STATS 401 Lab 3 Tutorial Web Development: Introduction

Since we are learning about data-scraping in Lab 4, we need to be familiar with HTML files. We will also be learning about web development later from week 3 and how to make functioning websites from scratch. Therefore, please pay additional attention to this lab because it will be the basics for most of the following labs. Here are the parts of web development that we'll know.

First, we'll start with HTML. HTML is the backbone of every web page you have ever visited. It contains the structure and content of a webpage. Next, we'll learn how to link CSS to our HTML pages. CSS defines the styling of a web page. It makes things look good, so we won't have to look at Times New Roman until the sun burns out. Then we'll look at Javascript. If you have programmed before, this section will be very familiar. Javascript enables web developers to add all types of functionality to their websites. HTML, CSS, and Javascript are the holy trinity of 'Front-End Development.

We don't require heavyweight development tools to build and create things for web development. We only need something called a **Text Editor**. Text Editors simply make files containing...text. This means you could technically create HTML, CSS, and js files using just notepad on mac or windows. The difference is that dedicated Coding Text Editors have many shortcuts, auto-completion of code, and other useful and time-saving functions. Here's what we recommend downloading and installing for this course: **(You only need to download one of these)**

1.) Sublime Text 3

+Highly useful for web dev (Many Shortcuts, autocomplete, etc).
+High skill ceiling (The better you get, the more you can do).
+Lightweight
+Looks pretty good
+Easy interface, very approachable.
+It's free to download and install, don't worry about the license popup.

2.) Atom

+Many Many advanced features.
+High skill ceiling (The better you get, the more you can do with it).

+Looks good
+Free

-Maybe daunting at first.
-Many Many Features (May bog down your focus, intimidating?)

3.) Notepad++

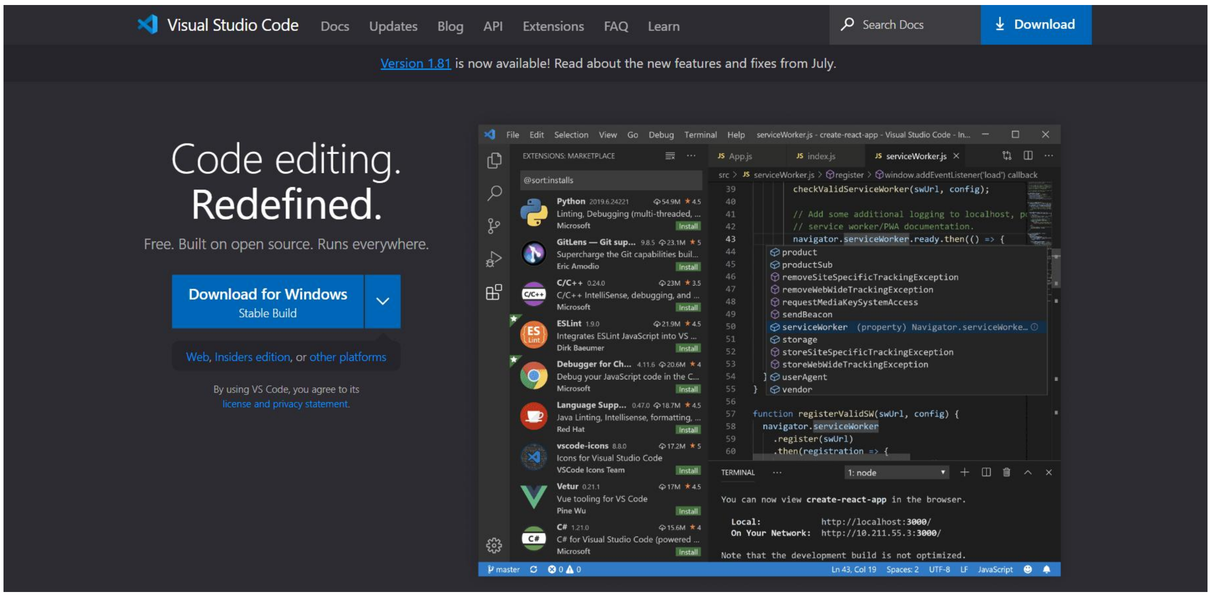
+ Tried and True, supported for a long time
+ Easy interface, very approachable.
+ Lightweight

-Boring looking
-Lacking many of the features of the others

4.) VS Code -- This will be what instructional booklets use.

Task0: Set up

Download VS Code from the official website [Visual Studio Code - Code Editing. Redefined](). (you can also use other editors like Atom if you want)



Version 1.81 is now available! Read about the new features and fixes from July.

Code editing. Redefined.

