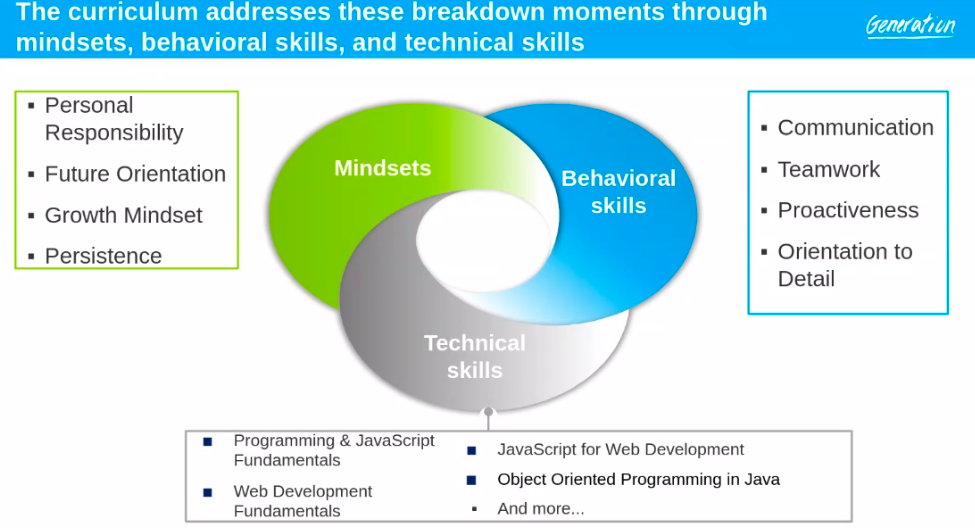
Quiz #1 18 questions

Quiz #2 21 questions

Both multiple choice

* Breakdown moments (Canvas Day 1 and etc.)

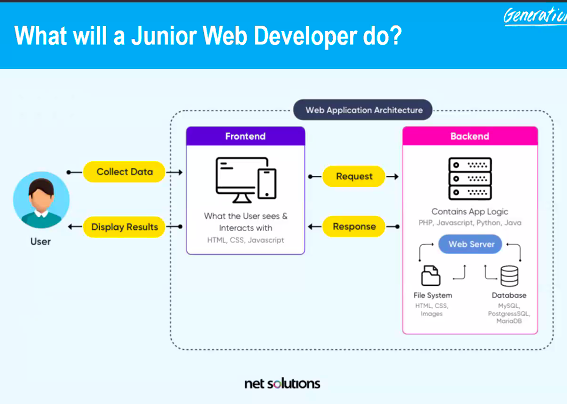


* Responsibilities of a Junior Web Developer / Full Stack Developer (Canvas Day 2)

The word “responsibility” can mean different things to different people. At Generation, we define personal responsibility as (1) taking ownership, (2) working hard, (3) keeping commitments, and (4) not making excuses. Read the characteristics of personal responsibility and their definition.

**Read each of the characteristics and definitions below.**

| **Characteristic** | **Definition** |
| --- | --- |
| Taking ownership | * Completing the work or task you start * Taking control of what you do |
| Working hard | * Putting forth your best effort * Following instructions and directions carefully |
| Keeping commitments | * Doing what you say you will do * Showing up for work on time (or early) |
| No excuses | * Not making any excuses for poor work, being late, or if you're unable to do something * Not blaming someone else when problems arise, especially if you had something to do with it |



<https://www.youtube.com/watch?v=GEfuOMzRgXo>

| **A Day in the Life of a Junior Web Developer** | |
| --- | --- |
| Beginning of the day | * Arrive early for any meetings with the team and/or with the supervisor. * Prepare for the day by turning on the computer, logging in to system, email, Slack, calling system, company databases, etc. * Ensure that accessories are present (keyboard, monitors, headset, etc.). * Ensure other materials are there to prevent interruptions to work (e.g. water bottle, notepad, etc.). * Mentally engage for the day by focusing on tasks at hand and goals for the day. * Ask clarifying questions as needed. |
| Throughout the day | * Participate in daily Scrum meetings or Sprint Planning/Retrospective. * Review tickets in the backlog and check email and collaboration channels to align on shifting priorities. * Carefully read and resolve any tickets in your assigned queue. * Scan issues for urgency and your ability to solve them. Resolve as quickly as possible. * As you encounter issues that you cannot resolve, research Google and other resources available to you to help you drive the resolution as far as possible. * If issues are beyond your skills, escalate or involve teammates as appropriate. * Follow company policy on documentation, ensuring that all of your actions are carefully captured in the internal system for tickets or within notes in code. |
| End of the day | * Complete shift by logging out of all systems, cleaning work area, and sending any necessary emails. * Perform any shift change responsibilities to ensure the next employee is adequately prepared to begin his/her shift immediately. |

* Qualities of a good a Junior Web Developer / Full Stack Developer (Canvas - ..)

