Hello Code 2 Summary

Congrats!

Congratulations! By taking this Hello Code 2 course, you have not only shown an interest in learning something new, but also have taken a big step into making it happen.

Assumptions

I assume you are familiar with 1) HTML (tags, elements, and attributes) and CSS (selectors)

HTML

div	**div** is an HTML tag
	This is a **div** tag used in a page. When an HTML tag is used in a page, it becomes an **element**
	This is a **div** element with a **class** attribute set to **"myClass"**

CSS

div{ color: red; } Hello	**Tag selector** Using **div** tag selector will turn text in all **div** elements red.
div**.myClass** { color: blue; } Blue Text 	**. Class selector** Using **div** tag selector with **.myClass** will turn all **div** elements that have the **myClass** class text red.
div**#myId** { color: green; } Green Text 	**# Id selector** Using **div** tag selector with **#myId** will turn all **div** elements that have the **#myId** id text green

JavaScript

JavaScript is how a website communicates with the person browsing a website. That person is called the **user**.

JavaScript is used to **send** and **get** data, say your bank account summary, from a server. Your bank account doesn't live on your computer. You first have to **authenticate** yourself, that is, prove you are you by providing (**sending**) your **username** and **password** to your bank by filling out that **form** on the login page. That information goes to the banks **server**, validates your **credentials** (username and password) and then the server goes to the bank's database, gets your account information, and sends it back to the browser. The browser then renders (displays) that data in a nice looking table that you can use to balance your checkbook.

Of course, it's a little more complicated than that, but this will work for now.

Where to put JavaScript

Where does JavaScript go? That depends on which school of thought you subscribe to.

School of Thought #1
JavaScript goes at the top of the page, between the **<head></head>** tags, just the CSS.

School of Thought #2
JavaScript goes at the bottom of the page, still between the **<body></body>** tags, just like HTML, but just above the **</body>** tag.

The idea behind #2 is that the browser loads everything from top to bottom. So, the page first loads the CSS, then the HTML, then the JavaScript loads and does its thing.
If the JavaScript were at the top, it would slow down the HTML from displaying, because it's loading the JavaScript, and users don't have a lot of patience for slow websites.

I subscribe to **School of Thought #2**.

Here's the layout of a basic web page that includes the **<script></script>** tags. The **script** tag automatically presumes **JavaScript**.

base-page.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Base Page</title>
  </head>
  <body>
    <!-- <- HTML MULTI-LINE COMMENT START
    HTML goes at the top of the body.
    -->

    <script>
      /* <- JAVASCRIPT MULTI-LINE COMMENT START,
       Anything between is a comment and WILL NOT BE EXECUTED
       JavaScript goes at the bottom of the body
      */
    </script>
  </body>
</html>
```

Hello World! Version 1

Hello World! Is the “first” application developers write as part of their *first step of a thousand mile journey*.

What We’re Going To Do

We’re going to use JavaScript to:

1) Use an **alert** function to display **Hello World!** when a user **clicks** a button
    a) **alert** is a built-in function inside of **JavaScript**.
    b) A **function** is a command that gets executed when **invoked** (called).

How We’re Going To Do That

2) Buttons have a built-in attribute called **onclick** that we can use to point to a JavaScript function.
3) **alert** displays that “annoying” popup message. We’re going to use that alert to display **Hello World!** and then not use **alert** much afterwards.

CAUTION: CASE MATTERS!!!!

Function names and variable names have to:

1) Match case
    a) **alert** *IS NOT THE SAME AS* **Alert**
    b) **SayHelloClicked** *IS NOT THE SAME AS* **sayhelloclicked**
2) Be spelled the same

a) **first_name** *IS NOT THE SAME AS* **first-name**

01-hello-world-v1-end.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Hello World 1</title>
  </head>
  <body>
    <button type="button" onclick="alert('Hello World!')">
      Say Hello World
    </button>
  </body>
</html>
```

What's going on?

`onclick="alert('Hello World!')"`

onclick

- is the HTML attribute event we want to use when the user **clicks** the button.

"alert('Hello World!')"

- is the attribute value
- Note the whole thing is surrounded in quotes
    - This is the same as if you were assigning a class or id.

alert

- is the JavaScript function to execute when button is clicked.

('Hello World!')

- is the **string *argument*** you want to send to the alert function.
    - Alert needs to know what you want to say.
- Anything within the parenthesis **(**...**)** is an argument being sent to the function.
- You can set this to whatever you want.

