HTML = NOUNS

CSS = ADJECTIVES

HTML

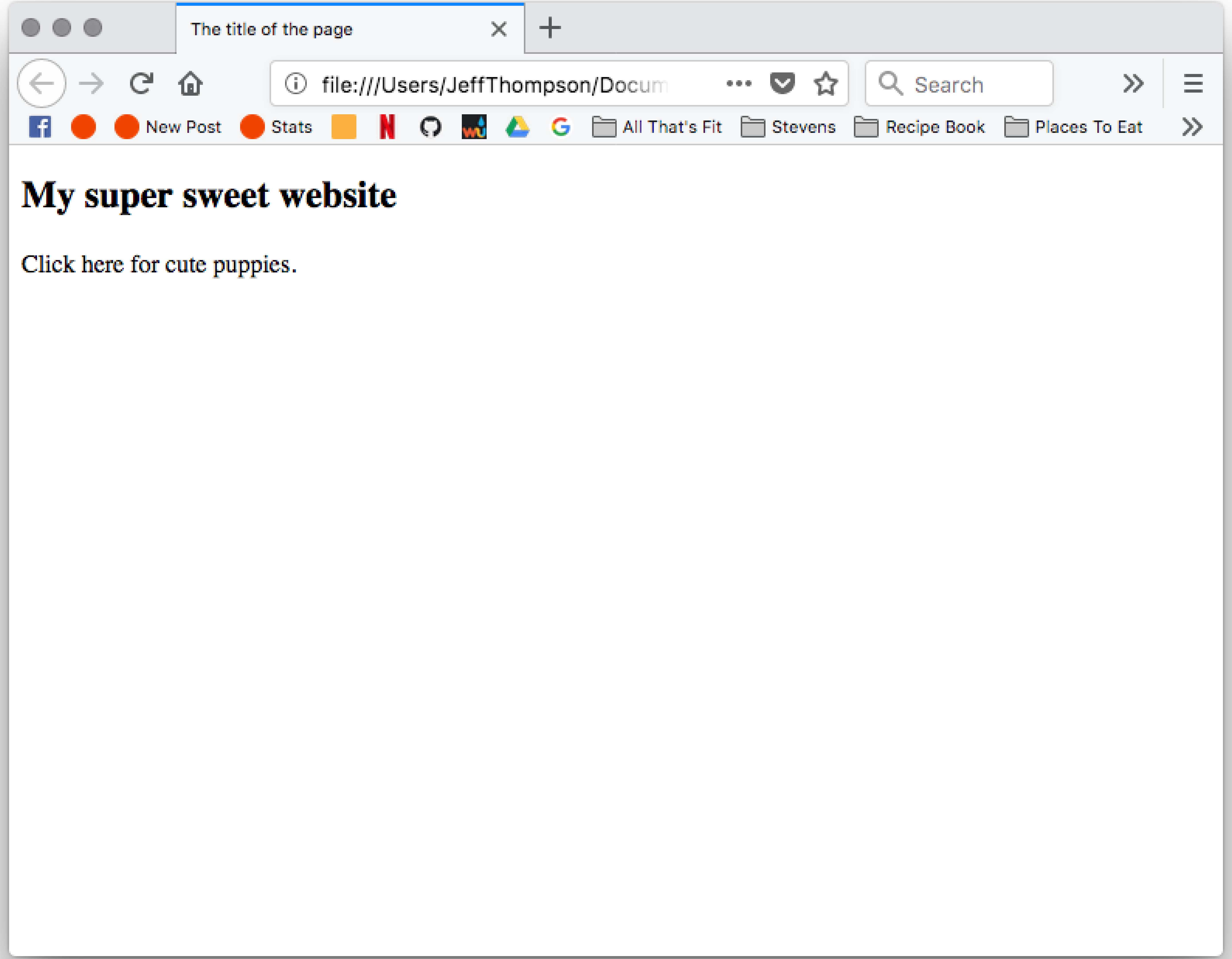
Hypertext Markup Language
Human-readable format based on XML

