How to know if you have learned something -

How do I know if I have learned something?

To check how well you understand something you are learning, test yourself against the 3 steps above, with the following questions.

1. Can you explain it to someone else?
2. Can you describe how the new material is similar or different to something else you already know?
3. Can you apply this new material to a different situation and use it to solve a problem?

Try this out with your pre-course work topics.

Ways to make notes/learn -

Making notes is a great way to embed new knowledge and to have something to refer back to later. However it's also time consuming and, at CodeClan, you won't always have that time. The CodeClan immersive experience means that codealongs go at typing speed. You will not have time to write down everything that is covered. Therefore during your pre-course work, get in to the habit of noting down only brief pointers on new material.

Here are some ideas of ways to take brief notes.

- Note down one summary sentence for the lesson.
- At each break, note down one headline from the previous section.
- Note down how this material is different to the last thing you covered.
- Note down only the shortcuts, or actions you take to put the new material in to practice (like a personal key).
- Note down only your questions as they come up.

When to ask a question -

Finding questions to ask about something you are learning is a good thing. It means you are testing the material you are learning, and checking how it fits with what you already know. It also shows others where you are with your understanding, and enables them to help you better. If you struggle to identify questions when covering new material, ask yourself:

- What do you already know?
- Does the new content make sense?
- Does it contradict something you already know?
- In what situations would this new content be useful/applicable?

Add the questions you come up with to your notes, or make a list. Some of these questions may be answered in upcoming parts of the material. Look out for opportunities to find answers to these questions.

How to problem solve -

We all problem-solve every day. By becoming more aware of the process, and by more quickly recognising times when you need to problem-solve, you can greatly improve this skill. Here's our handy cheat sheet to help you become more aware of this process.

1. Recognise when you need to problem-solve. If you feel stuck, or you're at a decision point in your work, it's time to start problem-solving.
2. Check your understanding. Can you describe the problem or situation to someone else? If you don't have a friend handy, explain it to your rubber duck. If the problem is not clear, now is the time to seek help. When trying solve a problem it's really important to understand what it is you are trying to solve. This might sound really obvious but often people don't get the results they expect because they haven't understood the problem. They've tried to solve what they ***thought*** the problem was, rather than what it actually was.
3. Plan. What is your first approach going to be? Are there any other routes you could explore?
4. Break down approach the first approach you have thought of. What are the sub-problems? Is there something simpler, or similar that you know how to do? How can you apply this knowledge?
5. Start solving - start solving your sub-problems, one by one.
6. Revisit the plan - do you need to explore your other approaches?
7. Don't give up!

