Examples**

The software package manager npm uses command line input to install software:

### **Windows Example**

C:\>npm install mysoftware

## **Mac OS Example**

>npm install mysoftware

**You can navigate your folders (directories) with command line commands:**

### **Windows Example**

C:\Users\myuser>cd ..  
C:\Users\>cd ..  
C:\>

Use CLI commands with great attention!!!

Wrong use can easily delete files or destroy your computer system completely.

## **Basic Linux CLI Commands**

|  |  |
| --- | --- |
| **Command** | **Description** |
| ls | List the directory (folder) system. |
| cd pathname | Change directory (folder) in the file system. |
| cd .. | Move one level up (one folder) in the file system. |
| cp | Copy a file to another folder. |
| mv | Move a file to another folder. |
| mkdir | Creates a new directory (folder). |
| rmdir | Remove a directory (folder). |
| clear | Clears the CLI window. |
| exit | Closes the CLI window. |
| man *command* | Shows the manual for a given command. |

## **Basic Windows CLI Commands**

|  |  |
| --- | --- |
| **Command** | **Description** |
| dir | List the directory (folder) system. |
| cd *pathname* | Change directory (folder) in the file system. |
| cd \ | Move to the root folder of the file system. |
| cd .. | Move one level up (one folder) in the file system. |
| copy | Copy a file to another folder. |
| move | Move a file to another folder. |
| type *filename* | Type a file. |
| mkdir or md | Creates a new directory (folder). |
| rmdir or rd | Removes a directory (folder). |
| cls | Clears the CLI window. |
| exit | Closes the CLI window. |
| help *command* | Shows the manual for a given command. |

## **DOS Commands Help**

You can display all available commands with the help command:

### **Example**

C:\Users\myuser>help

|  |  |
| --- | --- |
| ASSOC | Displays or modifies file extension associations. |
| ATTRIB | Displays or changes file attributes. |
| BREAK | Sets or clears extended CTRL+C checking. |
| BCDEDIT | Sets properties in boot database to control boot loading. |
| CACLS | Displays or modifies access control lists (ACLs) of files. |
| CALL | Calls one batch program from another. |
| CD | Displays the name of or changes the current directory. |
| CHCP | Displays or sets the active code page number. |
| CHDIR | Displays the name of or changes the current directory. |
| CHKDSK | Checks a disk and displays a status report. |
| CHKNTFS | Displays or modifies the checking of disk at boot time. |
| CLS | Clears the screen. |
| CMD | Starts a new instance of the Windows command interpreter. |
| COLOR | Sets the default console foreground and background colors. |
| COMP | Compares the contents of two files or sets of files. |
| COMPACT | Displays or alters the compression of files on NTFS partitions. |
| CONVERT | Converts FAT volumes to NTFS. You cannot convert the current drive. |
| COPY | Copies one or more files to another location. |
| DATE | Displays or sets the date. |
| DEL | Deletes one or more files. |
| DIR | Displays a list of files and subdirectories in a directory. |
| DISKPART | Displays or configures Disk Partition properties. |
| DOSKEY | Edits command lines, recalls Windows commands, and creates macros. |
| DRIVERQUERY | Displays current device driver status and properties. |
| ECHO | Displays messages, or turns command echoing on or off. |
| ENDLOCAL | Ends localization of environment changes in a batch file. |
| ERASE | Deletes one or more files. |
| EXIT | Quits the CMD.EXE program (command interpreter). |
| FC | Compares two files or sets of files, and displays the differences between them. |
| FIND | Searches for a text string in a file or files. |
| FINDSTR | Searches for strings in files. |
| FOR | Runs a specified command for each file in a set of files. |
| FORMAT | Formats a disk for use with Windows. |
| FSUTIL | Displays or configures the file system properties. |
| FTYPE | Displays or modifies file types used in file extension associations. |
| GOTO | Directs the Windows command interpreter to a labeled line in a batch program. |
| GPRESULT | Displays Group Policy information for machine or user. |
| GRAFTABL | Enables Windows to display an extended character set in graphics mode. |
| HELP | Provides Help information for Windows commands. |
| ICACLS | Display, modify, backup, or restore ACLs for files and directories. |
| IF | Performs conditional processing in batch programs. |
| LABEL | Creates, changes, or deletes the volume label of a disk. |
| MD | Creates a directory. |
| MKDIR | Creates a directory. |
| MKLINK | Creates Symbolic Links and Hard Links. |
| MODE | Configures a system device. |
| MORE | Displays output one screen at a time. |
| MOVE | Moves one or more files from one directory to another directory. |
| OPENFILES | Displays files opened by remote users for a file share. |
| PATH | Displays or sets a search path for executable files. |
| PAUSE | Suspends processing of a batch file and displays a message. |
| POPD | Restores the previous value of the current directory saved by PUSHD. |
| PRINT | Prints a text file. |
| PROMPT | Changes the Windows command prompt. |
| PUSHD | Saves the current directory then changes it. |
| RD | Removes a directory. |
| RECOVER | Recovers readable information from a bad or defective disk. |
| REM | Records comments (remarks) in batch files or CONFIG.SYS. |
| REN | Renames a file or files. |
| RENAME | Renames a file or files. |
| REPLACE | Replaces files. |
| RMDIR | Removes a directory. |
| ROBOCOPY | Advanced utility to copy files and directory trees. |
| SET | Displays, sets, or removes Windows environment variables. |
| SETLOCAL | Begins localization of environment changes in a batch file. |
| SC | Displays or configures services (background processes). |
| SCHTASKS | Schedules commands and programs to run on a computer. |
| SHIFT | Shifts the position of replaceable parameters in batch files. |
| SHUTDOWN | Allows proper local or remote shutdown of machine. |
| SORT | Sorts input. |
| START | Starts a separate window to run a specified program or command. |
| SUBST | Associates a path with a drive letter. |
| SYSTEMINFO | Displays machine specific properties and configuration. |
| TASKLIST | Displays all currently running tasks including services. |
| TASKKILL | Kill or stop a running process or application. |
| TIME | Displays or sets the system time. |
| TITLE | Sets the window title for a CMD.EXE session. |
| TREE | Graphically displays the directory structure of a drive or path. |
| TYPE | Displays the contents of a text file. |
| VER | Displays the Windows version. |
| VERIFY | Tells Windows whether to verify that your files are written correctly to a disk. |
| VOL | Displays a disk volume label and serial number. |
| XCOPY | Copies files and directory trees. |
| WMIC | Displays WMI information inside interactive command shell. |

## **Command Help**

For more information on a specific command, type help + command-name

### **Example**

C:\Users\myuser>help date

Displays or sets the date.  
  
DATE [/T | date]  
  
Type DATE without parameters to display the current date setting and  
a prompt for a new one. Press ENTER to keep the same date.  
  
If Command Extensions are enabled the DATE command supports  
the /T switch which tells the command to just output the  
current date, without prompting for a new date.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is npm?

npm is the world's largest Software Library (Registry)

npm is also a software Package Manager and Installer

## **The World's Largest Software Registry (Library)**

npm is the world's largest Software Registry.

The registry contains over 800,000 code packages.

Open-source developers use npm to share software.

Many organizations also use npm to manage private development.

## **Using npm is Free**

npm is free to use.

You can download all npm public software packages without any registration or logon.

## **Command Line Client**

npm includes a CLI (Command Line Client) that can be used to download and install software:

### **Windows Example**

C:\>npm install <package>

### **Mac OS Example**

>npm install <package>

## **Installing npm**

npm is installed with Node.js

This means that you have to install Node.js to get npm installed on your computer.

Download Node.js from the official Node.js web site: [https://nodejs.org](https://nodejs.org/)

## **Software Package Manager**

The name npm (Node Package Manager) stems from when npm first was created as a package manager for Node.js.

All npm packages are defined in files called package.json.

The content of package.json must be written in JSON.

At least two fields must be present in the definition file: name and version.

