**Video Script: Section 7 Video 2 – understanding scoping**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Description | Action on screen | Narration |
| 1 | Introduction  (Outcome and why it is desirable)  This should give the viewer an idea of the outcome of the task at the beginning of the videos and set the stage and expectations of the viewer. | Refer to PPT | In this video, we are going to have a look at scoping, i.e. knowing where variables can be accessed from. |
| 2 | Context(Problem/Solution)  Present the viewer with a real-world solution and how the situation would pose as a challenge. It always helps to draw the viewer's attention using a use-case. Metadata template can be used here. |  | If we want our webpage to be rendered efficiently, it’s important to know when the code is evaluated, for example to avoid loading a large file multiple times unnecessarily. |
| 3 | Guidance (How to do it and how it works): |  | The scope of a variable in Shiny is simply defined by where it was declared.  Let’s have a look at an example. |
| 4 |  | Open R, run  library(“shiny”)  runApp(“activity\_07\_02”) | Open R and run:  shiny::runApp(“activity\_07\_02”) |
| 5 |  | A description... | Open the webpage in a different browser or another tab. |
| 6 |  |  | Three things to notice:   * Both pages share the same random value 1. * Each user has its own random value 2. * A global variable, with the value 42 appears both in the side bar and the main panel. |
| 7 |  | Open RStudio and edit UI.R, server.R and global.R. | Let’s go through the code to see how this works. |
| 8 |  | Go to server.R  Highlight randomValue1 | In server.R, we see that a random randomValue1 is defined before the call to shinyServer(). This will make it available for all users. This is because server.R is run once, when you launch the server with runApp. |
| 9 |  | Highlight randomValue2 | randomValue2 is defined inside shinyServer(), which makes it only available for each user. This is why the two browsers show a different value for randomValue2.  This is because the function shinyServer() is called each time the server serves a new page. |
| 10 |  |  | Because the code sitting outside shinyServer() like randomValue1 is evaluated only once, when the server starts up, it’s a good idea to load large files there, for the sake of efficiency . |
| 11 |  | Overlay  randomValue1 <<- “new Session ID” | ~~As any R global variables, if you wanted to change their value globally, you’d need to use the double arrow for assignment:~~  ~~randomValue1 <<- “new Session ID”~~ |
| 12 |  | Open global.R in the editor. | Another way to use global variables is to define them in the file global.R.  Like ui.R and server.R, it is a reserved filename used by shiny.  Variables declared there have global scope but can also be accessed by UI.R, which is not the case for global variables declared in server.R.  AglobalVaribale is used here in server.R, so it appears in the main panel.  And it is used here in ui.R, so it appears in the side bar as well. |
| 13 |  |  |  |
| 14 |  |  |  |
| 15 |  |  |  |
| 16 | Conclusion:The video concludes by showing the viewer that the goal has been achieved, and reminding them why they should be happy about that. A PowerPoint summary slide with the key points emphasized would make it easier for the viewer to remember what was covered in the video | Back to PPT | In this video, we saw the different levels of scope for variables in shiny. Slow processes that only need to be run once like loading a large file should be put outside the call to shinyServer in server.R, as well as functions that are not reactive, e.g. helpers and libraries.  In the next video, we will see how a user can upload a file to the webpage. |