FlowChart diagram how to -

blems

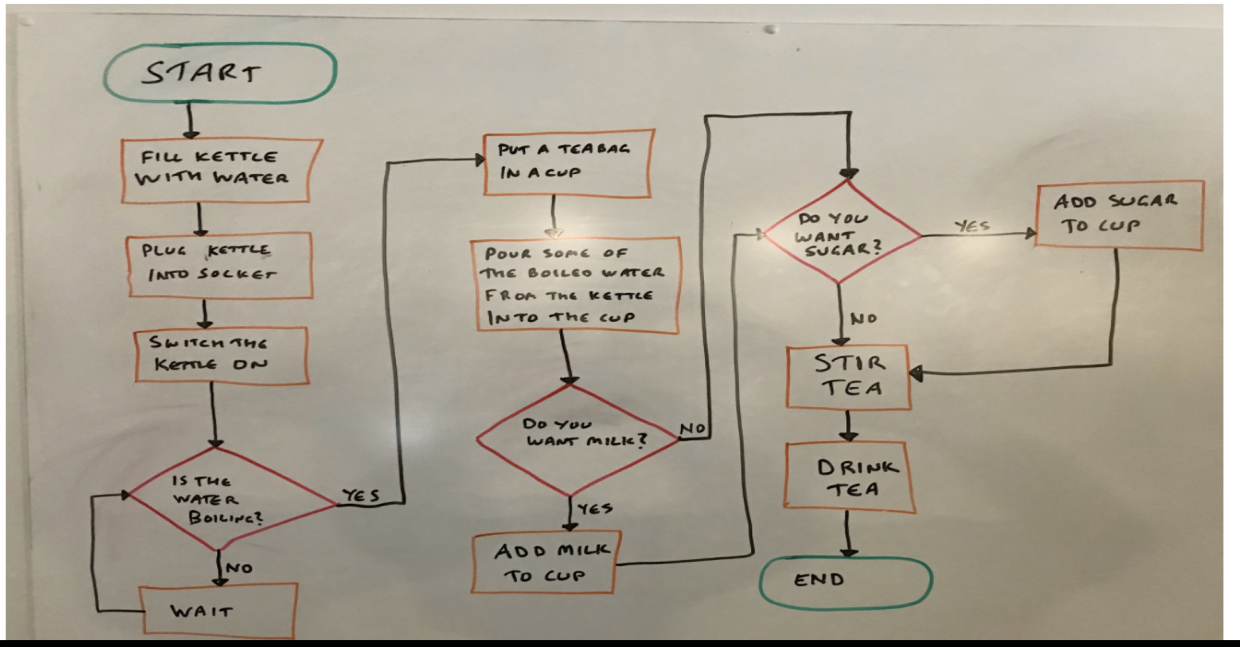
Flowcharts usually use the following set of symbols:

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

FlowChart example -

blems

Below is an example of a possible flowchart representing the process of making a cup of tea:



Basic coding terms/statement -

Word	Meaning
variable	a name for the place where a piece of information is stored
value	a piece of information that can be stored in a variable
assignment	the act of putting a value into a variable
array	a storage container for objects we have in our program
element	an individual piece of data stored in an array
index	the position of a specific element within an array
function	a piece of code which can be run over and over again
call	to run the code in a function
argument	a value that is passed into a function when it is called
parameter	a value that a function takes when it is called
return	the value that results from a completed function call
loop	a piece of code which repeats itself again and again
conditional	a statement which makes a decision depending on whether something is true or false

What is a URL -

What is a URL?

Elements of a URL

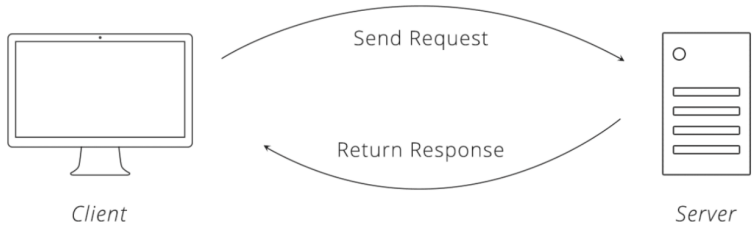
```
http://www.example.org/hello/world/foo.html
\___/ \_____________/\_____________________/
protocol  host/domain name       path
```

Element	About
protocol	the most popular application protocol used on the world wide web is HTTP. Other familiar types of application protocols include FTP, SSH, HTTPS
host/domain name	the host or domain name is looked up in DNS to find the IP address of the host - the server that's providing the resource
path	web servers can organise resources into what is effectively files in directories; the path indicates to the server which file from which directory the client wants

Client - Server relationship -

Web Requests

what is the server doing in the background?