### **Example**

{  
"name" : "foo",  
"version" : "1.2.3",  
"description" : "A package for fooing things",  
"main" : "foo.js",  
"keywords" : ["foo", "fool", "foolish"],  
"author" : "John Doe",  
"licence" : "ISC"  
}

## **Managing Dependencies**

npm can manage dependencies.

npm can (in one command line) install all the dependencies of a project.

Dependencies are also defined in package.json.

## **Sharing Your Software**

If you want to share your own software in the npm registry, you can sign in at:

<https://www.npmjs.com>

## **Publishing a Package**

You can publish any directory from your computer as long as the directory has a package.json file.

Check if npm is installed:

C:\>npm

Check if you are logged in:

C:\>npm whoami

If not, log in:

C:\>npm login  
Username: <your username>  
Password: <your password>

Navigate to your project and publish your project:

C:\Users\myuser>cd myproject  
C:\Users\myuser\myproject>npm publish

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is GitHub?



GitHub is a code hosting platform for collaboration and version control.

GitHub lets you (and others) work together on projects.

GitHub essentials are:

* Repositories
* Branches
* Commits
* Pull Requests
* Git (the version control software GitHub is built on)

### **Example**

$ git push origin heroku  
$ cd /etc/  
$ ls

## **Repository**

A GitHub repository can be used to store a development project.

It can contain folders and any type of **files** (HTML, CSS, JavaScript, Documents, Data, Images).

A GitHub repository should also include a **licence** file and a **README** file about the project.

A GitHub repository can also be used to store ideas, or any resources that you want to share.

## **Branch**

A GitHub branch is used to work with different **versions** of a repository at the same time.

By default a repository has a **master** branch (a production branch).

Any other branch is a **copy** of the master branch (as it was at a point in time).

New Branches are for bug fixes and feature work separate from the master branch. When changes are ready, they can be merged into the master branch. If you make changes to the master branch while working on a new branch, these updates can be pulled in.

## **Commits**

At GitHub, changes are called commits.

Each commit (change) has a description explaining why a change was made.

## **Pull Requests**

Pull Requests are the heart of GitHub **collaboration**.

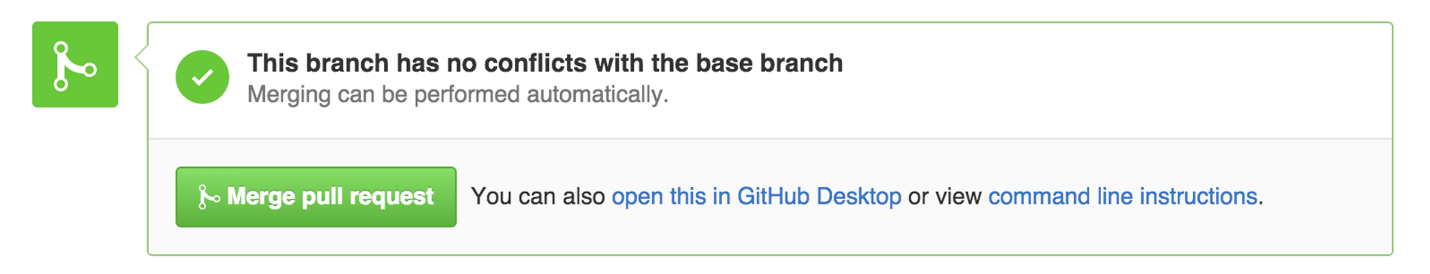
With a pull request you are **proposing** that your changes should be **merged** (pulled in) with the master.

Pull requests show content **differences**, changes, additions, and subtractions in **colors** (green and red).