The quotes can be either double-quotes(`"`) or a single-quote(`'`).

Note: You can't have double-quotes nested inside of double quotes.
Example: You can't do this: `"alert("Hello World!")"`
You could reverse the quotes: `'alert("Hello World!")'`

Links To Learn More

alert - https://www.w3schools.com/jsref/met_win_alert.asp
functions - https://www.w3schools.com/js/js_functions.asp
onclick - https://www.w3schools.com/jsref/event_onclick.asp
strings - https://www.w3schools.com/js/js_strings.asp

Challenge

1) Using **Hello World! Version 1** as a model and:
    a) Create a new button and alert to say **¡Hola Mundo!**
    b) Create a new button and alert to say **Bonjour le monde!**
    c) Again, but with the language of your choice!
    (Go to Google translate and copy paste)
        i) Russian: Привет, мир!
        ii) Japanese: <こんにちは世界！

Hello World! Version 2

In **Hello World! Version 1**, we used a JavaScript alert function inside the **onclick** attribute to say “Hello World!”.

In **Hello World! Version 2**, we’re going to create a **User Defined Function** to call the JavaScript alert function to say “Hello World!”.

User Defined Function

User Defined Function sound fancy, but it’s just a function that you create.

What is a **User Defined Function**?
https://www.w3schools.com/js/js_functions.asp
A User Defined Function is a “container” for a series of statements that get executed when the function is **invoked** (called).

What is a **statement**?
https://www.w3schools.com/js/js_statements.asp
A statement is a line of instruction executed by the browser.

A function can:

- Take 0 or more arguments
    - Arguments are the variables that get sent to a function
    - The *string* sent to alert was an argument.
- Return a value.

Creating your first function:

https://www.w3schools.com/js/js_functions.asp
https://www.w3schools.com/js/js_function_definition.asp
https://www.w3schools.com/js/js_function_parameters.asp

https://www.w3schools.com/js/js_function_invocation.asp

To create a function

1) You first identify that you are creating a function by declaring **function**
2) Then the function name.
3) Then parenthesis ()
    a) Arguments go inside the parenthesis
4) Then curly braces {}
    a) Statements to be executed
    b) This is the function body.

```
   1            2           3   4(start)
function mySimpleFunction( ) {
  //This is the function body
} 4 (end)
```

When naming a function:

- function names follow the same restrictions as variable names
    - Can contain Letters, numbers, underscore, and dollar signs
    - Must start with a letter, underscore, or dollar sign
    - Can take 0 or more arguments
    - Can, but doesn't have to, return a value
    - Tip
        - Have function name represent what it does
        - Use camelCasing (First word is lower case, every new word is has a capital first letter)
            - firstName, lastName, dateOfBirth, password

Some function examples:

```
// <- This is a SINGLE LINE COMMENT.
// COMMENTS DON'T GET EXECUTED
// This function takes two arguments: x and y
// and returns the result of x * y;
function mutiplyTwoNumbers( x, y ){
    return x * y;
}

// This function is what we're going to use below
function sayGreeting( greeting ){
    alert( greeting );
}
```

What We're Going To Do

We're going to use JavaScript to:

1) Create a function to display **Hello World!** when a user **clicks** a button
    a) Our function will use **alert** to display "Hello World!".

How We're Going To Do That

2) Our **onclick** function call will *send* "Hello World" as an argument.
3) sayHelloWorld will *receive* "Hello World" as an argument and it will be assigned the **variable name** *greeting*.

01-hello-world-v2-end.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Hello World 1</title>
  </head>
  <body>
    <button type="button" onclick="sayHelloWorld('Hello World!')">
      Say Hello World
    </button>

    <script>
      function sayHelloWorld(greeting){
        alert(greeting);
      }
    </script>
  </body>
</html>
```

What's going on?