Describes the *things* on a page

- **The page itself, paragraphs, images, sections, links, etc**

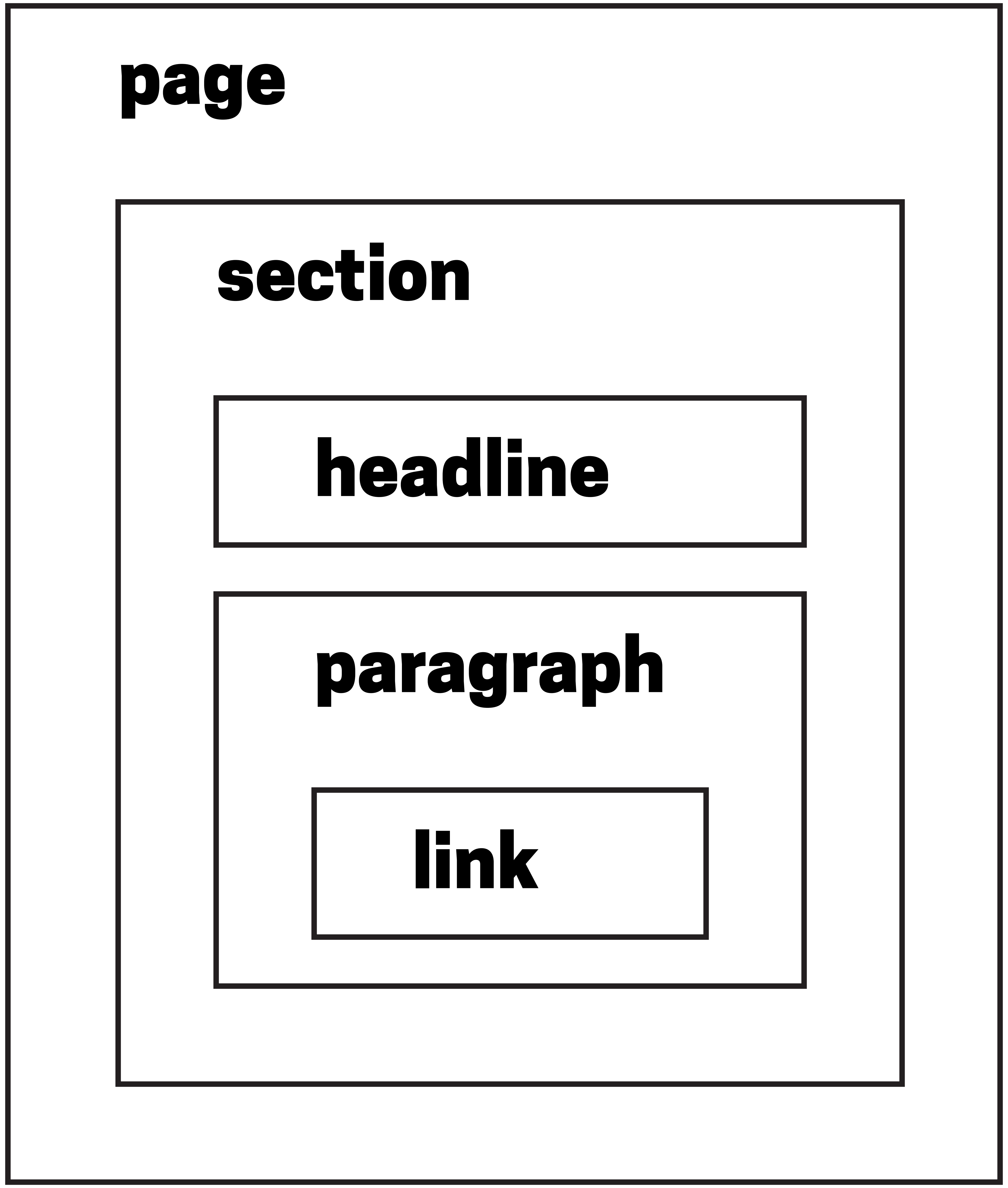
Things can be nested inside each other

- **A page contains paragraphs**
- **Paragraphs contain links**



My super sweet website

Click here for cute puppies.



page

section

headline

paragraph

link

```
<html>
<head>
    <title>The title of the page</title>
</head>

<body>
    <section>
        <h1>My super sweet website</h1>
        <p>Click here for <a>cute
        puppies</a>.</p>
    </section>
</body>
</html>
```

everything is in the <html> tag
<head> is metadata about the page
...like the title
note that content is enclosed in tags

<body> is the stuff that we see
a section of content inside the body...
...which includes headlines
...and paragraphs
...which contain links

everything gets a closing tag

CSS

Cascading Style Sheets

Defines how things (HTML tags) look

- Font size, background color, position, etc

Can address things three different ways

- All tags of the same kind (paragraphs, links)
- A tag with a unique ID
- All tags marked with the same “class” name