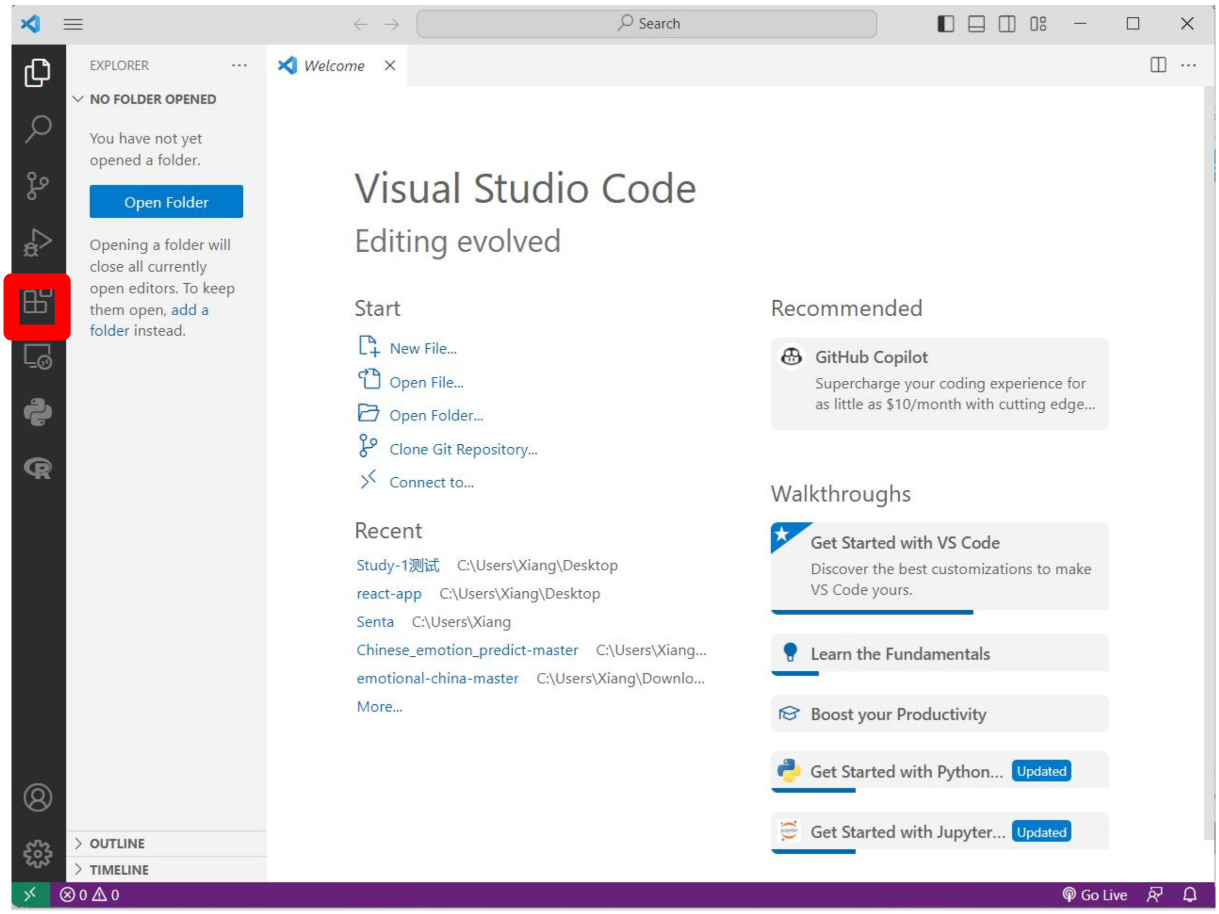
Free. Built on open source. Runs everywhere.

```
checkValidServiceWorker(swUrl, config);

// Add some additional logging to localhost, p
// service worker/PWA documentation.
navigator.serviceWorker.ready.then(() => {
```

```
function registerValidSW(swUrl, config) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
```

```
You can now view create-react-app in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://10.211.55.3:3000/

Note that the development build is not optimized.
```

Then the starting page will look like the below; click on "Extensions" and be ready to install some initial packages we need.



You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

Visual Studio Code

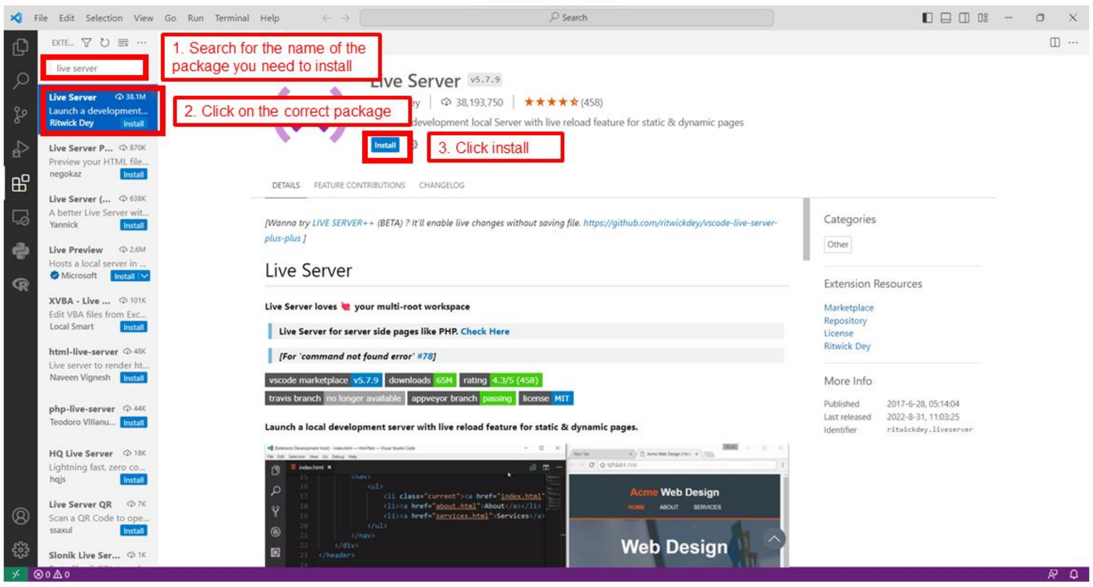
Editing evolved

Study-1测试	C:\Users\Xiang\Desktop
react-app	C:\Users\Xiang\Desktop
Senta	C:\Users\Xiang
Chinese_emotion_predict-master	C:\Users\Xiang...
emotional-china-master	C:\Users\Xiang\Downlo...

Supercharge your coding experience for as little as $10/month with cutting edge...

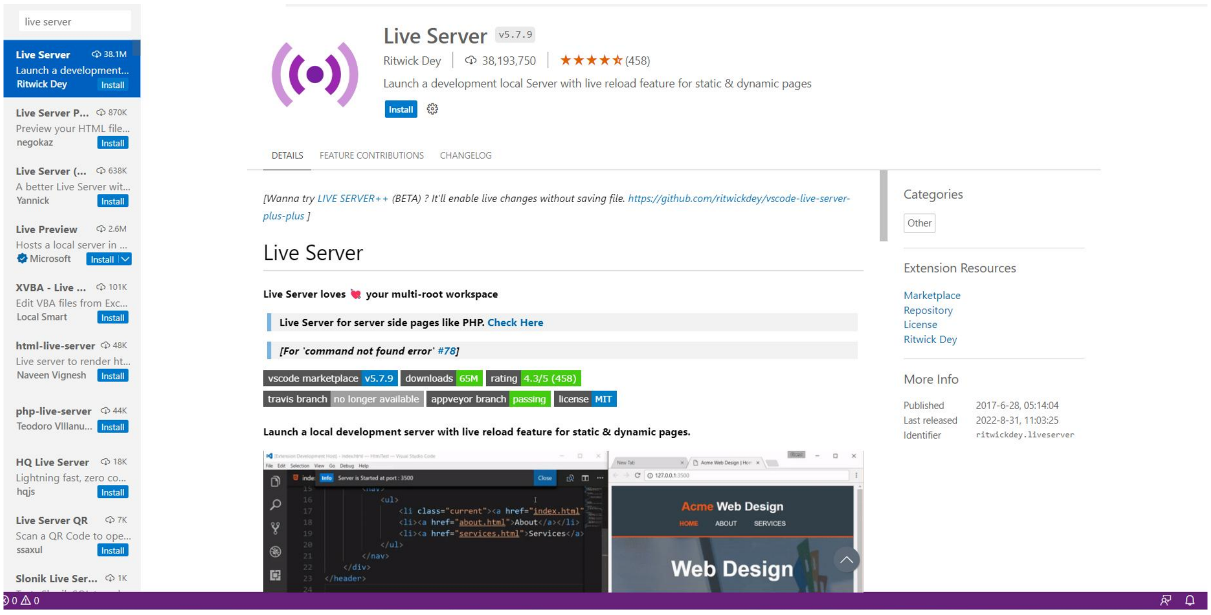
Discover the best customizations to make VS Code yours.