```
onclick="sayHelloWorld('Hello World!')"
```

Notice, we essentially just replaced the function name **alert** from version 1 with a new function name: **sayHelloWorld**.

We are sending `'Hello World!'` to our function `sayHelloWorld`

```
function sayHelloWorld(greeting){
  alert(greeting);
}
```

`function sayHelloWorld(greeting)` is our function declaration
`greeting` is our `'Hello World!'` now represented as a variable

Challenge

1) Using **Hello World! Version 2** as a model and:
    a) Create a new button and *sayHelloWorld* to say **¡Hola Mundo!**
    b) Create a new button and *sayHelloWorld* to say **Bonjour le monde!**
    c) Again, but with the language of your choice!
    (Go to Google translate and copy paste)
        i) Russian: Привет, мир!
        ii) Japanese: <こんにちは世界！

JavaScript document (DOM) API

DOM - Document Object Model (The Web Page from JavaScript's perspective)
API - Application Programmer Interface (toolbox)

From here on, we will be limiting our use of **alert** and using the **DOM** to display our messages.

JavaScript is Object based. What's an Object? In simple terms, an Object is a thing, a noun. Objects have attributes and methods.

An attribute is a characteristic.
A method is a verb.

You, as a person, are an object.
You have attributes:

- Height
- Weight
- Eye color

And you also have methods (You do something)

- Walk()
- Run()
- Eat()
- Sleep()

The very web page you're working on is an **object** and has properties and methods and you use the **document** API to interact with them.

This is where your CSS knowledge becomes even more valuable.

To interact with specific parts of your web page, you need to identify those parts of your web page.

To interact with a single element on your web page
You use the id attribute.

Example:
```
<div id="greetingsDiv"></div>
```

To interact with many similar elements on your web page
You use the class attribute.

Example:
```
<div class="tab"></div>
<div class="tab"></div>
```

But How Does JavaScript know how to interact with the DOM?
You use the document API

https://www.w3schools.com/js/js_htmldom_document.asp
```
document.getElementById(idNameGoesHere);
document.getElementsByTag(tagNameGoesHere);
document.getElementsByClassName(classNameGoesHere);
```

`document` is the object and `getElementById()` is a method.

Notice:
getElementById is singular, whereas **getElementsByTag** and **getElementsByClassName** are plural.

getElementById will return a single DOM element.

getElementsByClassName and **getElementsByTag** return an **array** of DOM elements

In our previous example, we used the built-in JavaScript provided **alert** to display a popup to our user.

In this next example, we're going to use the DOM API to display our message. We're going to have two buttons. One will put "Hello World" in an HTML element and the other button will clear that element so we can say it again.

Hello World! 2

What We're Going To Do
We're going to use JavaScript to:

1) Use another JavaScript API function to display **Hello World!** in an HTML element when a user **clicks** a button
    a) JavaScript API functions are functions built inside of **JavaScript**.
        i) API = Application Programmer's Interface
        ii) It's a *toolbox* of commands that help developers do their job.
        iii) Eventually, you'll write your own *toolbox* of commands that make your life easier..

How We're Going To Do That

2) Buttons have a built-in attribute called **onclick** that we can use to point to a JavaScript function.
3) We're going to send the text we want to say to our function. One button will send "Hello World!" and the other button is going to send an empty string, or **""**.

01-hello-world-v3-end.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Hello World 1</title>
  </head>
  <body>

    <button type="button" onclick="sayHelloWorld('Hello World!')">
      Say Hello World
    </button>

    <button type="button" onclick="sayHelloWorld('')">
      Clear
    </button>

    <div id="greetingsDiv"></div>

    <script>
      function sayHelloWorld(greeting){
        document.getElementById("greetingsDiv").innerHTML =
greeting;
      }
    </script>
  </body>
</html>
```

What's going on?

`<div id="greetingsDiv"></div>` is our HTML element where we're going to place our greeting. *Note:* We're using `id="greetingsDiv"` so JavaScript can *find* this element.

`document.getElementById("greetingsDiv").innerHTML = greeting` is where the magic happens.

There are actually multiple things going on here.

1) `document.getElementById("greetingsDiv")` is fetching the element.
2) `.innerHTML = greeting` is assigning our greeting to the inner HTML of the div element. Essentially, we're changing the content from nothing to something.
    a) Before: `<div id="greetingsDiv"></div>`
    b) After: `<div id="greetingsDiv">Hello World!</div>`

This line

```
document.getElementById("greetingsDiv").innerHTML = greeting;
```

assumes there will always be an element identified by `id="greetingsDiv"`. But what if it's not there? Say you misspelled it, or executed some JavaScript that removed it from the DOM?

Debugging

We're going to learn how to do some **debugging**.

