[Windows Users] Installation and Setup for Web3 Development (Updated 2024):

Steps to install and set up for Internet computer development initially written by Ms. Angela Yu in "The Complete 2024 Web Development Bootcamp"- Udemy on 2022 (Currently - Lesson 327):

1. Make sure that we've got the correct system requirements. So if you go into the start section and search for system information, then you can see what the OS version is you're running.
   - So at a minimum, you should be running Windows 10, version 2004 or above, and you can tell that that's correct version by looking at this version and seeing that it's **19041** or above. And also, if you're running Windows 11 or above, that's totally fine.
   - The second thing to check is to make sure that you're running **64 bit** Windows. So under system type you should see **x64**.

2. Find the **Windows PowerShell** in your Start menu and run it as the Administrator.
   Type this line (`wsl --install`) inside the Windows PowerShell prompt. It's going to install something called the **Windows sub machine for Linux**. Basically, it's going to give us a virtual Linux to work with allowing us to run up bash commands, which are required when we're working with the Dfinity internet computer. **(You need to make sure that Virtualization is enabled in your PC BIOS)**
   So this is going to take a little while to run, but once it's done, it'll tell you the request operation is successful, but you have to restart your machine for it to take effect. So go ahead and restart your computer.
   And once it has restarted and launched again, it should automatically bring up this pane where you need to set up a username and password for working with **Ubuntu**.
   So go ahead and enter those pieces of information.
   But remember when you type your password, it's not going to show up on screen. So just be sure you know what you're typing and keep it as simple (**1234**) as possible to be able to remember it because you're going to use frequently very shortly when we're installing other components.

3. Once that is complete, go ahead and open up Windows PowerShell as the administrator again. And run the following command by typing in and click Enter: `wsl --list --verbose`
   And this basically just check to make sure that wsl was correctly installed. If everything is OK, you should see the following two lines:
   NAME STATE VERSION
   *** Ubuntu Running 2**
   Then you're ready to go to the next step, which is to install Visual Studio code.

4. Go to the following link to download the latest stable version of VSCode:
   https://code.visualstudio.com/
   Once the download completed, run the downloaded setup file, go through the installation wizard, accept the agreement, choose all of the defaults, and make sure that you add a desktop icon so you can access Visual Studio code quickly. You can also pin it to the Task Bar for quick access.

5. **Install some required extensions to VSCode.**
   - Go to the following link to install the **Motoko** extension:
     https://marketplace.visualstudio.com/items?itemName=dfinity-foundation.vscode-Motoko
     Click on install in your browser, and then it should be able to bring up Visual Studio code and allow you to open that link inside VSCode. Now the reason because there are quite a few extensions called **Motoko**, but we want the one that's from the Dfinity Foundation.
   - And then install the remote **wsl** extension to allow you to use the terminal inside VSCode and tap into that wsl that you installed earlier on:
     https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl

6. **Install Homebrew**

Open up **Ubuntu** from the start menu, copy that following command and paste it inside Ubuntu prompt then hit Enter to run it:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

During the installation process it's going to ask you for the password that you set just a while ago when you set up a **Ubuntu**. When they ask you to confirm also go ahead and hit enter.

Now once it's done installing homebrew, there are two steps that you have to complete and they will be shown inside Ubuntu screen. Basically this is going to make homebrew available to use in the path.
It doesn't really matter if you don't understand what that means. It's just a little bit of manual setup that's required.

next steps section:

```
echo 'eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"' >> /home/ <My username> /.profile

eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"
```

Install Homebrew's dependencies if you have sudo access:

sudo apt-get install build-essential

Type each of the above lines separately in the Ubunto prompt and hit Enter.

Remember to enter your password and type 'Y' when they ask you to and let it go through the full installation process.

Once you see that $ sign again, that means it's all done and you can now make sure by typing in **brew --version**.
And if you see homebrew followed by some sort of version number, then that means everything was successful.

7. **Use Homebrew to install Node:**

Now finally, we're going to use Homebrew to install Node version 20. This is the latest stable version and this is the version that will work with the Internet computer.
In the Ubuntu window, type: **brew install node@20**
After installation completed type then: **brew link node@20**
And now you can check the installed & linked node version by typing: **node --version**

When it returns v20.11.1 that means that everything went successfully.

8. **Now finally we're ready to install dfx, so go ahead and copy that line of code and paste it into the Ubuntu command line:**
   For DFX version0.9.3: **DFX_VERSION=0.9.3 sh -ci "$(curl –fsSL https://sdk.dfinity.org/install.sh)"**
   For the Latest DFX Version: **sh -ci "$(curl -fsSL https://smartcontracts.org/install.sh)"**

Now it's going to take a little while to fetch dfx which is the package that's going to allow us to work with the Internet computer locally.

