

# CSE 503

## Introduction to Computer Science for Non-Majors

Dr. Eric Mikida

epmikida@buffalo.edu

208 Capen Hall

**Day 20**

**HTML and Front End JavaScript**

# Recap

- Now we can also write to files...

# Writing Text Files

To write to a text file, open the file in write mode instead ("**w**").

The file object has a `write` function we can call, passing in the string we want to write:

```
with open("testfile.txt", "w") as f:  
    f.write("Text on the first line\n")  
    f.write("Text on the second line!\n")
```

# Writing Text Files

If we have multiple things to write, we can use a for loop (note the addition of the newline character `'\n'`):

```
def write(filename, contents):  
    with open(filename, "w") as f:  
        for item in contents:  
            f.write(item + '\n')
```

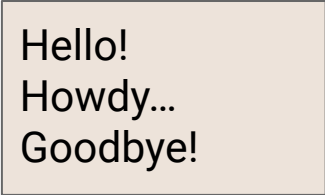
# Writing Text Files

If we have multiple things to write, we can use a for loop (note the addition of the newline character `'\n'`):

```
def write(filename, contents):  
    with open(filename, "w") as f:  
        for item in contents:  
            f.write(item + '\n')
```

```
write("test.txt", ["Hello!", "Howdy...", "Goodbye"])
```

test.txt



```
Hello!  
Howdy...  
Goodbye!
```

# Writing CSV Files

Writing CSV files uses the `csv.writer` object, which write records (sequences) to a single line, with each value separated by a comma.

```
import csv

def writeCSV(filename, dataTable):
    with open(filename, "w", newline='') as f:
        writer = csv.writer(f)
        for record in dataTable:
            writer.writerow(record)
```

# Writing CSV Files

```
import csv

def writeCSV(filename, dataTable):
    with open(filename, "w", newline='') as f:
        writer = csv.writer(f)
        for record in dataTable:
            writer.writerow(record)

dt = [ ['abc', 'def'] , ['ghij', 'klmn'] ]

writeCSV('file1.csv',dt)
writeCSV('file2.csv',dt[0])
```

*What will file1.csv and file2.csv look like?*

# Writing CSV Files

```
import csv

def writeCSV(filename, dataTable):
    with open(filename, "w", newline='') as f:
        writer = csv.writer(f)
        for record in dataTable:
            writer.writerow(record)
```

```
dt = [ ['abc', 'def'], ['ghij', 'klmn'] ]
```

```
writeCSV('file1.csv', dt)
writeCSV('file2.csv', dt[0])
```

**file1.csv**

```
abc,def
ghij,klmn
```



# Writing CSV Files

```
import csv
```

```
def writeCSV(filename, dataTable):  
    with open(filename, "w", newline='') as f:  
        writer = csv.writer(f)  
        for record in dataTable:  
            writer.writerow(record)
```

```
dt = [ ['abc', 'def'], ['ghij', 'klmn'] ]
```

```
writeCSV('file1.csv', dt)  
writeCSV('file2.csv', dt[0])
```

**file1.csv**

```
abc,def  
ghij,klmn
```

# Writing CSV Files

```
import csv

def writeCSV(filename, dataTable):
    with open(filename, "w", newline='') as f:
        writer = csv.writer(f)
        for record in dataTable:
            writer.writerow(record)

dt = [ ['abc', 'def'], ['ghij', 'klmn'] ]

writeCSV('file1.csv', dt)
writeCSV('file2.csv', dt[0])
```

## file1.csv

```
abc,def
ghij,klmn
```

## file2.csv

```
a,b,c
d,e,f
```

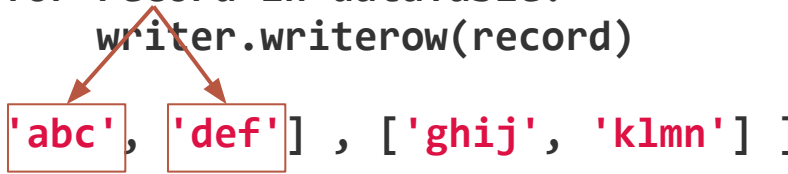
# Writing CSV Files

```
import csv

def writeCSV(filename, dataTable):
    with open(filename, "w", newline='') as f:
        writer = csv.writer(f)
        for record in dataTable:
            writer.writerow(record)

dt = [ [ 'abc', 'def' ], [ 'ghij', 'klmn' ] ]

writeCSV('file1.csv', dt)
writeCSV('file2.csv', dt[0])
```



## file1.csv

```
abc,def
ghij,klmn
```

## file2.csv

```
a,b,c
d,e,f
```

# Exercise

Define a function which, given the name of a csv file, reads data from that csv file. Each record in the file has fields  $f_0$  through  $f_N$ . Write a new csv file, whose file name is the same as the original but prefixed with 'R', which has the same records but with the fields reversed, from  $f_N$  through  $f_0$ .

# HTML

Hyper Text Markup Language

**Hyper Text:** Text that can contain links to other resources

**Markup Language:** Text that contains special markers. These markers add information to the text that is not displayed. In HTML we use *tags* that tell the browser how to display the text.