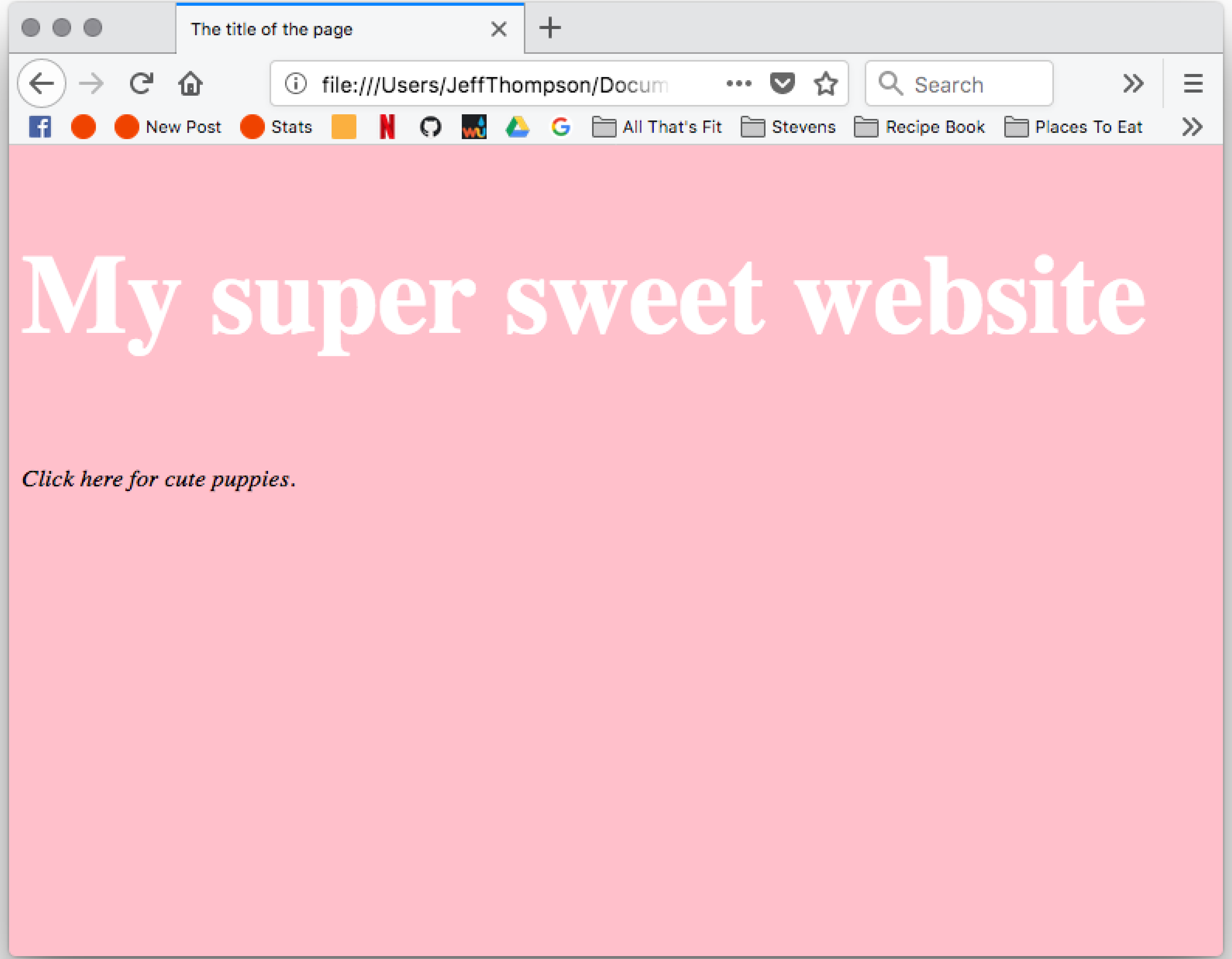
```
<html>
<head>
	<title>The title of the page</title>
</head>

<body>
	<section>
		<h1 id=”mainTitle”>My super
		sweet website</h1>
		<p class=”puppyInfo”>Click here
		 for <a>cute puppies</a>.</p>
	</section>
</body>
</html>
```

```
/* things applied to the entire page */
body {
	background-color: pink;
}

/* one item designated with an ID */
#mainTitle {
	font-size: 72px;
	color: white;
}

/* multiple items sharing the same class */
.puppyInfo {
	font-style: italic;
}
```



My super sweet website

Click here for cute puppies.

DOM

Document Object Model

Refers to all the things (HTML tags) in the page

- Since HTML is nested, objects have “parents” and “children”
- (We’ll see later how to search for things in the DOM)

Can be accessed by CSS

...or programmatically by Javascript!

JAVASCRIPT

(not related to the Java language*)

Developed in the early 1990s
94.5% of the 10-million most popular sites use JS

A client-side scripting language

- Code runs on your machine, not a server
- Cannot write or change files (for security reasons)

* Ok, they are slightly related. Java was the inspiration for some of the syntax, but Javascript is also inspired by languages like Scheme.

SYNC/ASYNC

There are two ways that code can be executed.

Eating out is a good metaphor:

- Syncronous: getting a burrito. You stand in line while everyone ahead of your gets served. Items get added one at a time to your food, too.
- Asyncronous: going to a restaurant. Food is ordered, cooked, and served continuously. You get your meal as soon as it is ready, regardless of when your order was placed. (A salad is quicker than a shepherds pie.)

SYNCRONOUS CODE

Code that runs one instruction at a time
Waits for the previous command to finish before continuing

All the code we wrote in Creative Programming 1 was syncronous:

- Processing draws a rectangle, then draws a square
- Python creates your tweet, then sends it

Makes intuitive sense, but code stops during long tasks

ASYNCRONOUS CODE

Code spawns tasks that may finish later without “blocking” other code

Javascript is a mix of syncronous and asyncronous

- **Adding two numbers or selecting items from the DOM are things that happen syncronously**
- **Loading a data file or animation events happen asyncronously, keeping your browser free to be interacted with**

THINKING ASYNCRONOUSLY

Instead of thinking about step-by-step instructions, async code requires thinking in terms of events

- An event can happen at any time, and needs to be wrapped in a function
- We can’t count on the output of code until it is finished
- When finished, an event may trigger other events in a cascade

It can be very confusing to plan code like this!