Lesson 0 notes

In part 1 you will develop enough [**technological empathy**](https://www.udacity.com/course/viewer#!/c-ud440/l-3521029360/m-3503049573) to understand what happens when you open (or create) a web page. Specifically, you'll gain:

* Basic understanding of how the internet works.
* Understanding of how **users, computers, servers, http,** and **the internet** all fit together.

**Part 2**

In part 2 you will use **HTML tags** to create a simple **HTML document** that a browser can display as a web page.

Note that the first video is Steve introducing the entire course. You won't actually take the whole course in this Nanodegree, but it's still worth learning what his course covers in case you decide you want to take the whole thing later.

**1. Procedural Thinking**

Procedural thinking involves creating perfectly clear and unambiguous instructions for a computer to follow.

It may be easy to tell a person how to sort a deck of cards, but getting a computer to do that requires more thought. But once you've learned to think this way you'll find that computers can do it **much** faster.

**2. Abstract Thinking**

Abstract thinking means finding similarity, or as programmers would say, **generality** amongst seemingly different things.

In this Nanodegree you will visit hundreds of Udacity web pages. It would be impractical for a programmer to specifically program each of these pages individually. The Udacity programmers used the power of what's called *abstraction* to avoid unnecessary repetition of work.

**3. Systems Thinking**

Systems thinking happens when you break a big problem down into smaller pieces. Programmers do this when they create a plan (often on paper) for how a program will work. It involves big-picture thinking and decision-making about a problem and how different pieces of a program can work together to solve it. If this is unclear now, don't worry! You'll understand what this means much more clearly by the end of the Nanodegree.

**4. Technological Empathy**

"Technological empathy" comes in many forms. For example, computer empathy is the ability to understand what a computer is, how it works, and what it's good and bad at doing.

A computer is a tool, as are the programs we use and the programming languages used to write them. It's almost impossible to program anything substantial without a basic understanding of how these tools work.

**5. Debugging**

Debugging is a systematic process of relentlessly identifying the cause of a computer program that doesn't work.

When a program doesn't work, it's because there is a mistake (also known as a "bug") somewhere in the computer code. Since these mistakes are an unavoidable part of programming, every good programmer has a system for fixing them and that system usually looks something like this:

1. Collect evidence (what makes this program fail?)
2. Generate theories (what may have caused this problem?)
3. Test those theories (if my theory is correct, how could I find out?)
4. Fix the problem

html hypertext markup language

hyper links= links for short

invented in 1990s

browser-the internet-http-server ( back again)

SERVER-COMPUTERS THAT HOST THE FILES THAT MAKE UP THE INTERNET

HTML

TEXT CONTENT WHAT YOU SEE

MARKUP-WHAT IT LOOKS LIKE

REFERENCES TO IMAGES/VIDEOS ETC

LINKS TO OTHER PAGES ETC

html markup

(name) contents (name)

opening closing

named element

b for bold

emphasis (italic)

<em> </em>

programmers often describe computers as "stupid."

When they say this, they (usually) aren't insulting computers. They are addressing an important point about how computers understand the instructions that human programmers give them.

Keep this idea in mind as you watch the next video and answer the next question. Think about how sensitive computers are to typos and how small mistakes (mistakes that a "smart" human could easily correct) can lead to huge problems when given to a "stupid" computer.

Steve is about to show you how to connect the HTML documents you write to *any other document on the internet* by adding links! This is an incredible amount of power!

If you're following along in [scratchpad.io](http://scratchpad.io/), be warned: the links you make may look correct but not actually work. If that happens, don't worry. You didn't do anything wrong, this is just one of the limitations of scratchpad.

html attributes

tag ATTR= attribute=value

tags can have multiple attributes

anchor = a tag

<a href(attribute) url-tag content tag

< a href=“udacity.com”> my favorite </a> ( links tag)

img tag for images

<img src ( source) =“url” alt(alternate)=“text”(required)>

images do not have closing tags there is no content aka a void tag

** It's OK to Forget**

You may be wondering why anchor tags use the href attribute while img tags use the src attribute. And you may become frustrated when (not if) you make the mistake of using one when you really need the other. Because when you make this mistake, your HTML will **not** work.