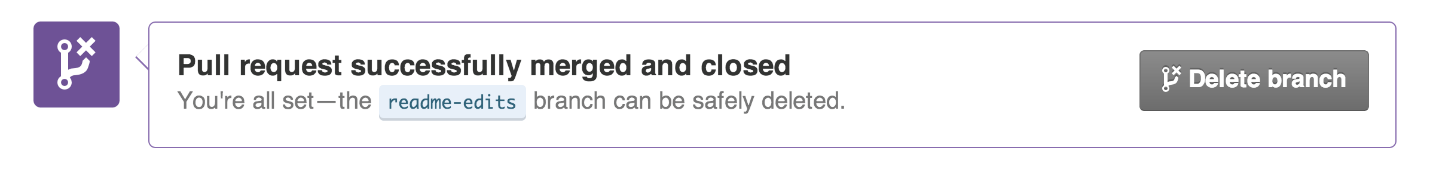
As soon as you have a commit, you can open a pull request and start a discussion, even before the code is finished.

A a great way to learn GitHub, before working on larger projects, is to open pull requests in your own repository and merge them yourself.

**You merge any changes into the master by clicking a "Merge pull request" button.**



**After merging you can delete the branch by clicking a "Delete branch button".**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is jQuery?

jQuery is a JavaScript library.

jQuery greatly simplifies JavaScript programming.

jQuery is easy to learn.

## **jQuery Hide and Show**

With jQuery, you can hide and show HTML elements with the hide() and show() methods:

### **Example**

$("#hide").click(function(){  
    $("p").hide();  
});  
  
$("#show").click(function(){  
    $("p").show();  
});

## **jQuery Sliding**

With jQuery you can create sliding effects on HTML elements:

* slideDown()
* slideUp()
* slideToggle()

### **Example**

$("#flip").click(function(){  
    $("#panel").slideDown();  
});

## **jQuery Animation**

### **Example**

$("button").click(function(){  
    $("div").animate({  
        left: '250px',  
        height: '+=150px',  
        width: '+=150px'  
    });  
});

## **jQuery Manipulating CSS**

### **Example**

$("button").click(function(){  
    $("h1, h2, p").toggleClass("blue");  
});

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is AngularJS?

AngularJS lets you **extend** HTML with HTML attributes called **directives**

AngularJS directives offers **functionality** to HTML applications

AngularJS provides **built-in** directives and **user defined** directives

## **AngularJS Directives**

AngularJS uses double braces {{ }} as place holders for data.

AngularJS directives are HTML attributes with the prefix ng-

The ng-app directive initializes an AngularJS application.

The ng-init directive initializes application data.

### **Example**

<div ng-app="" ng-init="message='Hello AngularJS!'">  
  <h1>{{ message }}</h1>  
</div>

## **The ng-model Directive**

The ng-model directive binds the value of HTML elements to application data.

### **Example**

<div ng-app="" ng-init="firstName='John'">  
  
<p>Name: <input type="text" ng-model="firstName"></p>  
<p>You wrote: <b>{{firstName}}</b></p>  
  
</div>

## **The ng-bind Directive**

The ng-bind directive binds data to a model.

### **Example**

<div ng-app="" ng-init="firstName='John'">  
  
<p>Name: <input type="text" ng-model="firstName"></p>  
<p>You wrote <b ng-bind="firstName"></b></p>  
  
</div>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is React?

React is a JavaScript library created by Facebook

React is a User Interface (UI) library

React is a tool for building UI components

## **React Quickstart Tutorial**

This is a quickstart tutorial.

Before you start, you should have a basic understanding of:

* What is [HTML](https://www.w3schools.com/whatis/whatis_html.asp)
* What is [CSS](https://www.w3schools.com/whatis/whatis_css.asp)
* What is [DOM](https://www.w3schools.com/whatis/whatis_htmldom.asp)
* What is [ES6](https://www.w3schools.com/js/js_es6.asp)
* What is [Node.js](https://www.w3schools.com/nodejs/default.asp)
* What is [npm](https://www.w3schools.com/whatis/whatis_npm.asp)

## **Adding React to an HTML Page**

This quickstart tutorial will add React to a page like this:

### **Example**

<!DOCTYPE html>  
<html lang="en">  
<title>Test React</title>  
  
<!-- Load React API -->  
<script src= "https://unpkg.com/react@16/umd/react.production.min.js"></script>  
<!-- Load React DOM-->  
<script src= "https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>  
<!-- Load Babel Compiler -->  
<script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>  
  
<body>  
  
<script type="text/babel">  
    //  JSX Babel code goes here  
</script>  
  
</body>  
</html>

## **What is Babel?**

Babel is a JavaScript compiler that can translate markup or programming languages into JavaScript.

