How do we answer these questions?

How important is it for DX with:

1. *The API has code examples* **Why does the API need to have code examples?  
   Who cares about this?**
   1. Let people implement something with/without code examples
   2. Interview questions:
      1. How common would you say it is that software platforms you like have code examples?
      2. In the software platforms you like, are the code examples always good?
      3. Are there software platforms you like that have poor code examples?
      4. Are there software platforms you like that have no code examples?
      5. What issues do you have when there are no code examples, or poor code examples?
      6. How does it make you feel when there are no or poor code examples?
      7. How important would you say it is that a software platform has good code examples in order for you to like using the platform?
         1. Why is it (not) important for you that there are code examples?
2. *The API gives thorough explanations* **What level of detail should the API have in order to give good DX? Does it matter?  
   Who cares about this?**
   1. Let people see two examples of code examples and let them choose which they like
   2. Interview questions:
      1. How common would you say it is that software platforms that you like thoroughly explain their API?
      2. How does it make you feel when the API is not thoroughly explained?
3. *There exists an active online community around the software***What is required of the community around a software in order to give good DX?  
   Who cares about this?**
   1. Interview questions:
      1. How common would you say it is that software platforms you like have an active online community around it?
      2. Why is it important that there is an active community?
      3. How does it make you feel if the software you’re using has an active community?
      4. How does it feel when the community is not active?
4. *I can have working code quickly* **Is it important that you can have working code quickly in order for it to be good DX?  
   Who cares about this?**
   1. Interview questions:
      1. How does it make you feel if it takes a long time to have working code when starting to work on a new software platform?
5. *The pricing of the software*  
   **Does the pricing have to be clearly stated and easy to find in order for it to be good DX?**
   1. Interview questions:
6. *The software is compatible with different platforms* **Is software compatibility a DX-concern?  
   Who cares about this?**
7. *The software uses the programming language I am most comfortable with* **Is providing different language compatibility a DX-concern?  
   Who cares about this?**
8. *How often the software is updated* **Is the frequency of software updates a DX-concern?  
   Who cares about this?**
9. *The software has the same features on all different platforms* **Is it a DX-concern that all features are available on all different platforms?**
10. *The documentation is easy to navigate* **How does it affect someone that the documentation is easy/hard to navigate?**
11. *The official website looks professional* **What are the consequences of the official website looking professional?  
    When / for what types of software does it matter?**
12. *The documentation doesn’t assume any prior expertise* **How does it make a developer feel if the documentation assumes prior expertise?  
    What are the consequences? (Do people give up, feel useless, etc)**
13. *The documentation has consistent language* **What consequences do inconsistency in language have?  
    What feeling is the user left with after reading inconsistent documentation?**
14. *The release- and change notes are thorough***When does it matter that the change/release notes are thorough?  
    What consequences**