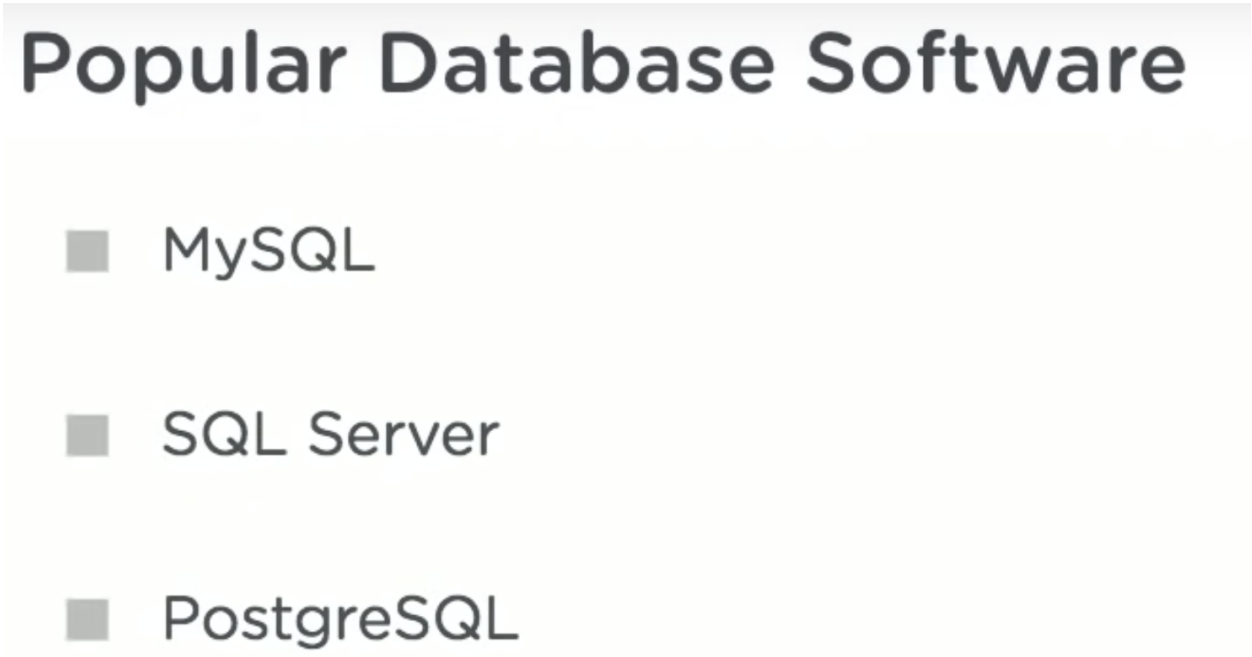
Essentially... the client doesn't care what the server does, as long as it gets the content it asked for.

- Could be serving static files
- Could be pulling something out of a database
- Could be calculating something and making it up as you go along

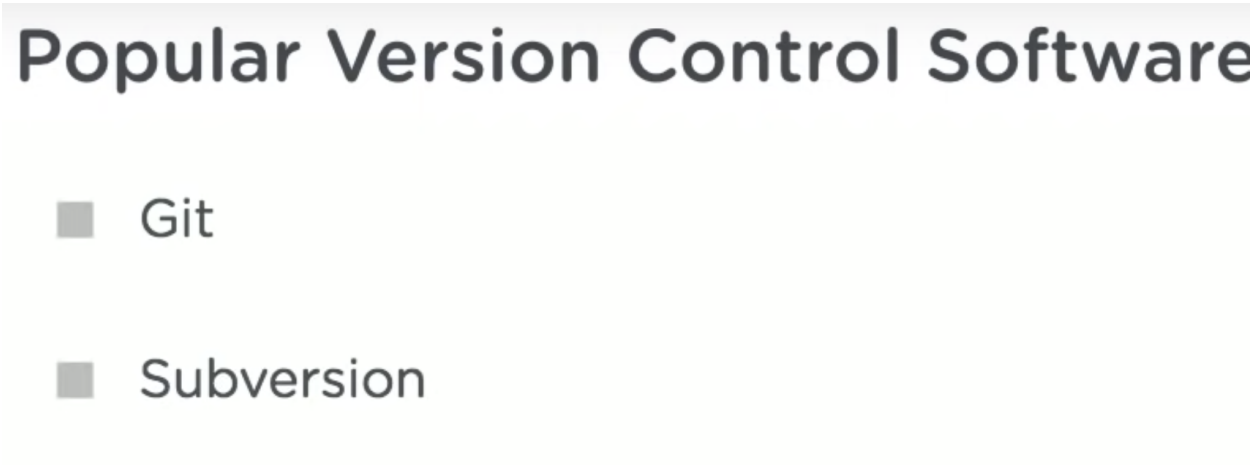
The language of the requests is formalised in the protocols they're made in. Mostly this will be HTTP (and HTTPS).