Go back to **01-hello-world-v3-end.html** and change
`document.getElementById("greetingsDiv").innerHTML = greeting;`

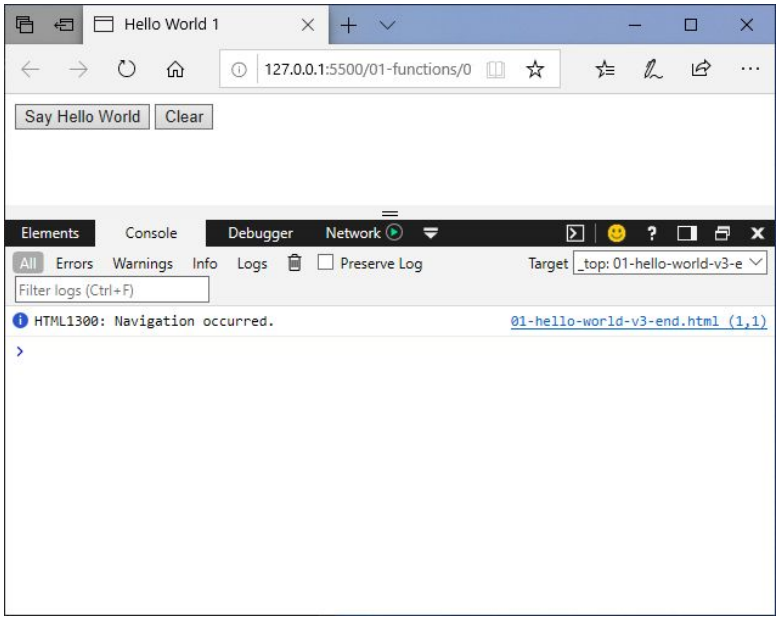
to
`document.getElementById("greetingsDiva").innerHTML = greeting;`

Now view your page and press **Say Hello World**.

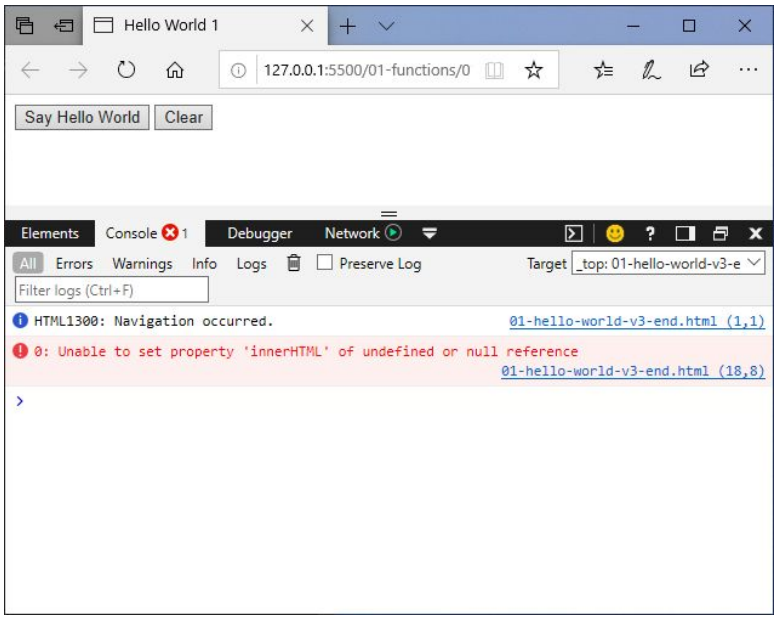
Nothing is happening. We broke the page. We're telling the DOM to go find an element that isn't there. We know it's not there, but we're going to pretend we don't and put on our **debugging hats** and going on a **bug hunt**.

Developer Tools (F12)

Developer Tools are usually available on browsers by pressing F12.



Press **Say Hello World** again and look at the **console** tab.



The **Console** is displaying the error.
Unable to set property ‘innerHTML’ of undefined or null reference

