

Introduction to Javascript

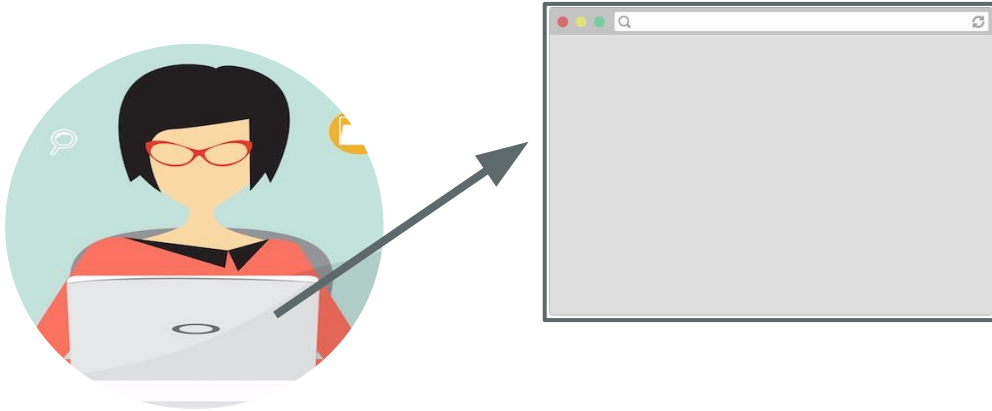
Content mostly taken from <https://web.stanford.edu/class/cs193x/>

How do web pages work
again?

You are on
your laptop

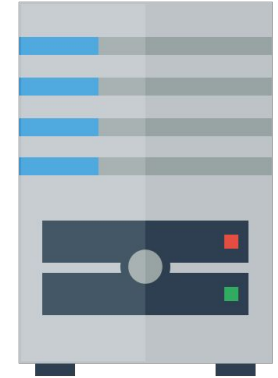
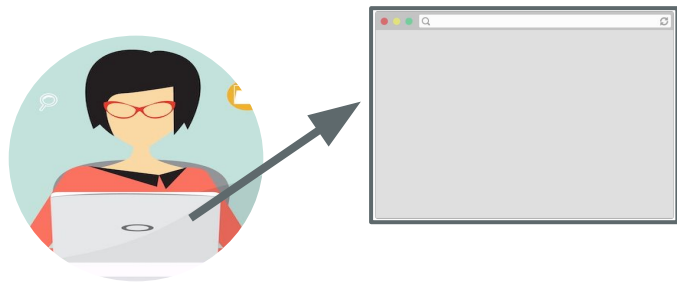


Your laptop is running
a web browser, e.g.
Chrome



You type a URL in
the address bar and
hit "enter"



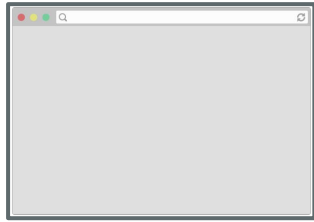


(Warning: Somewhat inaccurate,
massive hand-waving begins now.

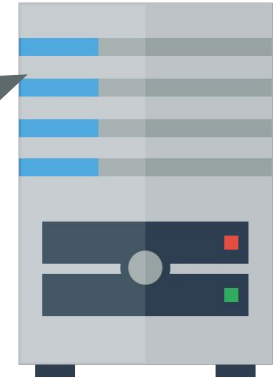
See [this Quora answer](#) for slightly more detailed/accurate handwaving)

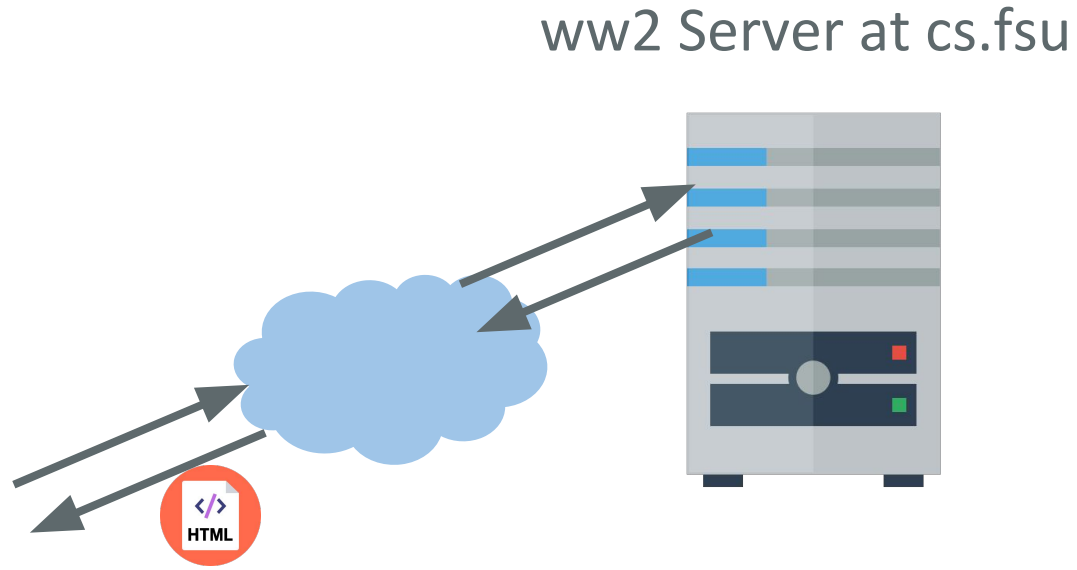
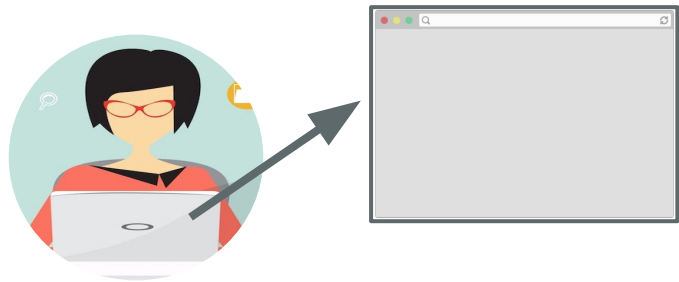
ww2 Server at cs.fsu