The following are the qualities of an exceptional Junior Web Developer:

* Efficiently solves complex or detailed problems
* Creates high quality code that is easily readable and reusable by other developers
* Actively problem solves issues as they arise
* Is able to quickly articulate their own knowledge gaps and find the appropriate resource for support
* Enjoys working with a team to overcome challenges
* Is motivated to learn new information quickly
* Receiving & giving feedback (Canvas - Day 11)

Successful junior developers use the following tips for effective communication when giving and receiving feedback. **When receiving feedback** from your supervisor you should always:

* Show appreciation
* Ask how you can improve
* Verify you understand the specific steps to improve
* Demonstrate through your work that you heard the feedback and are trying their suggestion

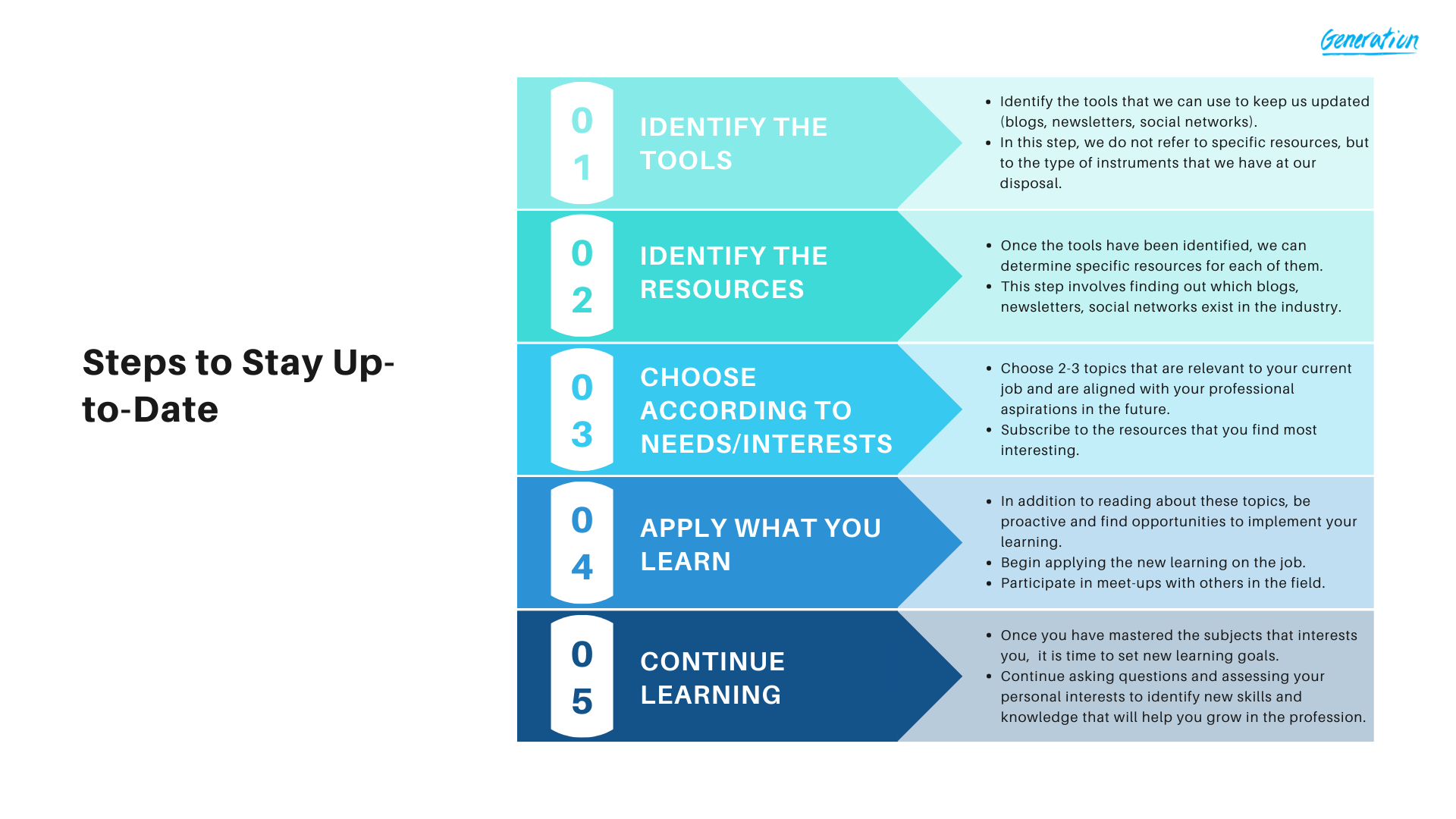
A developer may need to talk to a supervisor if:

* a developer is asked to complete a task for which s/he does not know the required programming language
* a colleague asks for help with a task that puts an assigned deliverable at risk
* a developer needs increased permissions within a project in order to complete the task
* a junior developer doesn’t know how to respond to a complicated issue

While there is no correct answer about whether or not to escalate an issue to your supervisor, the best way to go about dealing with these situations is by thinking about them and considering what you would do if they do arise.