So how will you ever remember details like this?

Well... you don't have to!

In fact, you may have *already* forgotten how to add links to HTML. That's fine. Unless you're writing HTML all the time, details like these aren't worth remembering. A Google search for "adding links in html" will always get you your answer.

New programmers often struggle with this because in other learning environments it's often important that you remember a lot of facts.

But programming doesn't work like that! There's just too many facts to remember them all.

Instead, you need to understand a few big ideas and know how to get the details you need when you need them.

So continue with this lesson and don't worry about remembering everything. At the end of the lesson, we'll do a recap of what **is** important.

white space-even though text on multiple lines still on same name doesn’t matter if its under over or whereever

<br> break ( void)

this text is really<br>

<br> for breaking up lines or sentences

<p> paragraph is not a void tag because the p tag has content

therefore <p> content </p>

inline / block

<br>- inline (actually ending a line)

text<br> text

<p>-block <p> text </-> text

text

<b> <em> <img> all inline links

<p> is the only block

n the previous video Steve explained the distinction between **inline** and **block**elements. It's not essential that you fully understand the distinction at this point.

What he said about block elements creating "invisible boxes" **will** be important, but you will learn more about this in the next lesson.

In the next video you will have your first encounter with **container elements**: elements that do nothing but contain stuff inside of them. The purpose of these elements will not be clear yet, but in the next lesson you will see how these elements can be used to create web pages with complex structures.

css language

<span class =“foo"> text inline

<div class= “bar"> text block (behaves like paragraph tag)

That's right! a, span, br, img, and strong are all inline elements while div, p, and form are block elements.

irst, recall the two main topics covered in this lesson.

1. **Overview of the Internet**: You were shown the big-picture view of how the internet works and were introduced to terms like **servers**, **browsers**, **the internet**, and **HTTP**.
2. **Introduction to HTML**: You were introduced to HTML tags (like <b>, <p>, and <em>) and saw how even though these tags aren't visible to users of a web page, they still carry meaning that *is* meaningful to web browsers.

These topics were covered quickly and it's okay if you forget a lot of the details. But there are a few points which you should keep with you.

**What to take away from this lesson.**

The five most important things to take from this lesson are the following:

**1 - What a Web Page is**

A web page is a text document written in a language called HTML. Web browsers read these documents, and then interpret and display them.

**2 - How Coding Works**

Coding happens when programmers write text in a language that a computer can understand. The computer can then follow the instructions the programmer wrote. For example, the computer might do this by making text like this:

I'm <b>learning</b> to code!

look like this:

I'm **learning** to code!

**3 - Computers are Stupid**

Programmers need to write *exactly* the way a computer understands (also known as writing with correct "syntax").

For example, if you forget to close a <b> tag, the computer won't be able to figure out what you had intended to make bold. This "stupidity" can be very frustrating, but it also gives programmers incredible power: if you know how to talk to a computer than you can tell it do do anything you want.

**4 - Programmers Can't Remember Everything**

There are too many details to keep everything in your head. And that's okay. If you forget how to make text italic in HTML, you can always just look it up.

**5 - Basic HTML Vocabulary**

You will be using HTML in the next few lessons, so it will be helpful if you're comfortable with the jargon.

* **Tag**: An HTML tag is always contained within angled brackets. Most tags have an opening tag (<p> for example) and a closing tag, (</p>). Some tags (called "void" tags) do not require a closing tag (like the <br> tag).
* **Element**: An HTML element refers to everything within a set of opening and closing tags.
* **Attribute**: This is a property of an HTML element. For example, to set the href attribute of an anchor tag to the Udacity URL, you would write <a href="www.udacity.com”> html document <!Doctype HTML>- Doctype  <html> <html> <head> (meta-data) javascript) <title>content title</title> </head> <body> <b>content</b> </body> </html>