Browser sends an HTTP request saying
"Please GET me the index.html file at
<http://ww2.cs.fsu.edu/~faizian/cgs3066>"



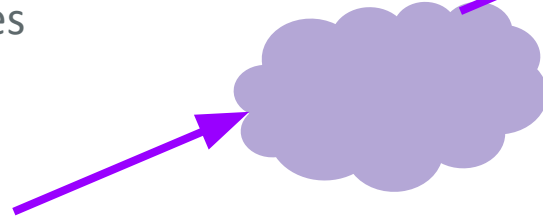
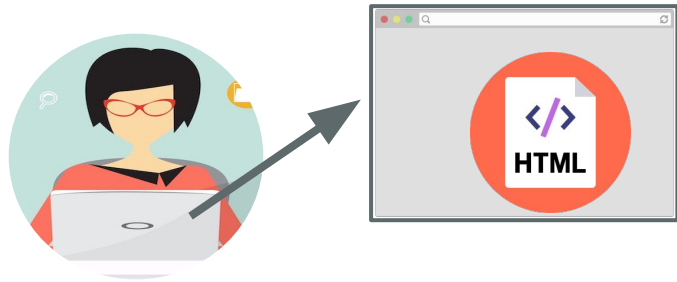
(Routing,
etc...)



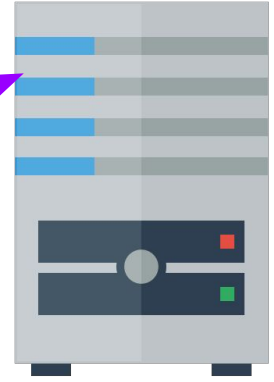


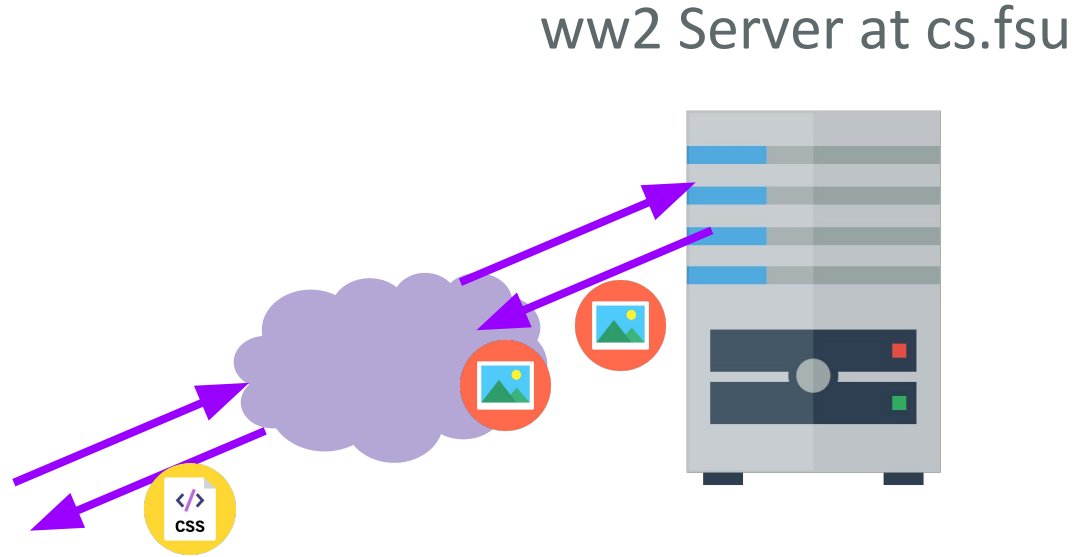
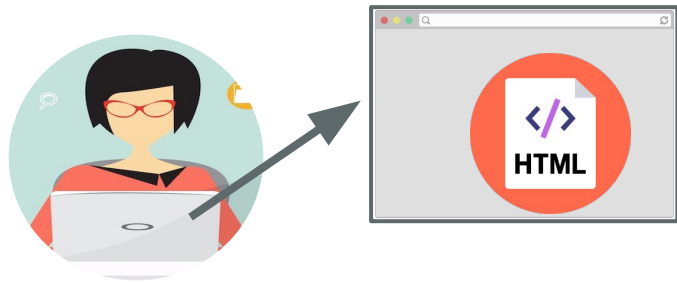
Assuming all goes well, the server responds by sending the HTML file through the internet back to the browser to display.