With Babel, you can use the newest features of JavaScript (ES6 - ECMAScript 2015).

Babel is available for different conversions. React uses Babel to convert JSX into JavaScript.

Please note that <script type="text/babel"> is needed for using Babel.

**What is JSX?**

JSX stands for JavaScript XML.

JSX is an XML/HTML like extension to JavaScript.

### **Example**

const element = <h1>Hello World!</h1>

As you can see above, JSX is not JavaScript nor HTML.

JSX is a XML syntax extension to JavaScript that also comes with the full power of ES6 (ECMAScript 2015).

Just like HTML, JSX tags can have a tag names, attributes, and children. If an attribute is wrapped in curly braces, the value is a JavaScript expression.

Note that JSX does not use quotes around the HTML text string.

## **React DOM Render**

The method ReactDom.render() is used to render (display) HTML elements:

### **Example**

<div id="id01">Hello World!</div>  
  
<script type="text/babel">  
ReactDOM.render(  
    <h1>Hello React!</h1>,  
    document.getElementById('id01'));  
</script>

## **JSX Expressions**

Expressions can be used in JSX by wrapping them in curly {} braces.

### **Example**

<div id="id01">Hello World!</div>  
  
<script type="text/babel">  
const name = 'John Doe';  
ReactDOM.render(  
    <h1>Hello {name}!</h1>,  
    document.getElementById('id01'));  
</script>

## **React Elements**

React applications are usually built around a single HTML element.

React developers often call this the root node (root element):

<div id="root"></div>

React elements look like this:

const element = <h1>Hello React!</h1>

Elements are rendered (displayed) with the ReactDOM.render() method:

ReactDOM.render(element, document.getElementById('root'));

React elements are immutable. They cannot be changed.

The only way to change a React element is to render a new element every time:

### **Example**

function tick() {  
    const element = (<h1>{new Date().toLocaleTimeString()}</h1>);  
    ReactDOM.render(element, document.getElementById('root'));  
}  
setInterval(tick, 1000);

## **React Components**

React components are JavaScript functions.

This example creates a React **component** named "Welcome":

### **Example**

function Welcome() {  
    return <h1>Hello React!</h1>;  
}  
ReactDOM.render(<Welcome />, document.getElementById('root'));

React can also use ES6 classes to create components.

This example creates a React component named Welcome with a render **method**:

### **Example**

class Welcome extends React.Component {   
    render() { return(<h1>Hello React!</h1>); }  
}  
ReactDOM.render(<Welcome />, document.getElementById('root'));

## **React Component Properties**

This example creates a React **component** named "Welcome" with property arguments:

### **Example**

function Welcome(props) {  
    return <h1>Hello {props.name}!</h1>;  
}  
ReactDOM.render(<Welcome name="John Doe"/>, document.getElementById('root'));

React can also use ES6 classes to create components.

This example also creates a React component named "Welcome" with property arguments:

### **Example**

class Welcome extends React.Component {   
    render() { return(<h1>Hello {this.props.name}</h1>); }  
}  
ReactDOM.render(<Welcome name="John Doe"/>, document.getElementById('root'));

## **JSX Compiler**

The examples on this page compiles JSX in the browser.

For production code, the compilation should be done separately.

## **Create React Application**

Facebook has created a Create React Application with everything you need to build a React app.

It is a a development server that uses Webpack to compile React, JSX, and ES6, auto-prefix CSS files.

The Create React App uses ESLint to test and warn about mistakes in the code.