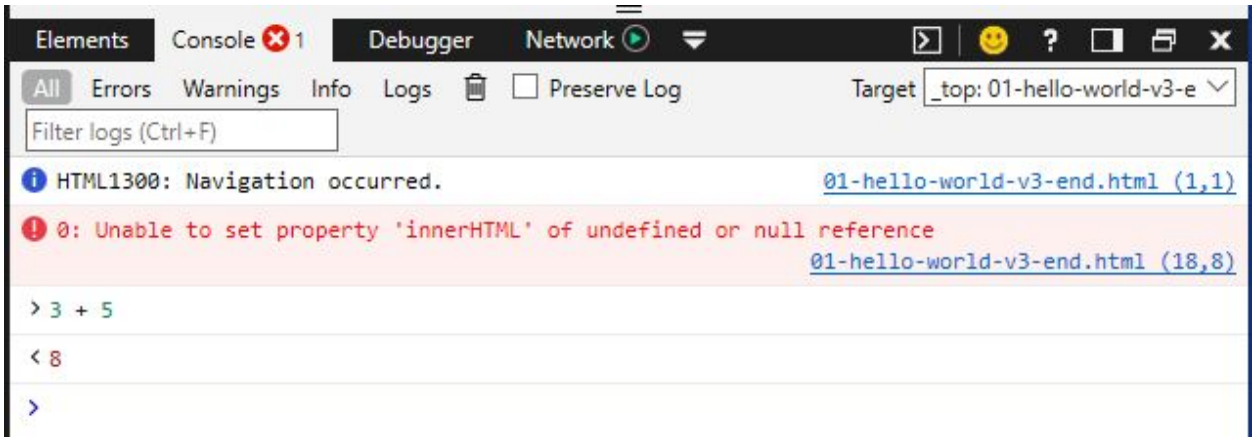
If you click on the link, 01-hello-world-v3-end.html (18,8) it will take you to the offending function.

The **Console** is saying it’s Unable to set property ‘innerHTML’ of undefined or null reference.

Where are we setting `.innerHTML`? On
`document.getElementById("greetingsDiva")`

See that > at the bottom of Console? That's where we can type in JavaScript statements.

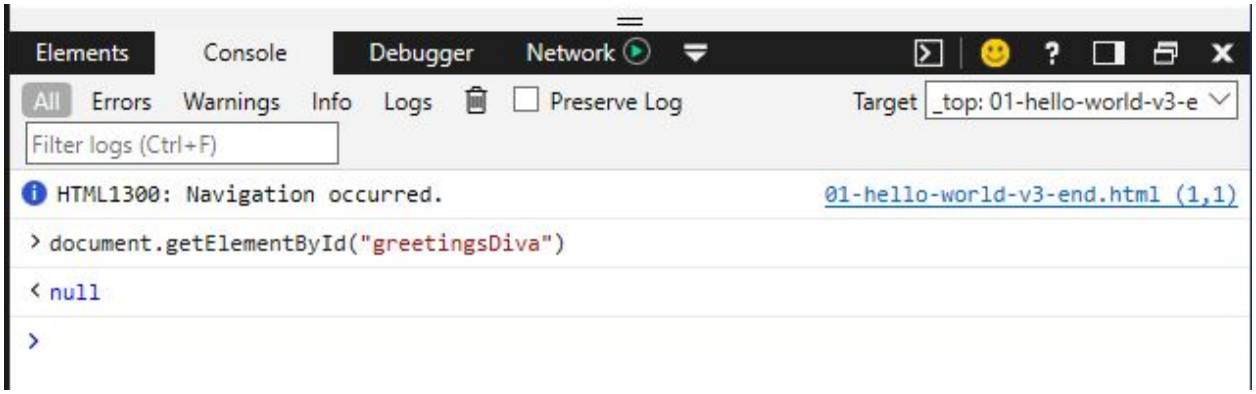
Go ahead, type in 3 + 5 and press enter.



The console returned the value of 3 + 5: 8.

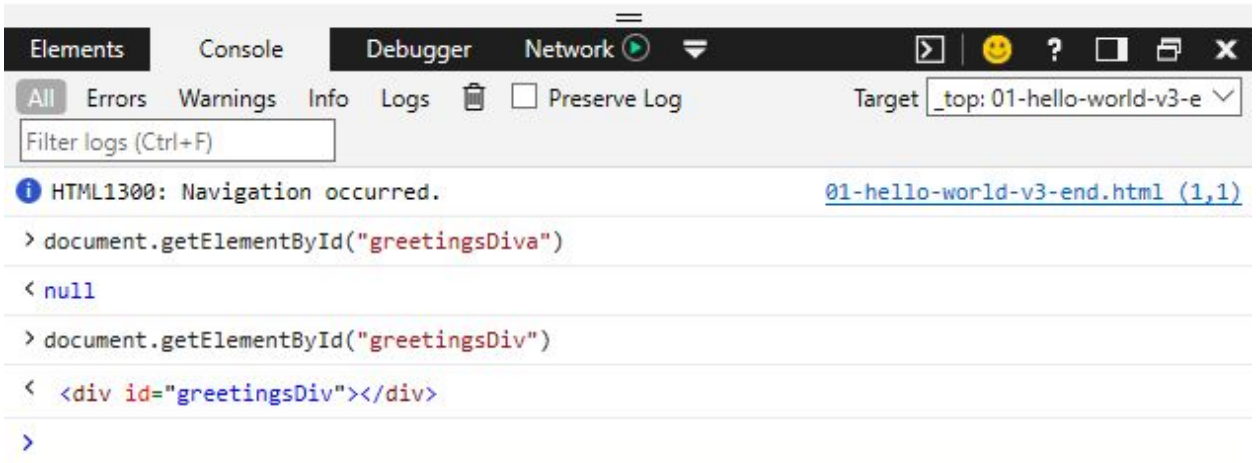
Now, let's take a look at what the browser sees..