The HTML will include things like `` and `<link src="style.css" .../>` which generate more requests for those resources



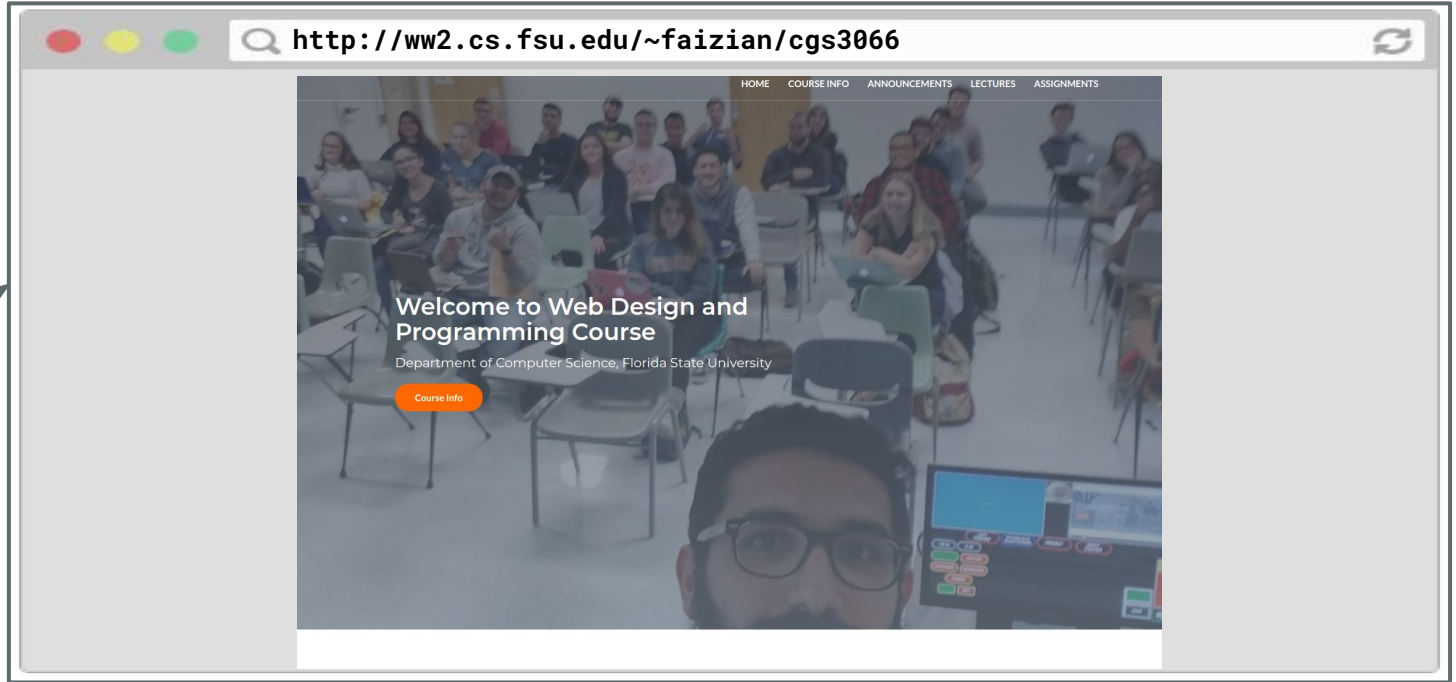
ww2 Server at cs.fsu





And the server replies with those resources for the browser to render

Finally, when all resources are loaded,
we see the loaded web page





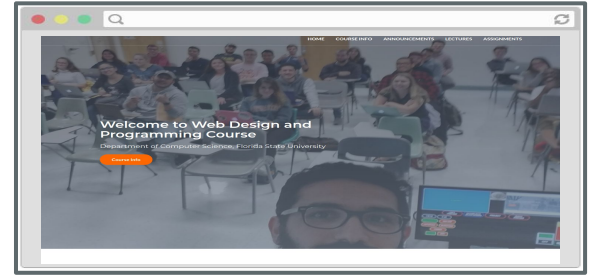
Describes the content and structure of the page

+



Describes the appearance and style of the page

produces



A web page...
that doesn't do anything

What we've learned so far

We've learned how to build web pages that:

- Look the way we want them to
- Can link to other web pages



But we don't know how to build web pages that **do** anything:

- Get user input
- Save user input
- Show and hide elements when the user interacts with the page
- etc.



What we've learned so far

We've learned how to build web pages that:

- Look the way we want them to
- Can link to other web pages



But we don't know how to build web pages that *do* anything:

- Get user input
- Save user input
- Show and hide elements when the user interacts with the page
- etc.



**Enter
JavaScript!**

JavaScript

JavaScript

JavaScript is a programming language.

It is currently the only programming language that your browser can execute natively. (There are efforts to change that.)

Therefore if you want to make your web pages do stuff, you must use JavaScript: There are no other options.



JavaScript

- Created in 1995 by Brendan Eich
 - (co-founder of Mozilla; resigned 2014 due to his homophobia)
- JavaScript has nothing to do with Java
 - Literally named that way for marketing reasons
- The first version was written in 10 days
- Several fundamental language decisions were made because of company politics and not technical reasons

"I was under marketing orders to make it look like Java but not make it too big for its britches ... [it] needed to be a silly little brother language." (source)

JavaScript in the browser

Code in web pages

HTML can embed JavaScript files into the web page via the `<script>` tag.

```
<!DOCTYPE html>
<html>
  <head>
    <title>CS 193X</title>
    <link rel="stylesheet" href="style.css" />
    <script src="filename.js"></script>
  </head>
  <body>
    ... contents of the page...
  </body>
</html>
```

console.log

You can print log messages in JavaScript by calling `console.log()`:

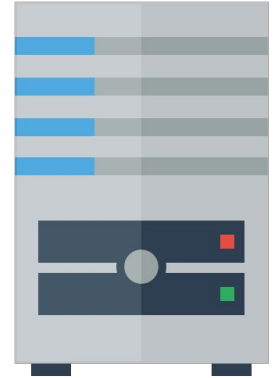
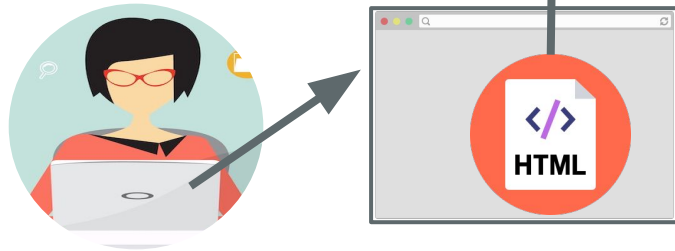
script.js

```
console.log('Hello, world!');
```

This JavaScript's equivalent of Java's `System.out.println`, `print`, `printf`, etc.

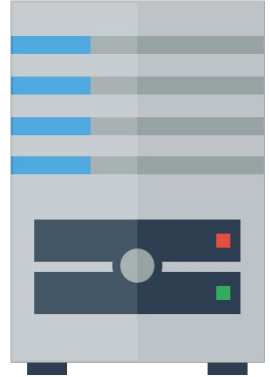
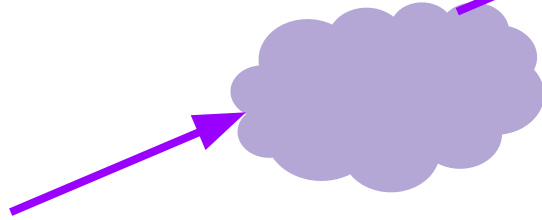
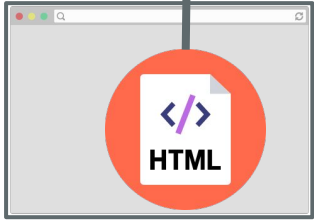
How does JavaScript get loaded?

```
<head>
  <title>CS 193X</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```



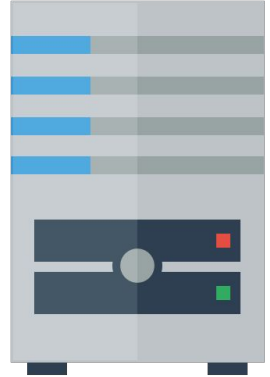
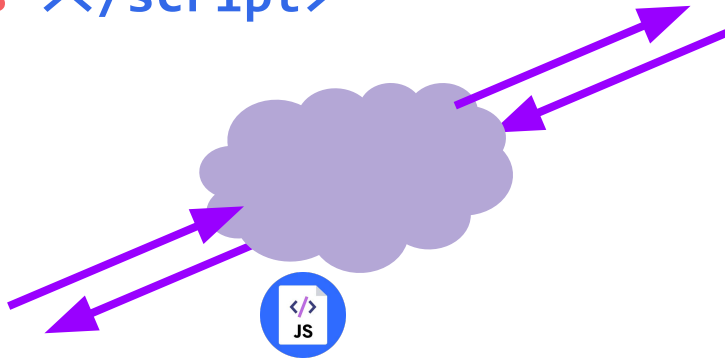
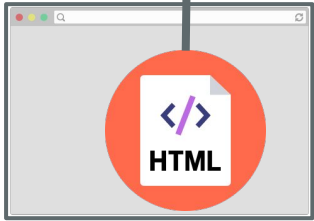
The browser is parsing the HTML file, and gets to a script tag, so it knows it needs to get the script file as well.

```
<head>
  <title>CS 193X</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```



The browser makes a request to the server for the script.js file, just like it would for a CSS file or an image...

```
<head>
  <title>CS 193X</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```



And the server responds with the JavaScript file, just like it would with a CSS file or an image...


```
<head>
  <title>CS 193X</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```



```
console.log('Hello, world!');
```

Now at this point, the JavaScript file will execute "**client-side**", or in the browser on the user's computer.

JavaScript execution

There is no "main method"

- The script file is executed from top to bottom.