Note that you do not necessarily need these extensions to do your work, but they do make your working process more efficient and your work space prettier. You can try to explore more extensions by yourself.

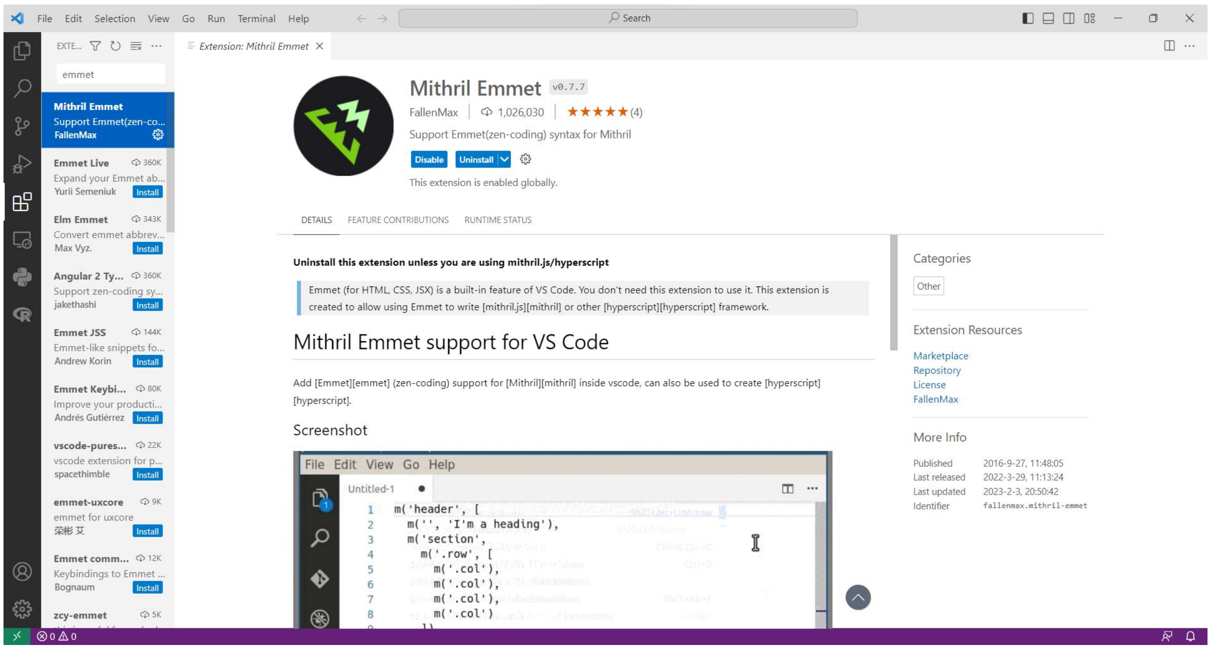

1. Search for the name of the package you need to install
2. Click on the correct package
3. Click install
live server
Live Server
Install

Here are some basic and highly recommended package for you:

1. install live-server



2. install
emmet



Mithril Emmet — Support Emmet(zen-co... — FallenMax
Emmet Live 360K — Expand your Emmet ab... — Yurii Semeniuk
Elm Emmet 343K — Convert emmet abbrev... — Max Vyz.
Angular 2 Ty... 360K — Support zen-coding sy... — jakethashi
Emmet JSS 144K — Emmet-like snippets fo... — Andrew Korin
Emmet Keybi... 80K — Improve your producti... — Andrés Gutiérrez
vscode-pures... 22K — vscode extension for p... — spacethimble
emmet-uxcore 9K — emmet for uxcore — 荣彬 艾
Emmet comm... 12K — Keybindings to Emmet ... — Bognaum
zcy-emmet 5K

Mithril Emmet v0.7.7
FallenMax | 1,026,030 | ★★★★★ (4)
Support Emmet(zen-coding) syntax for Mithril
This extension is enabled globally.

Uninstall this extension unless you are using mithril.js/hyperscript

> Emmet (for HTML, CSS, JSX) is a built-in feature of VS Code. You don't need this extension to use it. This extension is created to allow using Emmet to write [mithril.js][mithril] or other [hyperscript][hyperscript] framework.

Mithril Emmet support for VS Code

Add [Emmet][emmet] (zen-coding) support for [Mithril][mithril] inside vscode, can also be used to create [hyperscript][hyperscript].

Screenshot

```
m('header', [
  m('', 'I'm a heading'),
  m('section',
    m('.row', [
      m('.col'),
      m('.col'),
      m('.col'),
      m('.col')
```