Type in the statement that's causing the problem and press Enter:
`>document.getElementById("greetingsDiva")`



The console returned **null**, which means it didn't find an element with an id of `"greetingsDiva"`. A **null** is nothing. It has no methods and no properties because it's void of any value. In short, a **null** doesn't have a `.innerHTML` associated with it.

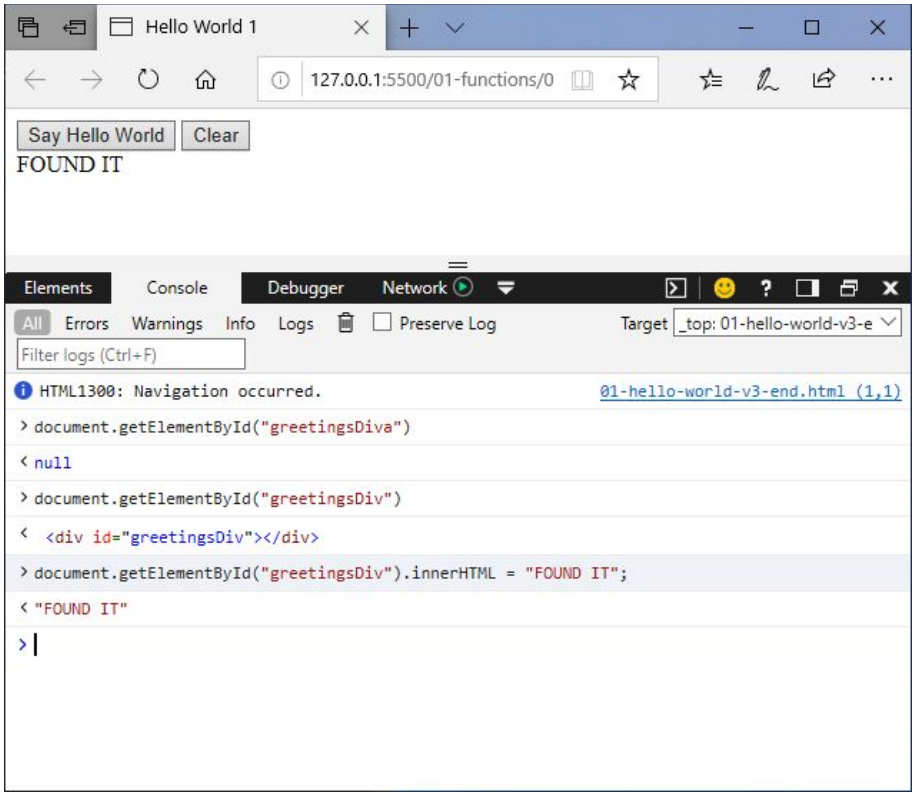
But, we know the element is really called `"greetingsDiv"`, so let's type that in correctly and press Enter.
`>document.getElementById("greetingsDiv")`



Now it returned `<div id="greetingsDiv"></div>`. That something we can do a `.innerHTML` on.

Now that you found the desired HTML div element, let's type in our command with .innerHTML
`document.getElementById("greetingsDiv").innerHTML = "FOUND IT";`

And look at the result.



Notice it not only returned `"FOUND IT"`, but it also set the HTML in your page as well.

Now, look to see what happens when you execute

```
>document.getElementById("greetingsDiv")
```

Notice the **innerHTML** of the div element now contains our text **FOUND IT**.

So, now we can go to the offending function, and fix the incorrect id and run our app again and it should work.

1) Using **Hello World! Version 3** as a model and:
    a) Create a new button and *sayHelloWorld* to say **¡Hola Mundo!**
    b) Create a new button and *sayHelloWorld* to say **Bonjour le monde!**
    c) Again, but with the language of your choice!
    (Go to Google translate and copy paste)
        i) Russian: Привет, мир!
        ii) Japanese: <こんにちは世界！