9. **Once it has installed, then it's going to tell you where it was installed and we have to manually set up the path to point to that location.**

So in the installation guide, I've got this stub for you which I want you to paste into notepad and replace the part which says 'replace with your installation path' with the installation path you got from the dfx install.

So copy that, paste that in there, make sure there's no space between the colon and the first forward slash.
export PATH=$PATH: <REPLACE WITH YOUR INSTALLATION PATH>

In my case it was: /home/ <My username> /.local/share/dfx/bin

Copy that entire command, paste it back into **Ubuntu**, hit enter to run that command. And now if you copy that line where it says echo and then a bunch of symbols: **echo "${PATH//:/$'\n'}"** you should be able to see that your dfx location was added to your path like I've got here.
Now, finally run **dfx --version** and you should the DFX installed version showing up.

And if you do, then that means dfx was successfully installed and you're ready to finally get started by creating your first internet computer application.

10. So open up VSCode and select new wsl window.
Now, once you've done that, you can close down the previous window if you want, but make sure that the window that you're working with says WSL: **Ubuntu**. And if you hover over it, it should say **running in Ubuntu**.
Now go ahead and open up **Ubuntu** from the start menu again and create a directory **ic-projects** by typing:
mkdir ic-projects and hit Enter to create this folder called **ic-projects** inside your main user folder, and then change directory to the created folder: **cd ic-projects**. Inside this folder you're going to create all of your Internet computer projects.

And inside that folder, go ahead and type the command **dfx new hello**.
And this is going to build a sample Internet computer application called **Hello**.
It is going to ask you questions awaiting for you to respond by selecting one of the multiple choices provided to you:
? Select a backend language: ›
❯ Motoko
Rust
TypeScript (Azle)
Python (Kybra)

✔ Select a backend language: · **Motoko**
? Select a frontend framework: ›
SvelteKit
React
Vue
❯ Vanilla JS
No JS template
No frontend canister

✔ Select a backend language: · **Motoko**
✔ Select a frontend framework: · **Vanilla JS**
? Add extra features (space to select, enter to confirm) ›

⬚ Internet Identity
⬚ Bitcoin (Regtest)
⬚ **Frontend tests**

Select the proper selection as per your project requirement and hit Enter every time and once it's built, you'll see the Dfinity logo show up and you can see where your folders and files are by using the command **explorer.exe .** Then you can see your hello folder with all the template files that dfx created.

And you can now go back to **VSCode,** and select **open folder**, select the **ic-projects/hello**. And it should now open up your project in VSCode.

Now if you take a look inside the source folder **src**, you'll see two main folders:

1- **hello_backend:** This folder includes **main.mo** which is our **Motoko** file. But at the moment, it doesn't actually have any syntax highlighting, which is what our **Motoko** extension is supposed to do. So if you head over to the extensions tab, you can see this extension **Motoko** is currently disabled. So, click on the button where it says install in WSL: **Ubuntu** so that we can make it actually available in our WSL remote. And if you head back to **main.mo** you should see all the syntax highlighting.
2- **hello_frontend:** This folder includes some files: index.html, package.json, tscinfig.json and vite.config.js. It also includes subfolders: assets and src folder which includes js and scss files.

To deploy this sample project created by dfx, go into the terminal menu and open new terminal. In the new terminal, write to **dfx start** in order to start the local Internet computer and hit Enter

And now once you see the localhost for the Dashboard, it means that dfx start was successful, and then I you need to split out a new terminal window by clicking on the Split Terminal button and then type in **dfx deploy** in order to deploy this hello project onto that **local dfx** that we just started.
It will ask you to enter your passphrase for your identity (your Ubuntu user password, you have defined earlier). Type your password and hit Enter

Once it's done and you see a dollar sign again, you can notice that above the Dollar sign, two links mentioned, one for **hello_frontend:** http://127.0.0.1:4943/?canisterId=d6g4o-amaaa-aaaaa-qaaoq-cai

And the other is **hello_backend** for Backend canister via Candid interface:
http://127.0.0.1:4943/?canisterId=dxfxs-weaaa-aaaaa-qaapa-cai&id=dzh22-nuaaa-aaaaa-qaaoa-cai

So you can start up your server either by following the **hello_frontend** link or by typing **npm start** to start up your server.
And if you scroll up, you actually see it tells you where the project is running.
http://localhost:3000 or whatever mentioned in your terminal.

So go ahead and copy that URL and paste it into your browser. And once it loads up, you'll see your starter project, type in your name in the “Enter your name” field, click on “Click me” button and you'll see the greeting show up.

And that means you have successfully installed and set up everything that's required to start developing on the Internet computer.