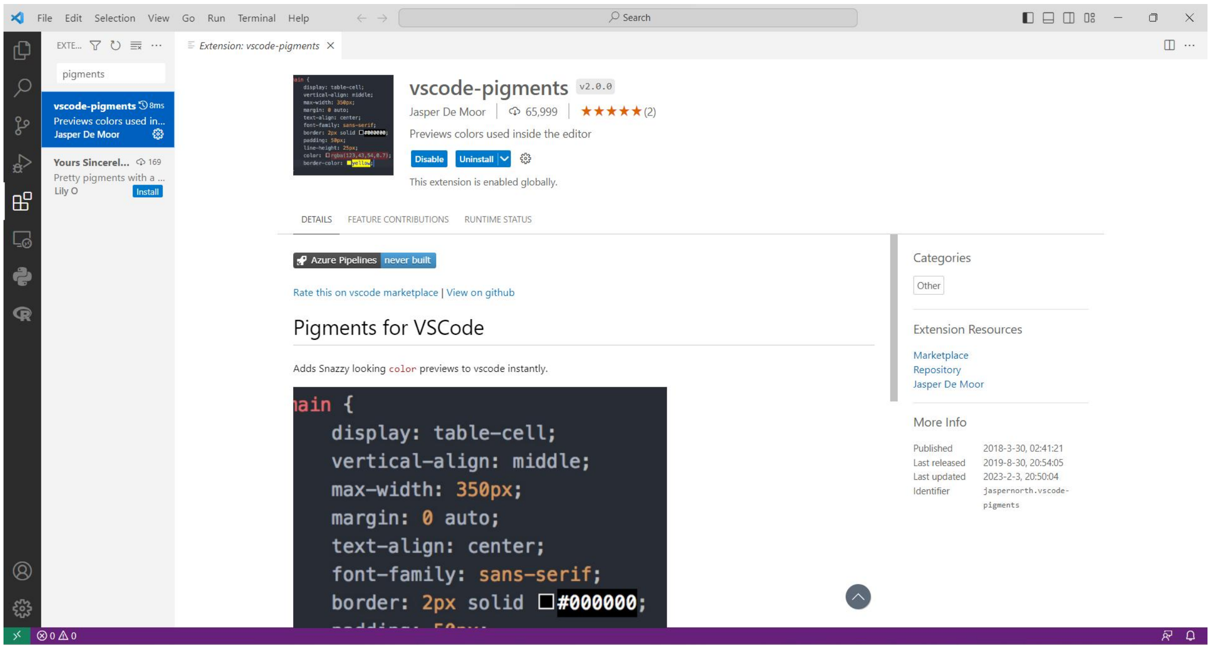
Categories
Other

Extension Resources
Marketplace
Repository
License
FallenMax

More Info

Published	2016-9-27, 11:48:05
Last released	2022-3-29, 11:13:24
Last updated	2023-2-3, 20:50:42
Identifier	fallenmax.mithril-emmet

3. install
pigments


File Edit Selection View Go Run Terminal Help
Search
EXTE...
pigments
Extension: vscode-pigments
Disable
Uninstall
DETAILS
FEATURE CONTRIBUTIONS
RUNTIME STATUS
Install
Azure Pipelines never built

vscode-pigments 8ms — Previews colors used in... — Jasper De Moor
Yours Sincerel... 169 — Pretty pigments with a ... — Lily O

vscode-pigments v2.0.0
Jasper De Moor | 65,999 | ★★★★★ (2)
Previews colors used inside the editor
This extension is enabled globally.

Rate this on vscode marketplace | View on github

Pigments for VSCode

Adds Snazzy looking `color` previews to vscode instantly.

```
ain {
  display: table-cell;
  vertical-align: middle;
  max-width: 350px;
  margin: 0 auto;
  text-align: center;
  font-family: sans-serif;
  border: 2px solid ■#000000;
```

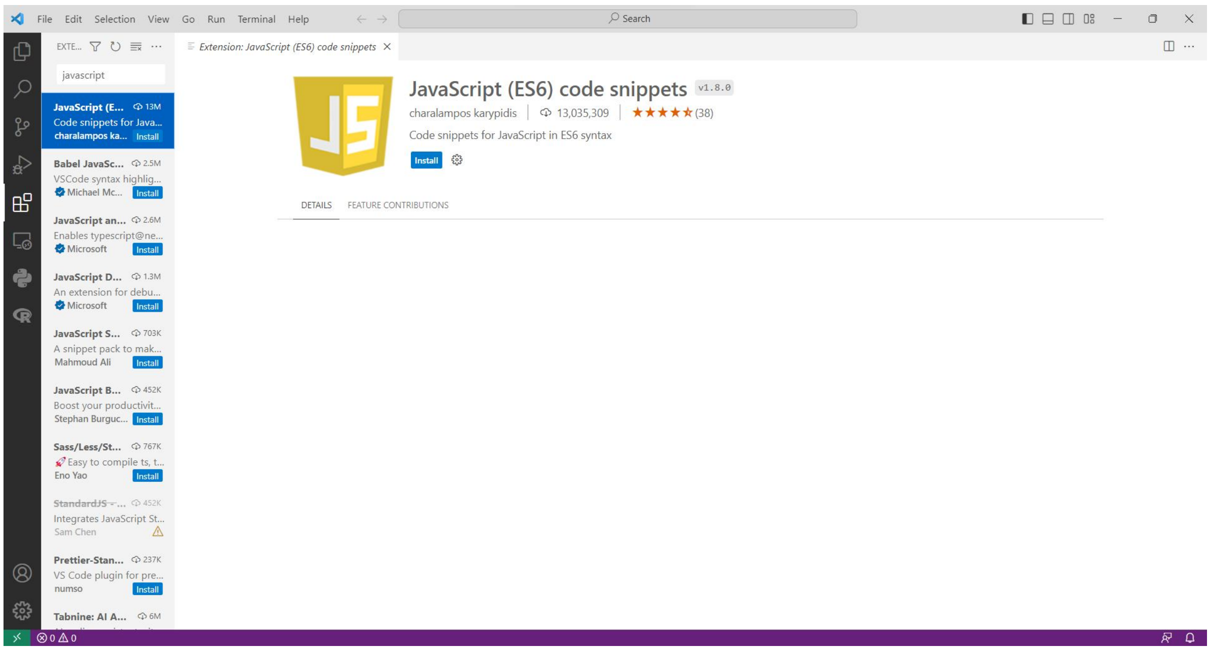
Categories
Other

Extension Resources
Marketplace
Repository
Jasper De Moor

More Info

Published	2018-3-30, 02:41:21
Last released	2019-8-30, 20:54:05
Last updated	2023-2-3, 20:50:04
Identifier	jaspernorth.vscode-pigments