Programming languages and their web frameworks -

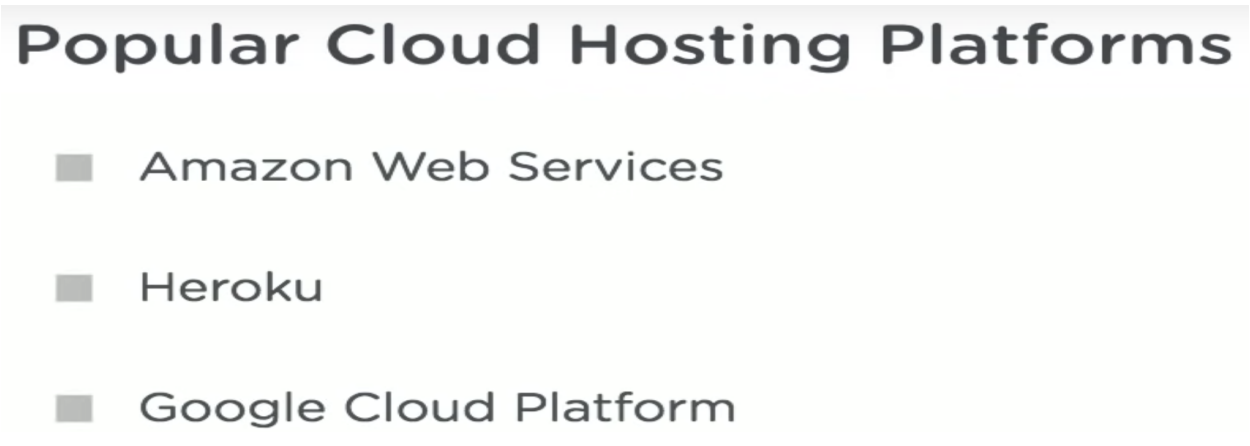
Popular database software -



Popular version control software



Popular cloud hosting platforms



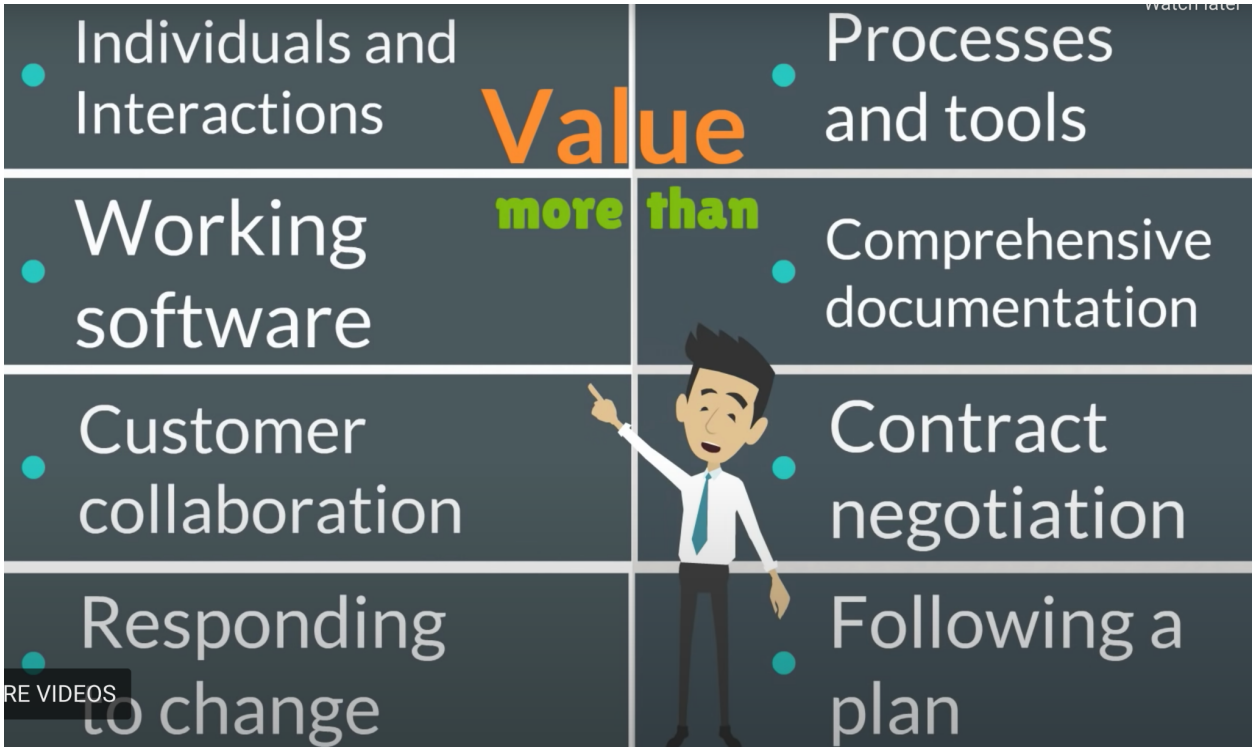
Front end developer - needs to be good with

