**1. Learn to code**

**1.1 Welcome to the Nanodegree**

**1.2 Nanodegree Qrientation**

**1.3 The World Wide Web**

**1.4 HTML Basics**

**1.5 PROJECT Making Your First Project**

**1.1 Welcome to the Nanodegree**

**1.1.1 Let’s Get Started**

This course is programming nanodegree. In this course, you can learn how to code. The best way to experience coding is by making your very first webpage.

**1.1.2 Setting up**

**Setting up**

What does building a web page entail? Well, you’ll first be writing code in HTML — or HyperText Markup Language — and then visualizing the resulting web page as it’s rendered by a web browser.

To get started on your own computer, you'll need two tools going forward:

* some place to write and save your code, and
* a web browser to view the rendered result.

**A Place to Write and Save Your Code — Installing a Text Editor**

You'll be writing and saving your code with something called a text editor. A text editor is just a piece of software that allows programmers to compose, edit, and save code — just as we might use “Microsoft Word” to edit text in natural language.

There are numerous text editors available for free download online, but most of them offer similar functionality. For example, one useful feature that’s common across editors is what we refer to as syntax highlighting, or the use of colors to format code and make it more readable.

Here are some examples of commonly used text editors:

Atom (recommended)

Sublime Text

Brackets

Any of these will work for our purposes, but we recommend installing Atom.

**1.1.3 Building your first Web Page**

**Writing and Saving Your Code**

With your text editor installed, you’re now ready to open it up and start writing code. Here, you’ll be writing in HTML, and that means you’ll need to save your code as an HTML file. Note that this is an important point! In order for a browser to be able to interpret your HTML code and render it as a web page, the file name must end with the appropriate extension of .html. That’s why we use a text editor (instead of a word processor) to write code — it allows us to save files in a particular format that suits our needs.

**Viewing the Rendered Web Page**

It’s now time to view the rendered web page in a browser like Chrome, Safari, or Firefox.

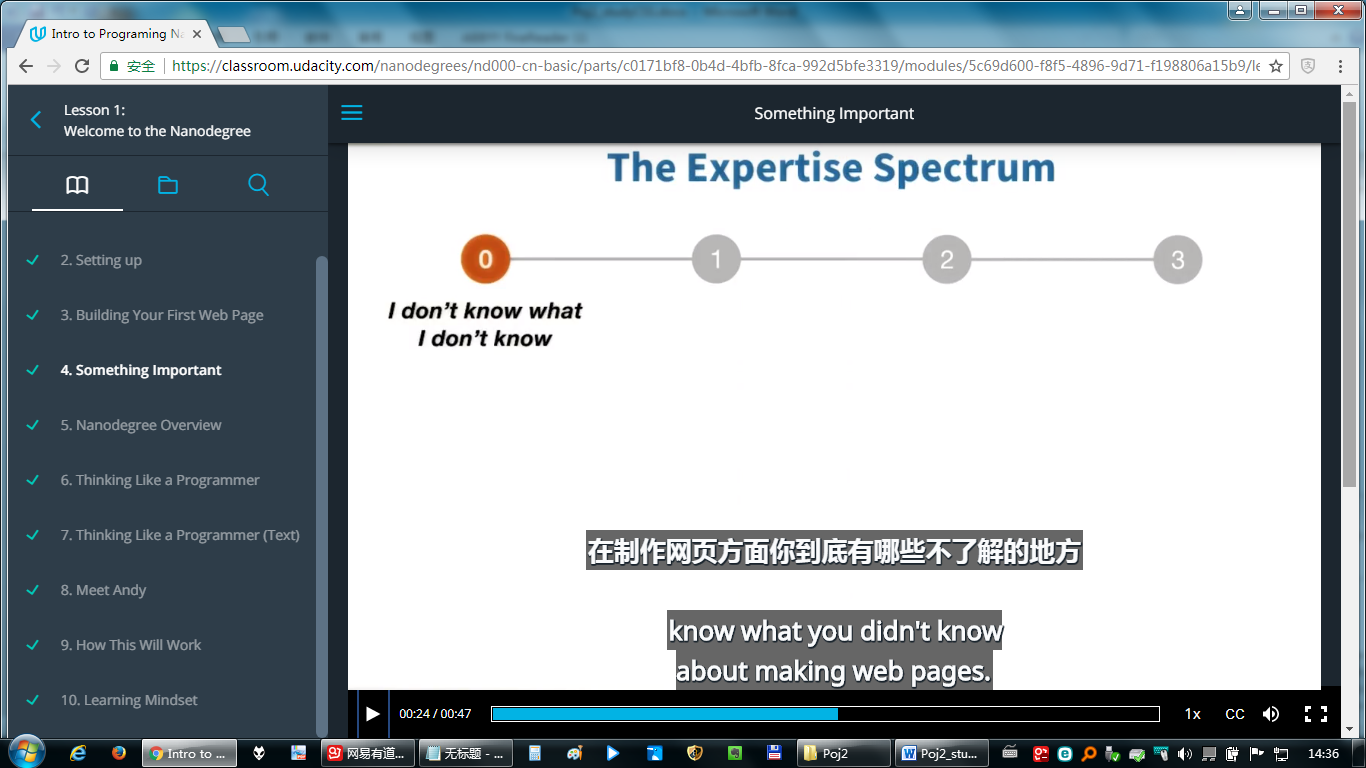
**Note:** If you make changes to your HTML code using your text editor, those changes will only be reflected in your browser’s rendered web page if you do two things:

1.You must first save your changes to the HTML file (in your text editor, File → Save).

2.You must then refresh the web page to render the modified HTML file.

**1.1.4 Something Important**

The Expertise Spectrum





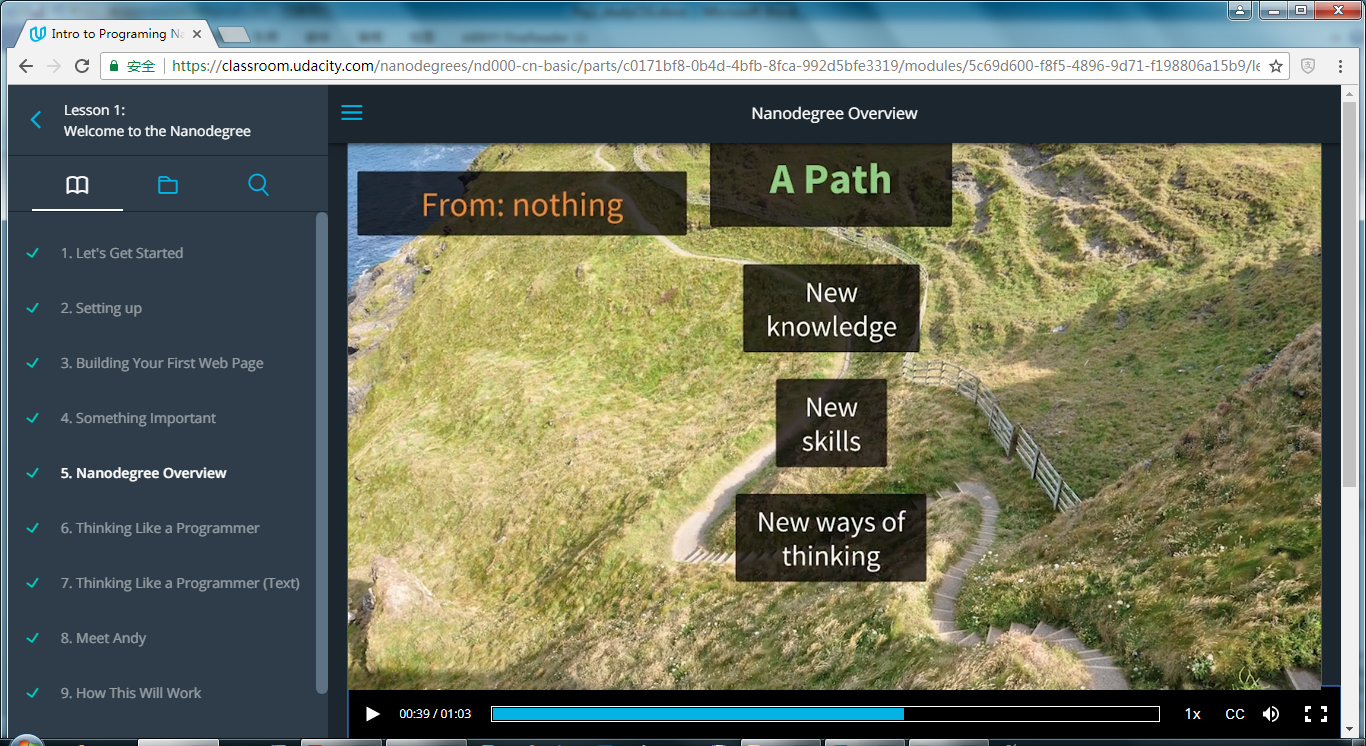
Computer has its own way to interpret things.