There's **no compilation** by the developer

- JavaScript is compiled and executed on the fly by the browser

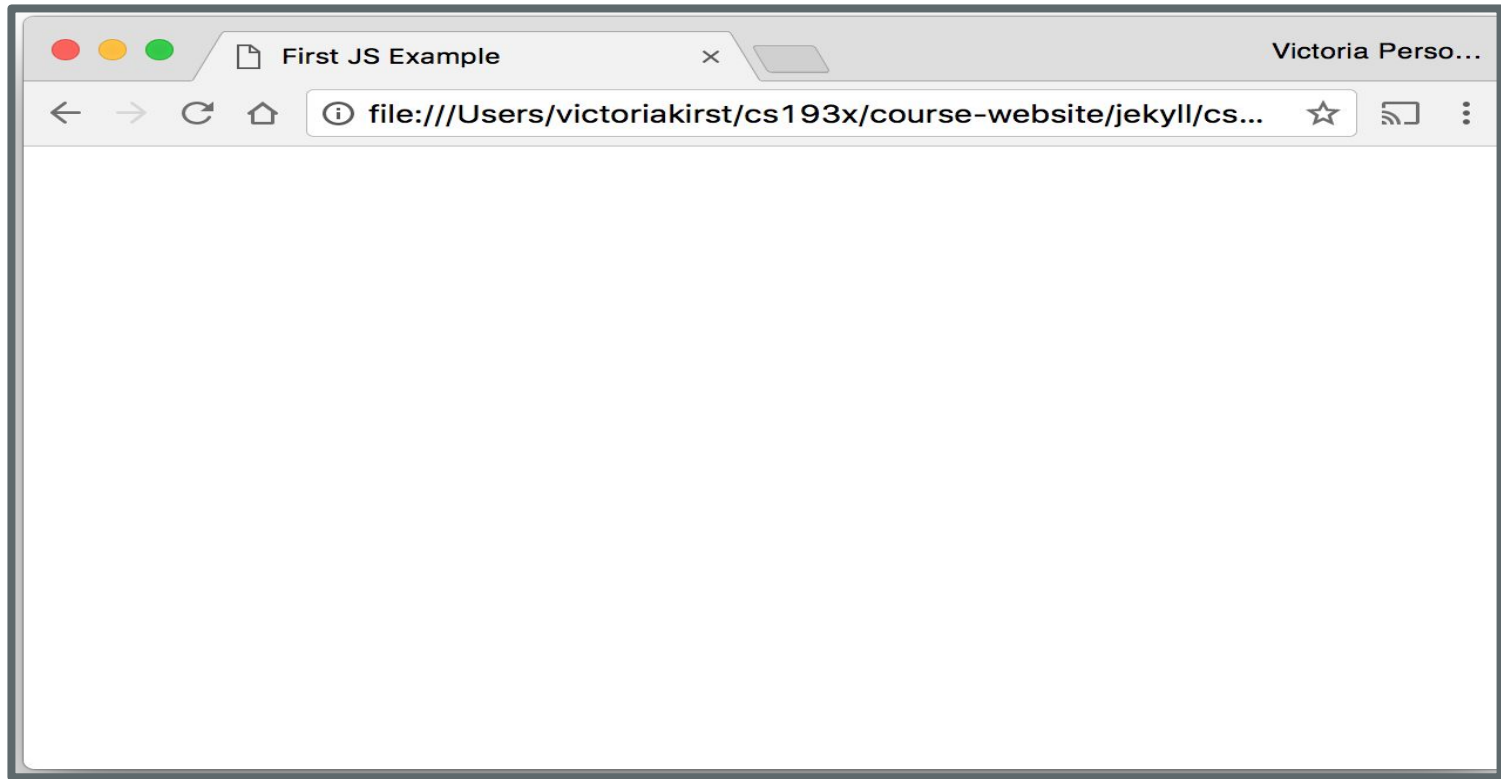
(Note that this is slightly different than being "interpreted": see [just-in-time \(JIT\) compilation](#))