Agile working -

Agile is a development process in which

- Focus should be more on individuals and interactions instead of processes and tools
- Working software is more important than comprehensive documentation
- Customer collaboration is more vital than contract negotiation
- The process should respond to change rather than follow a plan

What to value most in agile working -



The 12 principles of Agile -

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity - the art of maximizing the amount of work not done - is essential
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly

How to use terminal - mac

Learning Objectives

- be able to access the command line using Terminal
- be able to check the current directory using `pwd`
- know that `~`, `..` and `.` are shortcuts for the home directory, parent directory and current directory
- be able to create and delete directories using `mkdir` and `rm -r`
- be able to create, rename, and delete files using `touch`, `mv`, and `rm`

Basic terminal commands -

The Mac Application used to access the command line is called "Terminal".

To open it up, if it's not in your Dock, you can find it by using Spotlight Search (cmd + space) and typing "terminal"

When the terminal opens, you are at your personal "home" directory. The "tilde" symbol: `~` is a shorthand for your "home" directory.

You can check where you are at any time with the `pwd` command. (Stands for "Print Working Directory")

```
pwd
```

You expect to get back `/Users/user` as this is the full path of your home directory (aka `~`)

To see what you have in a directory, you use the “list” command: `ls`

```
ls
```

This should display a list of the various files and directories within your home directory, Applications, Desktop, Documents, etc.

> You can change how a command works by passing a “flag” or “option” to it.

```
ls -l
```

> The `-l` flag (l for “long” version) gives you extra information like the size of the files, who has access to the files, etc .

```
ls -a
```

> The `-a` flag (a for “all” files) displays hidden files and directories as well as normal ones. The name of hidden files starts with a `.`
>
> You can also combine these flags to get a long version of the list of all files

```
ls -al
```

Now that you have created a directory, you can move into that directory, in other words change your current directory to be the one you just created. To do this use the 'change directory' command (`cd`) followed by the name of the directory you want to move into:

```
cd pre_course_work
```

So if you think of the tree directory structure, you have now moved one level down the structure. To go back up one level you would still use the `cd` command, but rather than giving the directory name, we can use `..` which is a shorthand for the directory one level above where you are right now.

```
cd ..
```

The shortcut for your home directory is `~` so wherever you are in the directory tree you can immediately get back to your home directory with:

```
cd ~
```

You can go back to your home directory by using the `cd` command without giving the name of the place you want to go to:

```
cd
```

To create a new file you can use the `touch` command:

```
touch text_file.txt
```

You can also create multiple files at once by giving the `touch` command multiple filenames.

```
touch a_ruby_file.rb another_file
```

Notice we choose our file extensions explicitly, and can even create files without any file extension.