To create a Create React App run the following code on your terminal:

### **Example**

npx create-react-app react-tutorial

Make sure you have Node.js 5.2 or higher. Otherwise you must install npx:

### **Example**

npm i npx

Start one folder up from where you want your application to stay:

### **Example**

C:\Users\myUser>npx create-react-app react-tutorial

### **Success Result:**

npx: installed 63 in 10.359s  
Creating a new React app in C:\Users\myUser\react-tutorial.  
Installing packages. This might take a couple of minutes.  
Installing react, react-dom, and react-scripts...  
+ react-dom@16.5.2  
+ react@16.5.2  
+ react-scripts@2.0.4  
added 1732 packages from 664 contributors and audited 31900 packages in 355.501s  
found 0 vulnerabilities+ react@16.5.2  
  
Success! Created react-tutorial at C:\Users\myUser\react-tutorial  
Inside that directory, you can run several commands:  
  
npm start  
Starts the development server.  
  
npm run build  
Bundles the app into static files for production.  
  
npm test  
Starts the test runner.  
  
npm run eject  
Removes this tool and copies build dependencies, configuration files  
and scripts into the app directory. If you do this, you can't go back!  
  
We suggest that you begin by typing:  
  
cd react-tutorial  
npm start

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is Vue.js?

Vue.js lets you extend HTML with HTML attributes called directives

Vue.js directives offers functionality to HTML applications

Vue.js provides built-in directives and user defined directives

## **Vue.js Directives**

Vue.js uses double braces {{ }} as place-holders for data.

Vue.js directives are HTML attributes with the prefix v-

## **Vue Example**

In the example below, a new Vue object is created with new Vue().

The property el: binds the new Vue object to the HTML element with **id="app"**.

### **Example**

<div id="app">  
<h1>{{ message }}</h1>  
</div>  
  
<script>

var myObject = new Vue({  
    el: '#app',  
    data: {message: 'Hello Vue!'}  
})

</script>

## **Vue.js Binding**

When a Vue object is bound to an HTML element, the HTML element will change when the Vue object changes:

### **Example**

<div id="app">  
{{ message }}  
</div>  
  
<script>

var myObject = new Vue({  
    el: '#app',  
    data: {message: 'Hello Vue!'}  
})  
  
function myFunction() {  
    myObject.message = "John Doe";  
}

</script>

## **Vue.js Two-Way Binding**

The v-model directive binds the value of HTML elements to application data.

This is called two-way binding:

### **Example**

<div id="app">  
  <p>{{ message }}</p>  
  <p><input v-model="message"></p>  
</div>  
  
<script>

var myObject = new Vue({  
    el: '#app',  
    data: {message: 'Hello Vue!'}  
})

</script>

## **Vue.js Loop Binding**

Using the v-for directive to bind an array of Vue objects to an "array" of HTML element:

### **Example**

<div id="app">  
 <ul>  
   <li v-for="x in todos">  
   {{ x.text }}  
   </li>  
  </ul>  
</div>  
  
<script>

myObject = new Vue({  
    el: '#app',  
    data: {  
    todos: [  
        { text: 'Learn JavaScript' },  
        { text: 'Learn Vue.js' },  
        { text: 'Build Something Awesome' }  
        ]  
    }  
})  
</script>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is Full Stack?

## **Full Stack Web Developer**

A full stack web developer is a person who can develop both client and server software.

In addition to mastering HTML and CSS, he/she also knows how to:

* Program a browser (like using JavaScript, jQuery, Angular, or Vue)
* Program a server (like using PHP, ASP, Python, or Node)
* Program a database (like using SQL, SQLite, or MongoDB)

## **Client Software (Front End)**

* [HTML](https://www.w3schools.com/whatis/whatis_html.asp)
* [CSS](https://www.w3schools.com/whatis/whatis_css.asp)
* [Bootstrap](https://www.w3schools.com/whatis/whatis_bootstrap.asp)
* [W3.CSS](https://www.w3schools.com/whatis/whatis_w3css.asp)
* [JavaScript](https://www.w3schools.com/whatis/whatis_js.asp)
* [ES5](https://www.w3schools.com/whatis/whatis_es5.asp)
* [HTML DOM](https://www.w3schools.com/whatis/whatis_htmldom.asp)
* [JSON](https://www.w3schools.com/whatis/whatis_json.asp)
* [XML](https://www.w3schools.com/whatis/whatis_xml.asp)
* [jQuery](https://www.w3schools.com/jquery/default.asp)
* [Angular](https://www.w3schools.com/whatis/whatis_angularjs.asp)
* [React](https://www.w3schools.com/whatis/whatis_react.asp)
* Backbone.js
* Ember.js
* Redux
* [Storybook](https://storybook.js.org/basics/quick-start-guide/)
* GraphQL
* Meteor.js
* Grunt
* Gulp