first-js.html

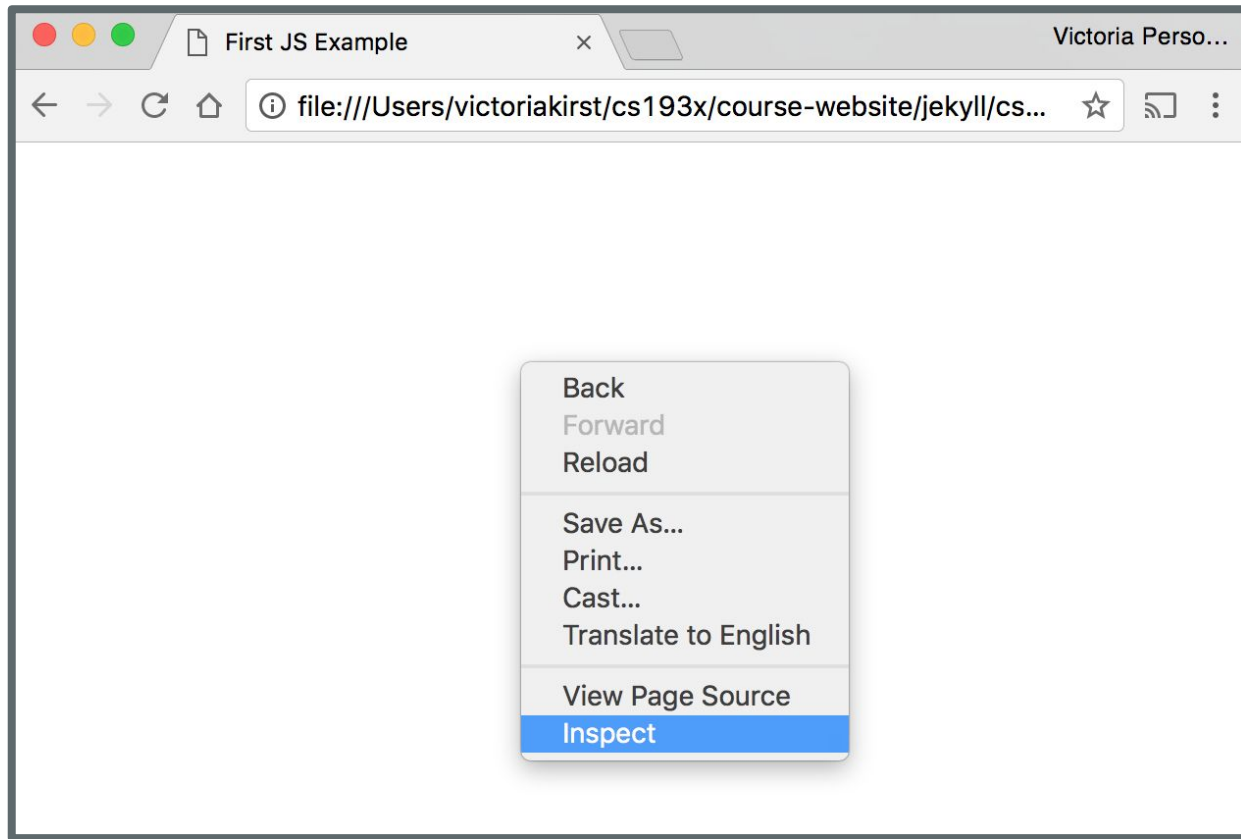
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>First JS Example</title>
    <script src="script.js"></script>
  </head>
  <body>
  </body>
</html>
```

script.js

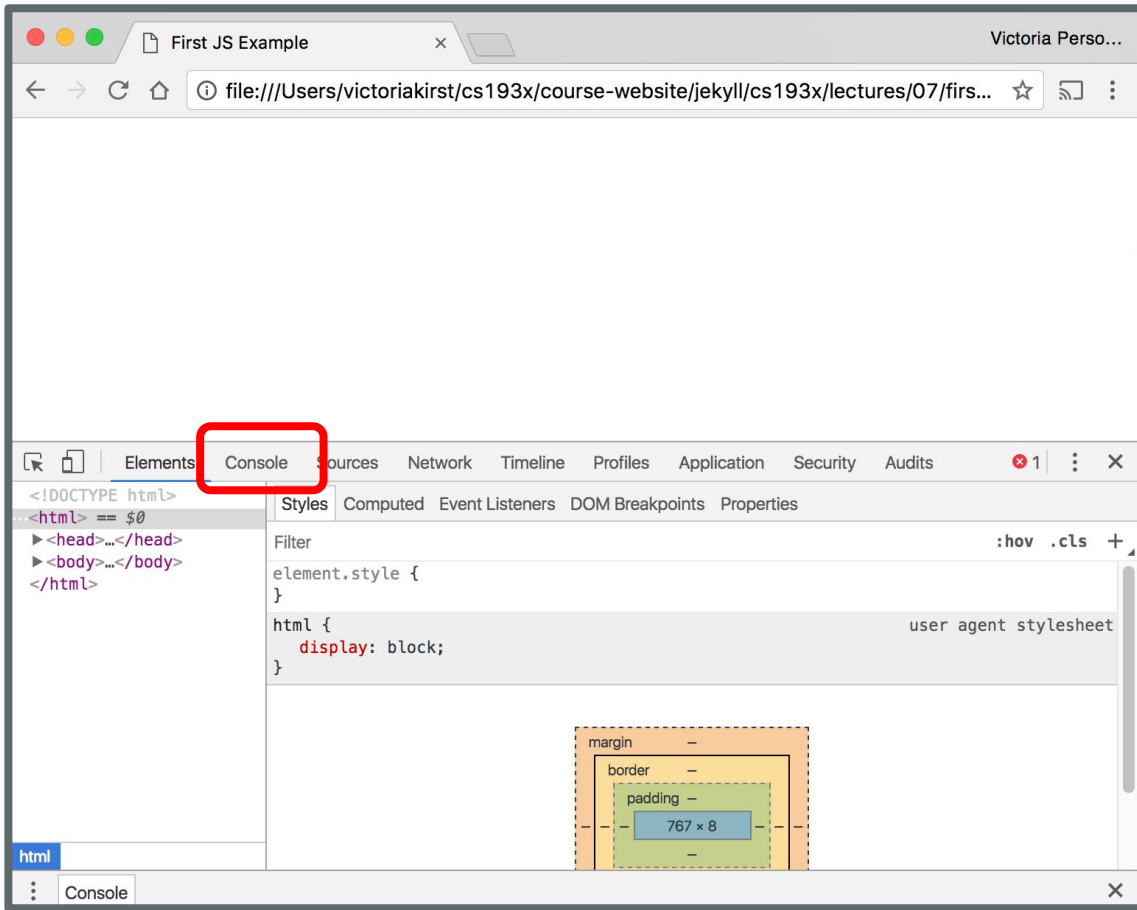
```
console.log('Hello, world!');
```



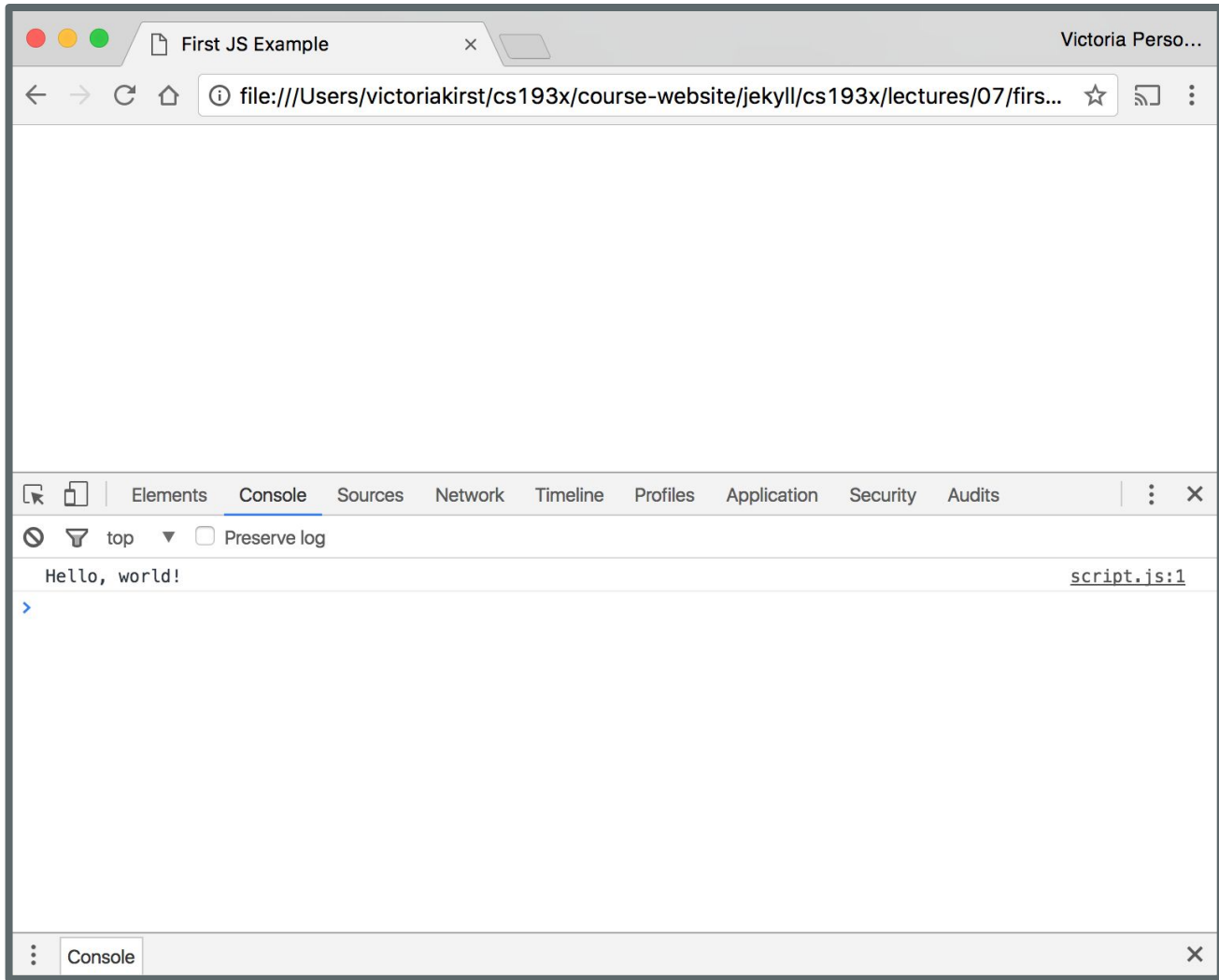
Hey, nothing happened!