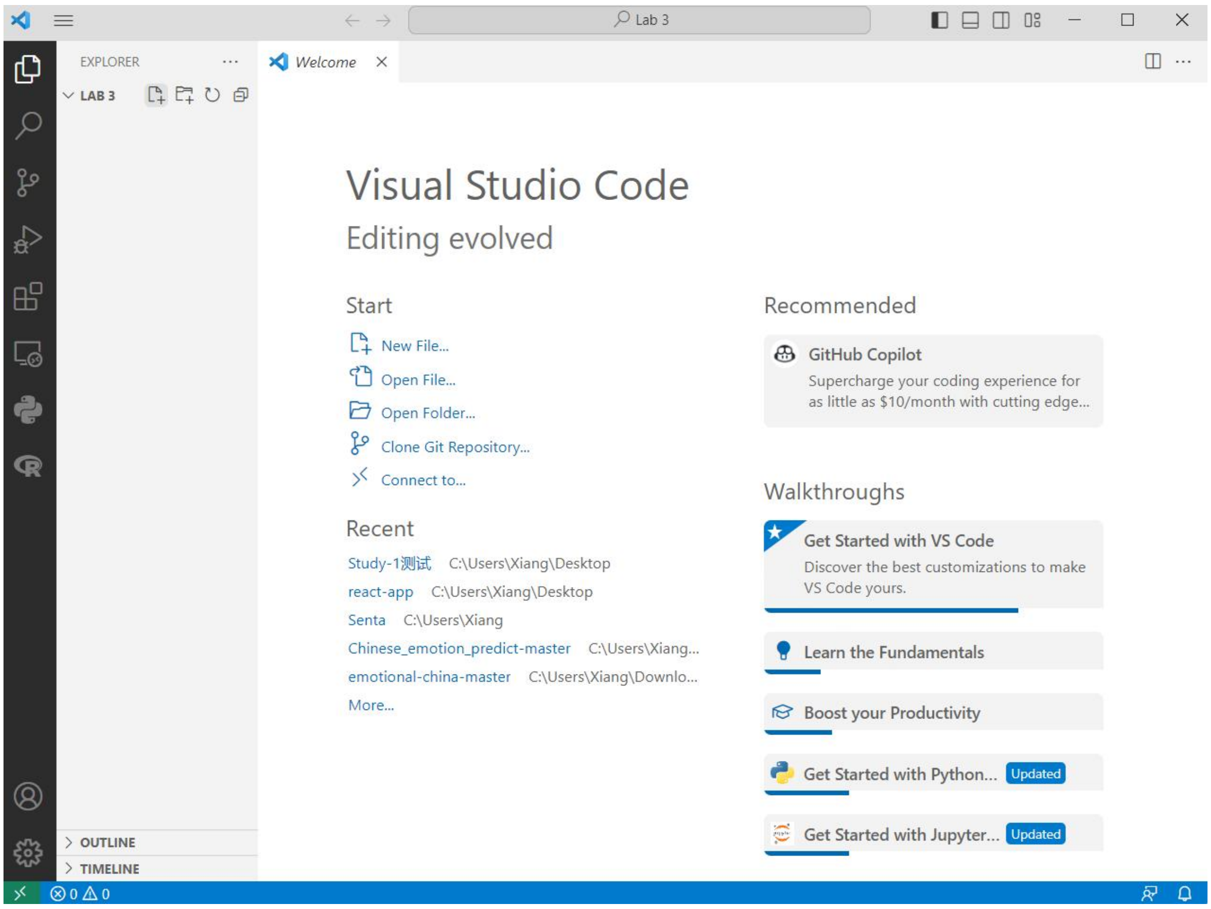
4. Code Snippets (e.g. JS snippet)


File Edit Selection View Go Run Terminal Help
Search
EXTE...
javascript
Extension: JavaScript (ES6) code snippets
JavaScript (ES6) code snippets v1.8.0
charalampos karypidis
13,035,309
(38)
Code snippets for JavaScript in ES6 syntax
Install
DETAILS
FEATURE CONTRIBUTIONS
JavaScript (E... 13M Code snippets for Java... charalampos ka... Install
Babel JavaSc... 2.5M VSCode syntax highlig... Michael Mc... Install
JavaScript an... 2.6M Enables typescript@ne... Microsoft Install
JavaScript D... 1.3M An extension for debu... Microsoft Install
JavaScript S... 703K A snippet pack to mak... Mahmoud Ali Install
JavaScript B... 452K Boost your productivit... Stephan Burguc... Install
Sass/Less/St... 767K Easy to compile ts, t... Eno Yao Install
StandardJS -... 452K Integrates JavaScript St... Sam Chen
Prettier-Stan... 237K VS Code plugin for pre... numso Install
Tabnine: AI A... 6M
0 0

Task1: Make your first HTML file

Create a new folder named “Lab 3” on your laptop, then click on “Open Folder” or directly drag this folder into VS Code to open it.

Then, click on “New File” which is aside of the folder name to create a new file.



Visual Studio Code

Editing evolved

Start

New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...

Recent

Study-1测试 C:\Users\Xiang\Desktop
react-app C:\Users\Xiang\Desktop
Senta C:\Users\Xiang
Chinese_emotion_predict-master C:\Users\Xiang...
emotional-china-master C:\Users\Xiang\Downlo...
More...

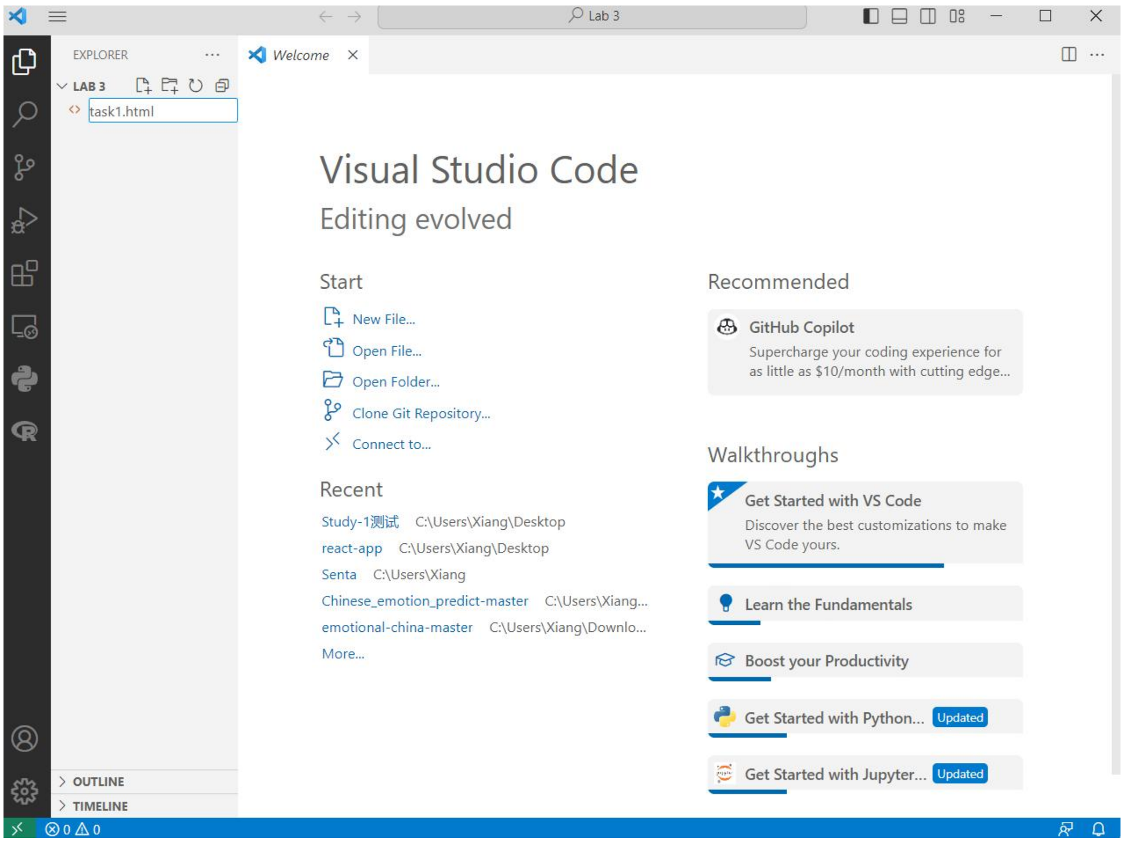
Recommended

GitHub Copilot
Supercharge your coding experience for as little as $10/month with cutting edge...