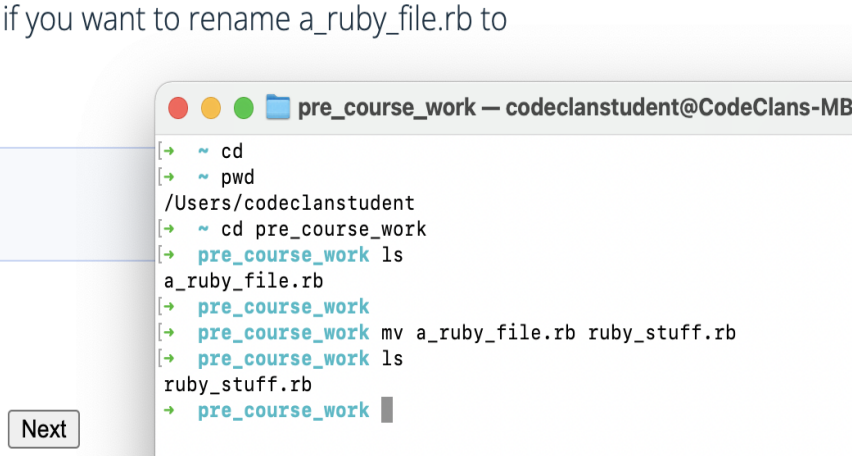
So we've got a ruby file in our home directory. If you want to move that into the pre_course_work directory:

```
mv a_ruby_file.rb pre_course_work
```

The `mv` command moves a file from the first location to the second location.

To rename a file you also use the `mv` command. Instead of giving a location to move a file to, you give a filename. So if you want to rename a_ruby_file.rb to ruby_stuff.rb:

```
mv a_ruby_file.rb ruby_stuff.rb
```



Next

To delete a file we use the 'remove' command: `rm`

```
rm text_file.txt
```

To delete an empty directory use the `remove directory` command: `rmdir`

```
rmdir pre_course_work
```

This command will only work for an ***empty*** directory.

WARNING: deleting things from the command line then ***immediately*** deletes the item ***forever***. There is no 'Trash' or 'Recycle Bin' where the file is put temporarily, so you should really be sure you want to delete something before you do so.

More command line statements and their uses -

Determining File Type

Sometimes the extension will not help you in determining the type of file. In that case the following command can help you pinpoint the exact type.

file <file>Determines file type

Change Directory commands

The following commands will let you change the directory that you are currently working in. This is especially useful when selecting the directory you want to run a command in.

cd	Go to Home Directory
cd [folder name]	Change directory (If the directory you would like to navigate to is not in the current directory, the complete file address is required)
cd ..	Move up to the parent directory
cd ../..	Move up two levels (Could be extended to as many levels as possible by adding ../)
cd ~	Go to Home Directory

File and Directory management

Let's look at how you can create, edit, and delete directories and folders.

mkdir <dir>	Create a new subdirectory in the current directory
mkdir <dir1> <dir2> <dir3>	Create several directories at once.
mkdir "<dir>"	Create a folder with a space in its name
rm -R <dir>	Remove a directory and its contents
cp -R <dir> <"newdir">	Copy a folder to another folder with spaces in its name
touch <file>	Create a new file
nano <file>	Opens a Terminal file editor. You can make changes to your files right from the Terminal.
cp <file> <dir>	Copy a file to a directory
cp <file> <newfile>	Copy a file to the current directory with the name given as <newfile>
rm <file>	Remove a file completely. This will remove it completely from the system so be careful when using this command.
rm -i <file>	Deleting a file after providing confirmation.
mv <file> <newfile>	Move a file to another file/ Rename a file
mv <file> <dir>	Move a file to a folder and will overwrite existing files
mv *.txt <dir>	Move all text files of the current folder to a different folder