## **Server Software (Back End)**

* [PHP](https://www.w3schools.com/php/default.asp)
* [ASP](https://www.w3schools.com/asp/default.asp)
* [C++](https://www.w3schools.com/cpp/default.asp)
* [C#](https://www.w3schools.com/cs/default.asp)
* [Java](https://www.w3schools.com/java/default.asp)
* [Python](https://www.w3schools.com/python/default.asp)
* [Node.js](https://www.w3schools.com/nodejs/default.asp)
* Express.js
* Ruby
* REST
* GO
* [SQL](https://www.w3schools.com/whatis/whatis_sql.asp)
* [MongoDB](https://www.w3schools.com/nodejs/nodejs_mongodb.asp)
* Firebase.com
* Sass
* Less
* Parse.com
* PaaS (Azure and Heroku)

## **Popular Stacks**

* LAMP stack: JavaScript - Linux - Apache - MySQL - PHP
* LEMP stack: JavaScript - Linux - Nginx - MySQL - PHP
* MEAN stack: JavaScript - MongoDB - Express - AngularJS - Node.js
* Django stack: JavaScript - Python - Django - MySQL
* Ruby on Rails: JavaScript - Ruby - SQLite - Rails

## **Advantages**

**The advantage of being a full stack web developer is:**

* You can master all the techniques involved in a development project
* You can make a prototype very rapidly
* You can provide help to all the team members
* You can reduce the cost of the project
* You can reduce the time used for team communication
* You can switch between front and back end development based on requirements
* You can better understand all aspects of new and upcoming technologies

## **Disadvantages**

* The solution chosen can be wrong for the project
* The solution chosen can be dependent on developer skills
* The solution can generate a key person risk
* Being a full stack developer is increasingly complex

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# What is SQL?

SQL stands for Structured Query Language

SQL is a standard language for accessing databases

SQL has been an international standard (ISO) since 1987

**SQL Statements**

To access a database, you use SQL statements.

The following SQL statement selects all records in a database table called "Customers":

### **Example**

SELECT \* FROM Customers;

## **Database Tables**

A database most often contains one or more tables.

Each table is identified by a name like "Customers" or "Orders".

Below is a selection from a "Customers" table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | CustomerName | ContactName | Address | City | PostalCode | Country |
| 1 | Alfreds Futterkiste | Maria Anders | Obere Str. 57 | Berlin | 12209 | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Avda. de la Constitución 2222 | México D.F. | 05021 | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mataderos 2312 | México D.F. | 05023 | Mexico |
| 4 | Around the Horn | Thomas Hardy | 120 Hanover Sq. | London | WA1 1DP | UK |
| 5 | Berglunds snabbköp | Christina Berglund | Berguvsvägen 8 | Luleå | S-958 22 | Sweden |

**The table above contains five records (one for each customer) and seven columns:**

1. CustomerID (ID)
2. CustomerName
3. ContactName
4. Address
5. City
6. PostalCode
7. Country

## **The Most Important SQL Statements:**

* **SELECT** - extracts data from a database
* **UPDATE** - updates data in a database
* **DELETE** - deletes data from a database
* **INSERT INTO** - inserts new data into a database
* **CREATE DATABASE** - creates a new database
* **ALTER DATABASE** - modifies a database
* **CREATE TABLE** - creates a new table
* **ALTER TABLE** - modifies a table
* **DROP TABLE** - deletes a table
* **CREATE INDEX** - creates an index (search key)
* **DROP INDEX** - deletes an index

SQL keywords are NOT case sensitive: select is the same as SELECT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*