Walkthroughs

Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Boost your Productivity
Get Started with Python... Updated
Get Started with Jupyter... Updated

Name your file as “task1.html”, but remember to add a .html at the end to classify it as an HTML file.



Visual Studio Code

Editing evolved

Start

New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...

Recent

Study-1测试 C:\Users\Xiang\Desktop
react-app C:\Users\Xiang\Desktop
Senta C:\Users\Xiang
Chinese_emotion_predict-master C:\Users\Xiang...
emotional-china-master C:\Users\Xiang\Downlo...
More...

Recommended

GitHub Copilot
Supercharge your coding experience for as little as $10/month with cutting edge...

Walkthroughs

Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Boost your Productivity
Get Started with Python... Updated
Get Started with Jupyter... Updated

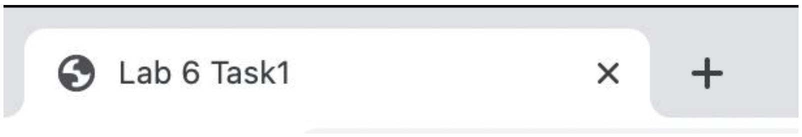
Now that you have a new HTML file take a look at this code.

```
<!DOCTYPE html>
<html>
<head>
          <title></title>
</head>
<body>

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

This long piece of code is how we define an HTML file. Developers call this **boilerplate code**. Take a look at line 4. Notice how the title is surrounded by < >. This defines <title> as the **start tag**. And </title> defines the **end tag**. Together, they define an **element**. Anything you type inside the tag will become the **content** of that tag. Let's try giving a title to our webpage: add something inside the title tag.

```
<head>
          <title>Lab 3 Task1</title>
</head>
```

Directly click on the HTML file in the "Lab 3" folder you created to open the HTML file. Your web browser's tab will have whatever title you put!



All other elements in HTML follow this same format. **Whatever is between the tags is the content** you wish to include.

Now that you have some concept of what elements do, let's go down that boilerplate code again and define what everything is.

<!DOCTYPE html>: Declares this document to be HTML5. Always at the top.
<html>: Contains the entire document. Also called the root element.
<head>: The header element. Contains meta-data (all information that is **not visible**).
<title>: The title of the web page/ document.
<body>: The body element contains all **visible** page content.

Task2: Add simple tags

Now that you have an idea of the structure of HTML files and some tags, let's quickly go through some essential tags.

!!! - Through all these examples, we'd recommend coding out the example in your text editor. Then open the file in a web browser. Every time you make a change, save your file and refresh your web browser to see your changes.

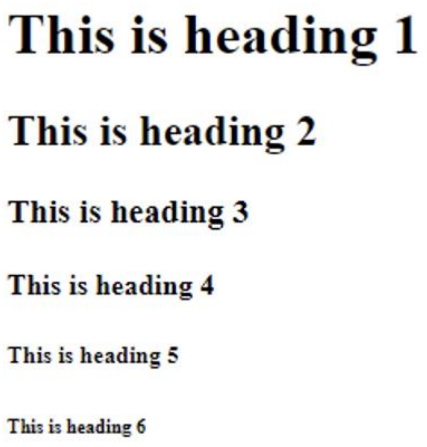
Headings:
Headings are emphasized text titles. Here's an example of how you write one. Since this is a visible element, always put heading tags inside the <body> element.

```
<h1> This is a heading 1 </h1>
```

Notice there's a 1 beside the heading. That's because there are six different-sized headings. With 1 being the largest/ most emphasized heading. Here's a side-by-side example of the different headings. Now try the code below:

```
<!DOCTYPE html>
<html>
<head>
          <title>Headings</title>
</head>
<body>

          <h1>This is heading 1</h1>
          <h2>This is heading 2</h2>
          <h3>This is heading 3</h3>
          <h4>This is heading 4</h4>
          <h5>This is heading 5</h5>
          <h6>This is heading 6</h6>

</body>
</html>
```

Save the file and refresh the browser; you can see:



```
This is heading 1
This is heading 2
This is heading 3
This is heading 4
This is heading 5
This is heading 6
```

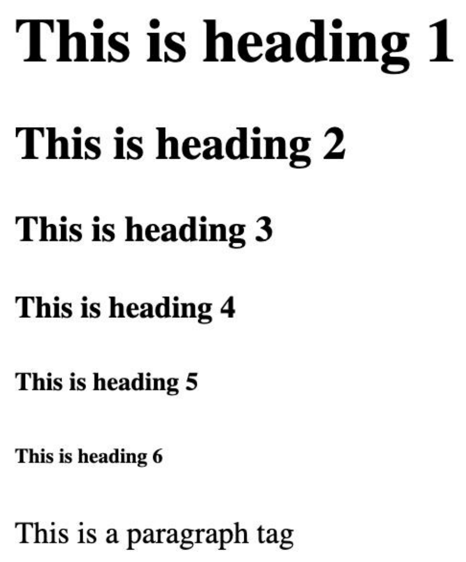
By using headers, you can emphasize different pieces of content better.

Paragraphs

Paragraphs are just blocks of text. Try to add the code below inside the <body>

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

Save the file and refresh the browser; you can see:



This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

This is a paragraph tag

Buttons:

Press them; they're fun. For now, they do nothing, but later down the line, we can add interactions.