**Junior developers** should always approach their supervisor with a question or situation if it has the potential to cause a bigger problem if it goes unaddressed.

* + They should use discretion when interrupting their supervisor’s work for a question that could be answered by a coworker or by reading an employee handbook.
  + They shouldn’t interrupt their supervisor’s work for small talk but wait until a more casual opportunity presents itself (lunch break, etc.).
* Staying up to date in the industry (Canvas - Day 15)



Each of the following is a tool that we can use to keep us updated as junior developers:

1. Blogs & Newsletters
2. Social Networks
3. Webinars, Courses and Tutorials
4. Meet-ups

If you are not familiar with any of the above tools, read below:

* **Blogs:** Discussion forums in which different users ask and answer specific questions.
* **Newsletters:** Publication that companies send to affiliate members to communicate interesting information about advances or new developments.
* **Social networks:**. As a junior developer, you can follow experts in areas of your interest, take ideas from what others publish and establish contact with people who can teach you something you want to learn.
* **Webinars, courses, and tutorials:** Online classes or videos specialized in certain topics of interest.
* **Meet-ups:** Conferences in which the members of the industry participate and where they can learn about new trends and advances. These can be face-to-face or virtual.

Some resources include:

* Blogs & Newsletters:
  + <https://stackoverflow.com/>
  + <https://www.geeksforgeeks.org/>
  + <https://www.w3schools.com/>
  + <https://martinfowler.com/>
  + <https://www.infoq.com/>
  + <https://medium.com/>
* Social Networks
  + Reddi
  + Twitter
  + LinkedIn
  + Facebook
* Webinars, Courses, and Tutorials
  + Podcasts
  + YouTube
  + Online learning platforms (such as Udacity or EdX)
* Meetups/Conferences
  + <https://www.meetup.com/topics/software-engineering/>
  + <https://techmeetups.com/>
* Problem solving steps (Canvas - Day 18)

7 key steps to solving a problem, which include:

1. Define the Problem: Step 1 is to **define the problem**, which means **to make sure you are handling the right problem.** Sometimes problems can be complicated and confusing, and it is important to break them down to uncover the real problem.
2. Come up with Solutions: Step 2 is to **come up with as many solutions as possible (brainstorm) using your findings from the previous step, your knowledge and your creativity.**
3. Do Your Research but Don’t Reinvent the Wheel: Step 3 is to do research but don’t reinvent the wheel. This **means figuring out which questions you have to ask and which existing resources you can pull from to help you implement your solutions.** With a plan of action for what to research, you make sure you don’t reinvent the wheel as you start gathering data. Whatever problem you are facing, chances are that someone somewhere has worked on something similar. So your next step here is to look through all possible internal documents and then look externally.
4. Think About and Choose a Solution: Step 4 is to **choose the best options to provide a solution to your problem.** This involves narrowing down the list of ideas by crossing out any you do not want to pursue, then ranking the remaining ideas. Choosing the best option can often be a challenge. Professionals have developed some tools to help quantify your options and see how they compare against each other! One of these tools is the decision matrix.
5. Take Action: Step 5 is to **take action, to commit to the solutions you selected.** Remember, some problems may even require multiple solutions. Let’s brainstorm actions we could have taken for the problem addressed in Step 1 (being late for the morning huddle)!
6. Hit Singles: Step 6 is to **hit singles, which means to do few things well rather than a ton with mediocre execution or results. Stick to targeted focus rather than perfection and drilling into every little piece. Quality over quantity.**
7. Respect Your Time: Step 7 is to **respect your time, which means to know when there’s a lot of work to be done, and delegate around your limitations.**

* Terms:
  + URL
  + API
  + CLI
  + DevTools
  + Environmental variables
* Agile and SCRUM review sheet

**Read** this [Agile Essentials](https://www.agilealliance.org/agile-essentials/)

guide, sections:

* [What is Agile Software Development?](https://www.agilealliance.org/agile101/)
* [Agile Manifesto for Software Development](https://www.agilealliance.org/agile101/the-agile-manifesto/)
* [12 Principles Behind the Agile Manifesto](https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/)
* [Agile Glossary and Terminology](https://www.agilealliance.org/agile101/agile-glossary/)

**Take the [Scrum Training Series](http://scrumtrainingseries.com/)**

**sections:**

* [Introduction to Scrum](http://scrumtrainingseries.com/Intro_to_Scrum/)
* [Backlog Refinement Meeting](http://scrumtrainingseries.com/BacklogRefinementMeeting/)
* [Daily Scrum Meeting](http://scrumtrainingseries.com/DailyScrumMeeting/)
* [Sprint Review Meeting](http://scrumtrainingseries.com/SprintReviewMeeting/)
* [Sprint Retrospective Meeting](http://scrumtrainingseries.com/SprintRetrospectiveMeeting/)
* <http://www.scrumreferencecard.com/>

Rhea’s notes: <https://docs.google.com/document/d/13w78lBOuXjZzlfEALQcNjq3Z3z0g1ofpdIzSxnq66gQ/edit>

* Command Line & Git review sheet