Coding a computer is just telling a computer what you want it to do, by writing text, in a way it can understand.

**1.1.5 Nanodegree Overview**

**What is Nanodegree?**

Nanodegree is a path starts with nothing. As you progress you're going to gain three important things:

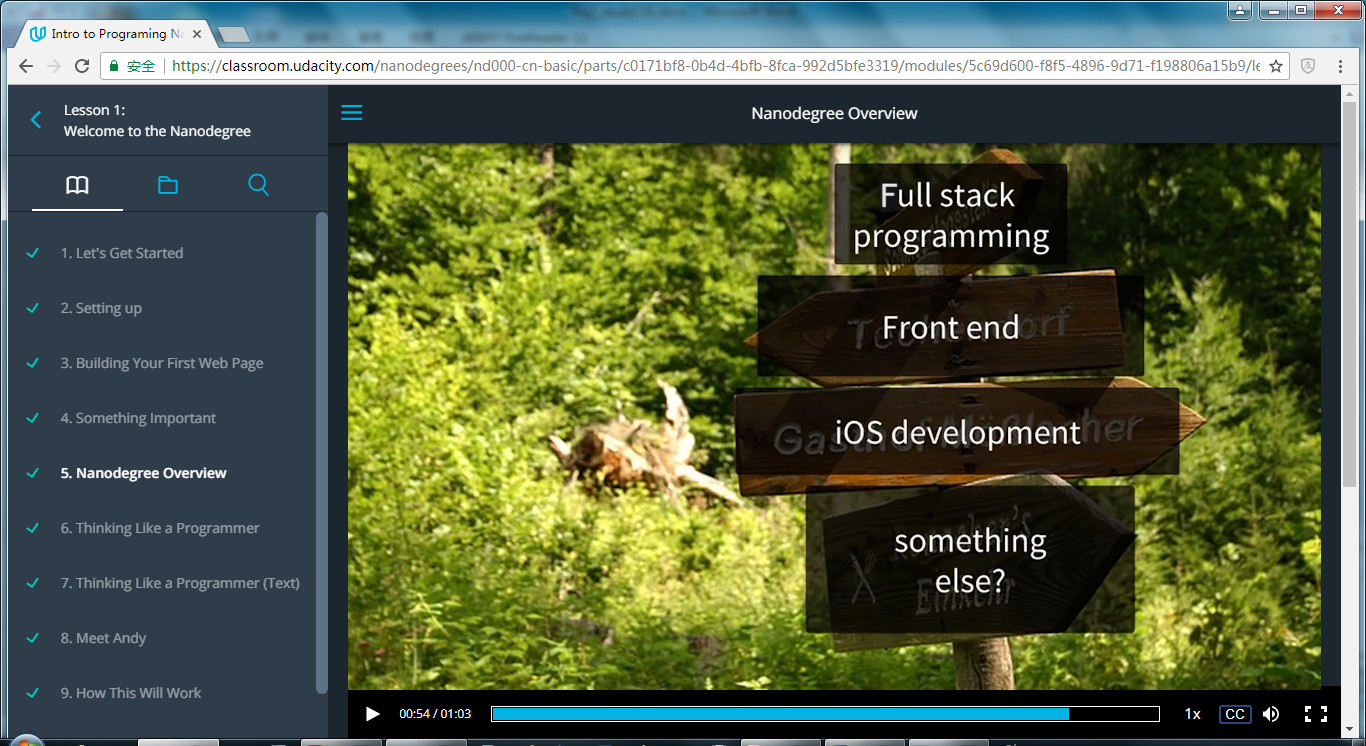


New Knowledge

New Skills

New way of thinking

What you can do **NEXT**?



At the end of it all, you'll get to a point where you can make a choice. You'll understand your options and be able to evaluate what it is you want to do next.

**1.1.6 Thinking Like a Programmer**

A large goal of this Nanodegree is to give you practice thinking like a computer programmer. Longtime programmers will often talk about how the programming thought process has helped them in other areas of their life, but what does it mean to "think like a programmer?" In this Nanodegree we will focus on five specific ways of thinking that all programmers have to use all the time. We will come back to these ideas repeatedly throughout the Nanodegree.

**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.

**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.

**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.

**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.

**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:

Collect evidence (what makes this program fail?)

Generate theories (what may have caused this problem?)

Test those theories (if my theory is correct, how could I find out?)

Fix the problem