```
                    <button> Click me </button>
```

Now try the code below:

```
<!DOCTYPE html>
<html>
<head>
          <title>Buttons</title>
</head>
<body>

          <button>Click Me</button>

</body>
</html>
```

Save the file and refresh the browser; you can see:

Lists:

When you have a lot of related things you want to group in a listed fashion, you can make lists.

There are two parts to every list.

1.) The Definition: What kind of list it is
2.) List elements: The stuff the list contains.

There are two types of lists.

Unordered List:

<ul>
<li> list element </li>
</ul>

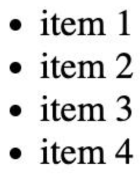
Ordered List:

<ol>
<li> list element </li>
</ol>

Now try to add the code below:

```
<ul>
         <li>item 1</li>
         <li>item 2</li>
         <li>item 3</li>
         <li>item 4</li>
</ul>
```

Save the file and refresh the browser; you can see:



- item 1
- item 2
- item 3
- item 4

Task 3: Link CSS to HTML

CSS stands for Cascading Style Sheets. It is used to style HTML elements. Think of HTML as nouns: things, objects. CSS are the adjectives describing how those things look, how they are styled, and where. It is probably the most fundamental thing that lets websites look good. CSS files are saved as .css files. Like you created new HTML files using the .html extension, do the same for CSS files.

Create a new file under the folder Lab 3 named test.css, and a new HTML file named task3.html

Ok, now how to link the CSS to the HTML project? To do this, we need a link tag. This goes in the head section of your HTML file. Here's an example.

```
<!DOCTYPE html>
<html>
<head>
       <title>Test CSS Link</title>
       <link rel="stylesheet" href="test.css"/>
</head>

<body>
       <h1>Hello World</h1>
</body>
</html>
```

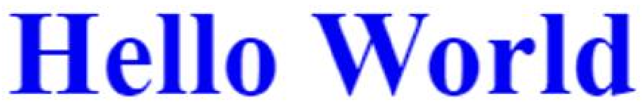
The **rel** attribute here says that the thing we are linking is a stylesheet. Type says that this stylesheet will be composed of CSS. Here is the location and name of the CSS file you want to link. If your CSS file is in the same folder as the HTML file, you simply need to write its name like in the example.
So how do I style something? CSS syntax is defined in three parts: a Selector, a Property, and a Value. Imagine we have an h1 in our HTML.

```
<!DOCTYPE html>
<html>
<head>
       <title>Test CSS Link</title>
       <link rel="stylesheet" href="test.css"/>
</head>

<body>
       <h1>Hello World</h1>
</body>
</html>
```

Type the following code to CSS file:

```
h1{
   color: blue;
   Font-size: 50px;
}
```

The **Selector** is the h1 in the beginning. It says this is the thing we want to style. Inside the brackets are two things. On the left is your **property**; this is what you want to change. After a colon, you have the **value** you want to assign. For example, setting the color blue will change the text color of the h1 to blue. Changing the font size will make the text bigger.


Hello World

This is what your H1 should look like now. So just like HTML's tags, CSS has many pre-made properties, too many in fact to list here. But just like HTML, W3 Schools, web developer documentation, and other sites that compile CSS style properties are available.
https://www.w3schools.com/css/default.asp

Keep a dedicated documentation page open, but we'll begin to introduce the more critical properties and the most valuable tools CSS offers. We will dive deeper into CSS in the next chapter. Feel free to play around with the basics and check out the attached link for extra practice.

Task 4: Introduction to JavaScript

So far, we've seen what we can do with HTML and CSS. HTML provides the markup or the page's content, and CSS adds styles and decorates the page accordingly. However, What if we want to add some functionality to the page? For example, how can we add an image slider to our page or target specific HTML elements and manipulate them? This is where the programming language Javascript comes into play.

Before we jump ahead to see how we can use Javascript to add functionality to a web page, let's first examine the language and learn its syntax.

Variables in Javascript:

Simply add the var keyword before the variable name to declare a variable in JS (Javascript). The following displays two ways to create variables.

```
var a = 10;
var b = 15;
var z = a + b;
```

// creating multiple variables in a single line

```
var x, y, z;
x = 1;
y = 2;
z = 3;
```

Notice how we end every statement with a semicolon (like with C#!). However, in Javascript, no matter the type of variable you wish to create, all you need to do is to make sure that you add the var keyword at the very beginning of the statement. Here are some basic examples of how you can declare variables of various types:

```
var name = “John”;
var letter = “a”;
var number = 5.95;
Var isFound = False;
```

Javascript Operators:
You can perform a wide range of operations on variables in Javascript.

Operators on numbers:
When working with numbers such as integers and float, you can add, subtract, multiply, increment, and more. Here is a snippet of some of the most common operations on numbers:

```
var x = 10;
var y = 5;
var z = 10 + 5;                   // Addition

var w = 10 - 5;                   // Subtraction
var a = 10 * 5;                   // Multiplication
var b = 10 / 5;                   // Division
var x++;                          // Increment
// This is the same as x = x + 1 → 11
var y--;                               // Decrement
// This is the same as y = y - 1 → 4
var exp = x ** 2;                 // Exponentiation
```

Operators on String variables:
You have to be careful when using operators such as addition on strings. Whenever you use the + operator on strings, you are basically “gluing” them together. Another term for that is **string concatenation**.

Here’s an example of string concatenation.

```
var firstName = “John”;
var lastName = “Smith”;
var fullName = firstName + lastName;
```

What do you think fullName will show when we print it?

The actual result is **JohnSmith**.
When you add strings together, Javascript “glues” the ‘n’ and ‘S’, and because there is no space between them, the variable fullName prints out JohnSmith.

Javascript Conditionals:
When you are writing code, you will almost certainly run into a situation where you have to tell your program what to do when faced with multiple decisions. You can use conditional statements in your code to do this. In Javascript, we have the following conditional statements:

1) The **if** statement executes a block of code if the condition is true.
```
If (condition) {
   // execute some code if the condition is true
}
```

2) The **else** statement is used to specify a block of code to be executed if the condition is false.
```
If (condition) {
   // execute some code if the condition is true
} else {
   // execute this code of the condition is false
}
```

3) The **else if** statement specifies a new condition if the first condition is false.
```
 If (condition 1) {
   // execute some code if the condition is true
} else if (condition 2) {
   // execute this code of the condition is false
} else {
   // execute this code of the condition is false
}
```

Loops in Javascript:
Like most programming languages, Javascript offers programmers the ability to execute a block of code several times through loops.

For loop:
For loops in Javascript have the following syntax (similar to C#)

```
For (statement 1; statement 2; statement 3) {
   // execute block of code
}
```

Statement 1, also known as the **start**, is executed once and usually provides us with a starting point for the loop to run.
Statement 2, also known as the **stop**, is the condition that must be true for the loop to continue running. Once this condition is false, the loop stops.
Statement 3, also known as the **step**, is executed every time after the code block is executed. It is usually used to increment the first states to allow the loop to be advanced or run.

Example:
```
for (i = 0; i < 10; i++) {
       console.log("number is: " + i);
}
```

Notice here that we declare a variable i (our index variable) and initialize it to 0, our starting point. Then, we check if i is less than 10. If it is, we create an alert box that pops up on the page and displays the number that i is currently assigned. Then, i is incremented by adding 1 to its previous value, and the process repeats itself, this time starting from statement 2.

While loop:
Another loop used in Javascript is what's called the while loop. This loop executes a block of code as long as a specified condition is true. Here is how a while loop looks like:

```
while (condition) {
   // code block to be executed
}
```

Example:
```
i = 0
while (i < 10) {
     console.log("number is: " + i);
     i++;
}
```

Arrays in Javascript:
Arrays in JavaScript are used to store multiple values in a single variable. That is, it is a special variable that can hold more than one value at once. The following is an example of how to create an array in JS.