**HTML is not a programming language!**

# HTML

```
<html>
<head></head>
<body>
<h1>First Web Page</h1>
<p>My Content</p>
<div id="myDiv"></div>
</body>
</html>
```

← Save this in a file with .html as the file extension and open it in a browser. You will see the web page below

## First Web Page

My content

# HTML - Elements

```
<html>
<head></head>
<body>
<h1>First Web Page</h1>
<p>My Content</p>
<div id="myDiv"></div>
</body>
</html>
```

HTML uses angle brackets to define *elements*

# HTML - Elements

```
<html>
<head></head>
<body>
<h1>First Web Page</h1>
<p>My Content</p>
<div id="myDiv"></div>
</body>
</html>
```

HTML uses angle brackets to define *elements*

Each element has an open tag, and a close tag. ie `<h1>` and `</h1>`



# HTML - Elements

```
<html>
<head></head>
<body>
<h1>First Web Page</h1>
<p>My Content</p>
<div id="myDiv"></div>
</body>
</html>
```

HTML uses angle brackets to define *elements*

Each element has an open tag, and a close tag. ie `<h1>` and `</h1>`

Everything between the open and close tag is the content of that element.

# HTML - Elements

```
<html>
<head></head>
<body>
<h1>First Web Page</h1>
<p>My Content</p>
<div id="myDiv"></div>
</body>
</html>
```

HTML uses angle brackets to define *elements*

Each element has an open tag, and a close tag. ie `<h1>` and `</h1>`

Everything between the open and close tag is the content of that element.

In this example we used header 1 (h1) and paragraph (p) tags to display text with different sizes.

# HTML - Properties

```
<html>
<head></head>
<body>
<h1>First Web Page</h1>
<p>My Content</p>
<div id="myDiv"></div>
</body>
</html>
```

Elements can contain *properties* which are defined in the open tag of the element

# HTML - Properties

```
<html>
<head></head>
<body>
<h1>First Web Page</h1>
<p>My Content</p>
<div id="myDiv"></div>
</body>
</html>
```

Elements can contain ***properties*** which are defined in the open tag of the element

These properties are key-value pairs

# HTML - Properties

```
<html>
<head></head>
<body>
<h1>First Web Page</h1>
<p>My Content</p>
<div id="myDiv"></div>
</body>
</html>
```

Elements can contain *properties* which are defined in the open tag of the element

These properties are key-value pairs

In this example we have an empty division with a property named `id` that has a value of `"myDiv"`

# HTML

## HTML is not a programming language

This is as much as we cover in this course.

For much, much, much more information about HTML and other web technologies, visit w3schools: <https://www.w3schools.com>

# Front End JavaScript

```
<html>
<head></head>
<body>
<h1>First Web Page</h1>
<p>My Content</p>
<div id="myDiv"></div>
</body>
</html>
```

Instead of learning more HTML, we will instead write **JavaScript** to add more power to our web pages.

# Front End JavaScript

```
<html>
<head></head>
<body>
<h1>First Web Page</h1>
<p>My Content</p>
<div id="myDiv"></div>
<script src="myCode.js"></script>
</body>
</html>
```

Instead of learning more HTML, we will instead write **JavaScript** to add more power to our web pages.

We can "import" our JavaScript code by adding a script element with a src property containing our JS filename.



# Front End JavaScript

```
<html>
<head></head>
<body>
<h1>First Web Page</h1>
<p>My Content</p>
<div id="myDiv"></div>
<script src="myCode.js"></script>
</body>
</html>
```

Instead of learning more HTML, we will instead write **JavaScript** to add more power to our web pages.

We can "import" our JavaScript code by adding a script element with a src property containing our JS filename.

This runs our script once the body is loaded.

# Front End JavaScript

Save the following code in a file named "myCode.js" and it will run once the content of our HTML page is loaded.

```
let myDiv = document.getElementById("myDiv");  
myDiv.innerHTML = "Content added from JavaScript";
```

# Front End JavaScript

Save the following code in a file named "myCode.js" and it will run once the content of our HTML page is loaded.

```
let myDiv = document.getElementById("myDiv");  
myDiv.innerHTML = "Content added from JavaScript";
```

Here we call the document.getElementById method with the id of our div element as the argument.

# Front End JavaScript

Save the following code in a file named "myCode.js" and it will run once the content of our HTML page is loaded.

```
let myDiv = document.getElementById("myDiv");  
myDiv.innerHTML = "Content added from JavaScript";
```

Here we call the document.getElementById method with the id of our div element as the argument.

The element is an object with a key "innerHTML" whose value is the content of the element.

# Front End JavaScript

Now when the body loads and our script runs, our page looks like this:

**First Web Page**

My content

Content added from JavaScript

# Front End JavaScript

More information for the Document Object Model (DOM) can be found at w3schools: [https://www.w3schools.com/js/js\\_htmlDOM.asp](https://www.w3schools.com/js/js_htmlDOM.asp)

# JavaScript Libraries

We can also download external libraries using the script tag in the same way we downloaded our own code:

```
<html>  
<head>  
<script src="https://momentjs.com/downloads/moment.js"></script>  
</head>  
...
```

# JavaScript Libraries

We can also download external libraries using the script tag in the same way we downloaded our own code:

```
<html>  
<head>  
<script src="https://momentjs.com/downloads/moment.js"></script>  
</head>  
...
```

We can put the script element in the head, since we aren't displaying any HTML elements...just downloading the library



# JavaScript Libraries

When the page is loaded, there will be a request to automatically download the library

```
<script src="https://momentjs.com/downloads/moment.js"></script>
```

The library is simply a JavaScript file that someone else has written and shared with us for free. This is called **open-source** since everyone has access to the source code

Source: <https://github.com/moment/moment/>

# JavaScript Libraries

*So...what did we just download? We'll find out next lecture...*