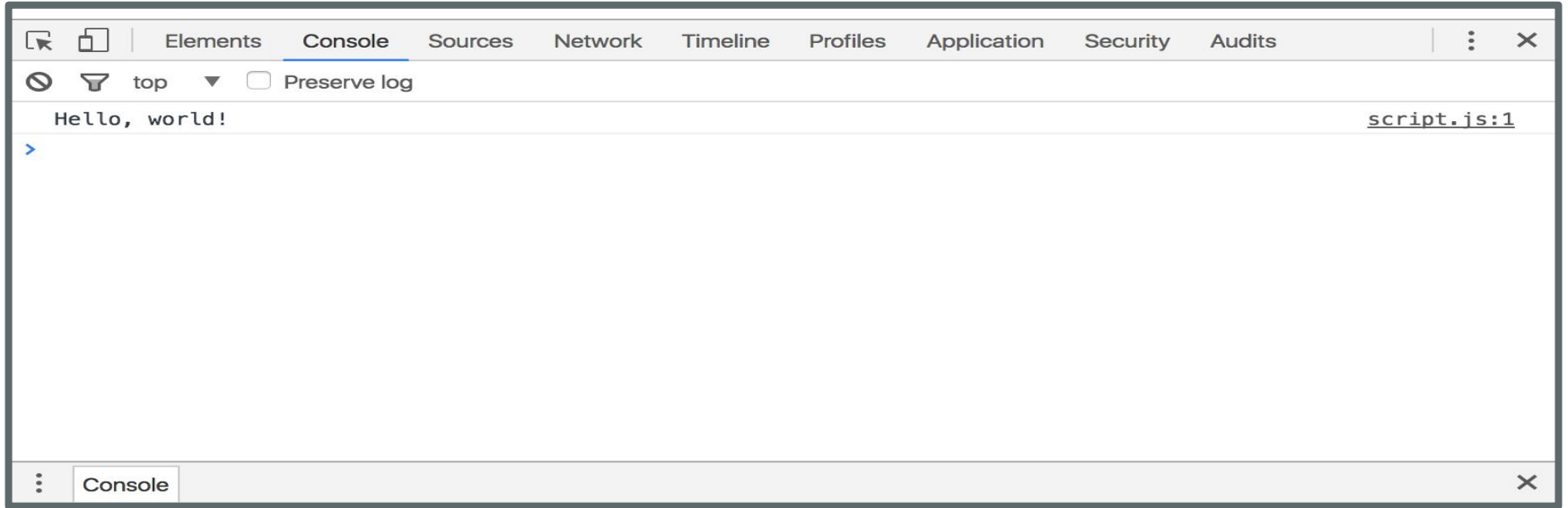


Right-click (or control-click on Mac) and choose "Inspect"



Click "Console" tab





The "Console" tab is also a REPL, or an interactive language shell, so you can type in JavaScript expressions, etc. to test out the language.

We will be using this throughout the quarter!

JavaScript language features

Same as Java/C++/C-style langs

for-loops:

```
for (let i = 0; i < 5; i++) { ... }
```

while-loops:

```
while (notFinished) { ... }
```

comments:

```
// comment or /* comment */
```

conditionals (if statements):

```
if (...) {  
    ...  
} else {  
    ...  
}
```

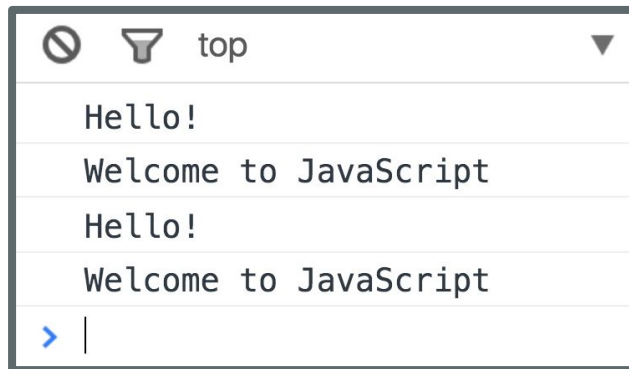
Functions

One way of defining a JavaScript function is with the following syntax:

```
function name() {  
    statement;  
    statement;  
    ...  
}
```

script.js

```
function hello() {  
  console.log('Hello!');  
  console.log('Welcome to JavaScript');  
}  
  
hello();  
hello();
```



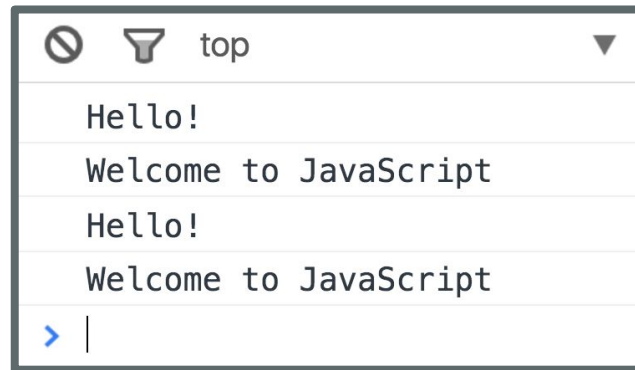
The screenshot shows a browser console window with a dark border. At the top, there is a filter icon (a circle with a slash), a funnel icon, and the text "top" followed by a downward-pointing triangle. Below this, there are four lines of output: "Hello!", "Welcome to JavaScript", "Hello!", and "Welcome to JavaScript". At the bottom of the console, there is a blue prompt character ">" followed by a vertical bar "|".

Console output

script.js

```
function hello() {  
  console.log('Hello!');  
  console.log('Welcome to JavaScript');  
}  
  
hello();  
hello();
```

The browser "executes" the function definition first, but that just creates the hello function (and it doesn't run the hello function), similar to a variable declaration.



Console output

script.js

```
hello();
```

```
hello();
```

```
function hello() {
```

```
    console.log('Hello!');
```

```
    console.log('Welcome to JavaScript');
```

```
}
```

Q: Does this work?

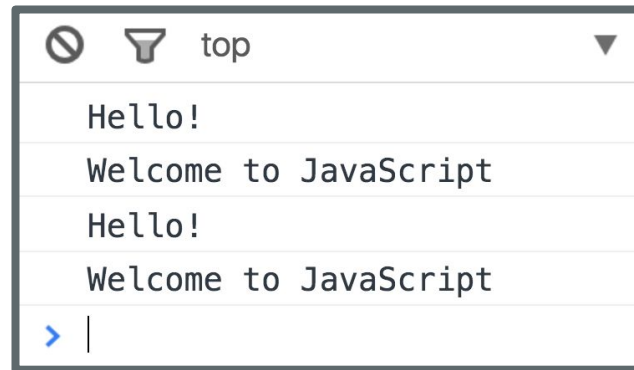
script.js

```
hello();  
hello();  
  
function hello() {  
  console.log('Hello!');  
  console.log('Welcome to JavaScript');  
}
```

A: Yes, for this particular syntax.

This works because function declarations are "hoisted" ([mdn](#)).

You can think of it as if the definition gets moved to the top of the scope in which it's defined (though that's not what actually happens).



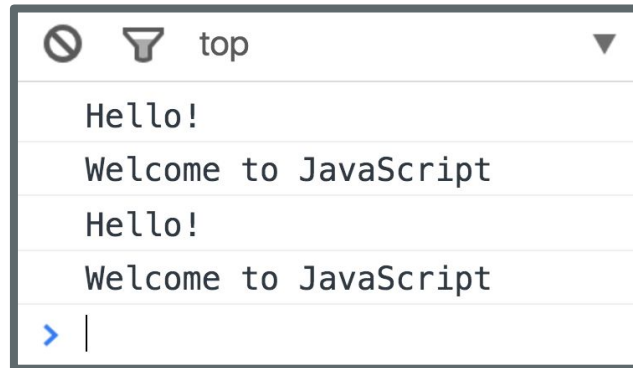
Console output

script.js

```
hello();  
hello();  
  
function hello() {  
  console.log('Hello!');  
  console.log('Welcome to JavaScript');  
}
```

Caveats:

- There are other ways to define functions that do not get hoisted;
- Try not to rely on hoisting when coding. It gets bad.



Console output