```
var scores = [75, 45, 70, 65, 55, 90, 95];
var cities = ["Paris", "Vancouver", "London", "Berlin",
"Cairo", "Seattle"];
```

Arrays are very useful because, for example, if we wanted to create a separate variable for each of those cities, we would have to create 6 different variables. Imagine an array with over a million elements! Since arrays are special variables, they also have special rules. Let’s say we wanted to access the city of London on its own. How would we do that?

To answer this question, we need to understand that arrays are indexed from 0 to the number of elements - 1. To elaborate, every element in the array has an index number. So, for the cities array, the array’s indices look like the following:

“Paris”	**“Vancouver”**	**“London”**	**“Berlin”**	**“Cairo”**	**“Seattle”**
0	**1**	**2**	**3**	**4**	**5**

Therefore, we need to look for the array’s second index (3rd element) to retrieve London. Notice here how the last index is 5 and not 6. Even though the cities array has 6 elements, the last index is 5. This is because, as mentioned earlier, the indexing starts at 0 and not 1. So now that we understand indexes, the code to retrieve the city London and Seattle is as follows:

```
cities[2];
cities[5];
```

If we want to find out how many elements are in an array, we can use the length function that is available for all arrays.

```
cities.length;
```

This will come in handy when iterating (loop) through an array.

Looping through an array:

Imagine you had to check every link in the navigation bar or that you were trying to find a specific element on a web page. In this case, you would need to use a loop to iterate through the set of elements on the page. To illustrate a simple example, let’s go back to our city's array. If we wanted to print out every city simply, we’d have to do the following:

```
for (i = 0; i < cities.length; i++) {
        // dialog("Current city: " + cities[i]); no longer work
        alert("Current city: " + cities[i]);
}
```

JavaScript Functions

The last topic that we will cover before we conclude our introduction to Javascript will be functioning in JavaScript. So, what are functions? Imagine that you need to compute the sum of two numbers in your program and that you have to print out the result afterward. This can look something like this:

```
sum = a + b;
console.log(sum);
```

And for the sake of argument, imagine that you had to use this computation in multiple places in your program. Instead of having to write these 2 lines every time you need to add two numbers, you can put this code inside a function and call that function instead. For example, the code for how you would perform this could look something like this,

```
function addNums(a, b) {
   sum = a + b;
   console.log(sum);
}
```

Therefore, using functions enables us to reuse code: define the code once and use it many times. You can also use the same code many times with different arguments to produce different results. For example, arguments a and b can be different numbers every time we call the function. You can use the same code many times with different arguments to produce different results.

The syntax for a Javascript function is quite simple. It is defined with the **function** keyword, as shown above, followed by a name and then parentheses (). Names can contain letters, numbers, and underscores. Therefore, addNums, person2, and last_name are valid function names.

The parentheses may include parameter names separated by commas:

(parameter1, parameter2, ...)

The code to be executed by the function is placed inside curly brackets: {}.

```
function name(parameter1, parameter2, parameter3) {

  // code to be executed

}
```

Function parameters are listed inside the parentheses () in the function definition. Function arguments are the values received by the function when it is invoked. Inside the function, the arguments (the parameters) behave as local variables. Calling a function is very simple. You simply put the name of the function and include any arguments that are needed for the position to work. Using our addNums example, we simply type

```
addNums(5, 6);
```

Return statement:

Functions can also return values. Instead of printing the sum of the numbers a and b, we can return that value instead. The following example demonstrates return statements:

```
function addNums(a, b) {
   sum = a + b;
   return sum;
}
```

Once this function is invoked (called), it will output the value of the sum of the two numbers that were passed in as parameters. That is, it will “return” the sum. As you will see later, this is very important because we can store this value in a variable. For example,

```
var sum = addNums(5,6);
```

Here, the variable sum will store the result of the function’s output, which happens to be 11 in this case. Do not be confused with the sum in the return statement and the sum variable below. They’re entirely different.

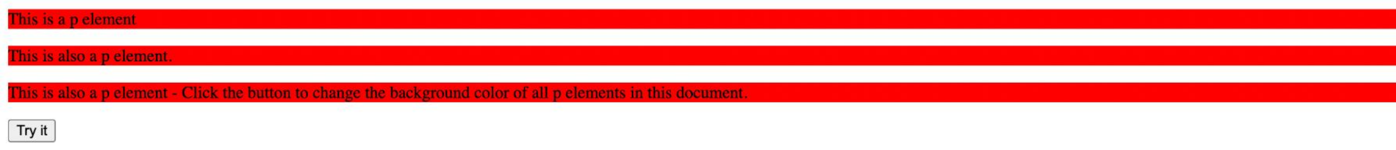
Okay. Let’s now see an example of what JavaScript can do. Don’t worry if you don’t fully understand everything in the following example. We will explain everything in the next lab!

CODE:

```
<!DOCTYPE html>

<html>

<body>

<p>This is a p element</p>

<p>This is also a p element.</p>

<p>This is also a p element - Click the button to change the
background color of all p elements in this document.</p>

<button onclick="changeBackgroundToRed()">Try it</button>

<script>
```

```
          function changeBackgroundToRed() {

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

               for (var i = 0; i < x.length; i++) {

                    x[i].style.backgroundColor = "red";

               }

          }

     </script>

     </body>

     </html>
```

Results:


This is a p element

This is also a p element.

This is also a p element - Click the button to change the background color of all p elements in this document.

As you can see, within the script tags, we've created a JavaScript function that searches for all p tags in the document and changes their background color to red. Some parts of this code might seem a bit overwhelming, and that is ok. There are still a few things left before you fully understand what is happening here. This is just a brief demo to show what JavaScript can do.