**Component Splitting**

* Break UI into components.
* Decide when to create new components.

**Component Size**

* Classify components based on size: small to large.
* Avoid extremes: both very small and very large components are problematic.

**Problems with Large Components**

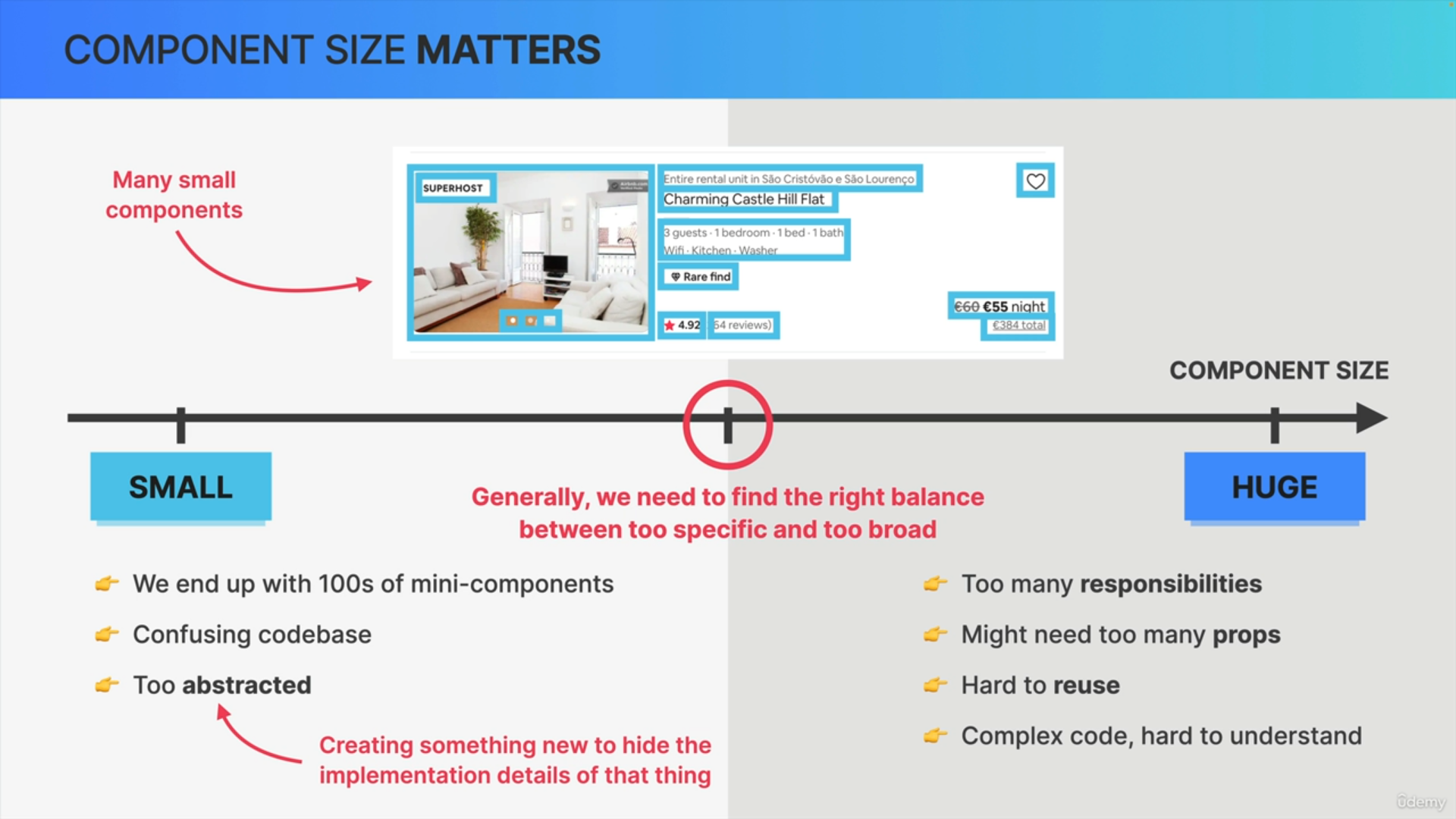
* Too many responsibilities.
* Require many props, making them hard to reuse.
* Complex and intertwined code, hard to understand and use.

**Problems with Small Components**

* Too many small components create a confusing and overly abstracted codebase.

**Optimal Component Size**

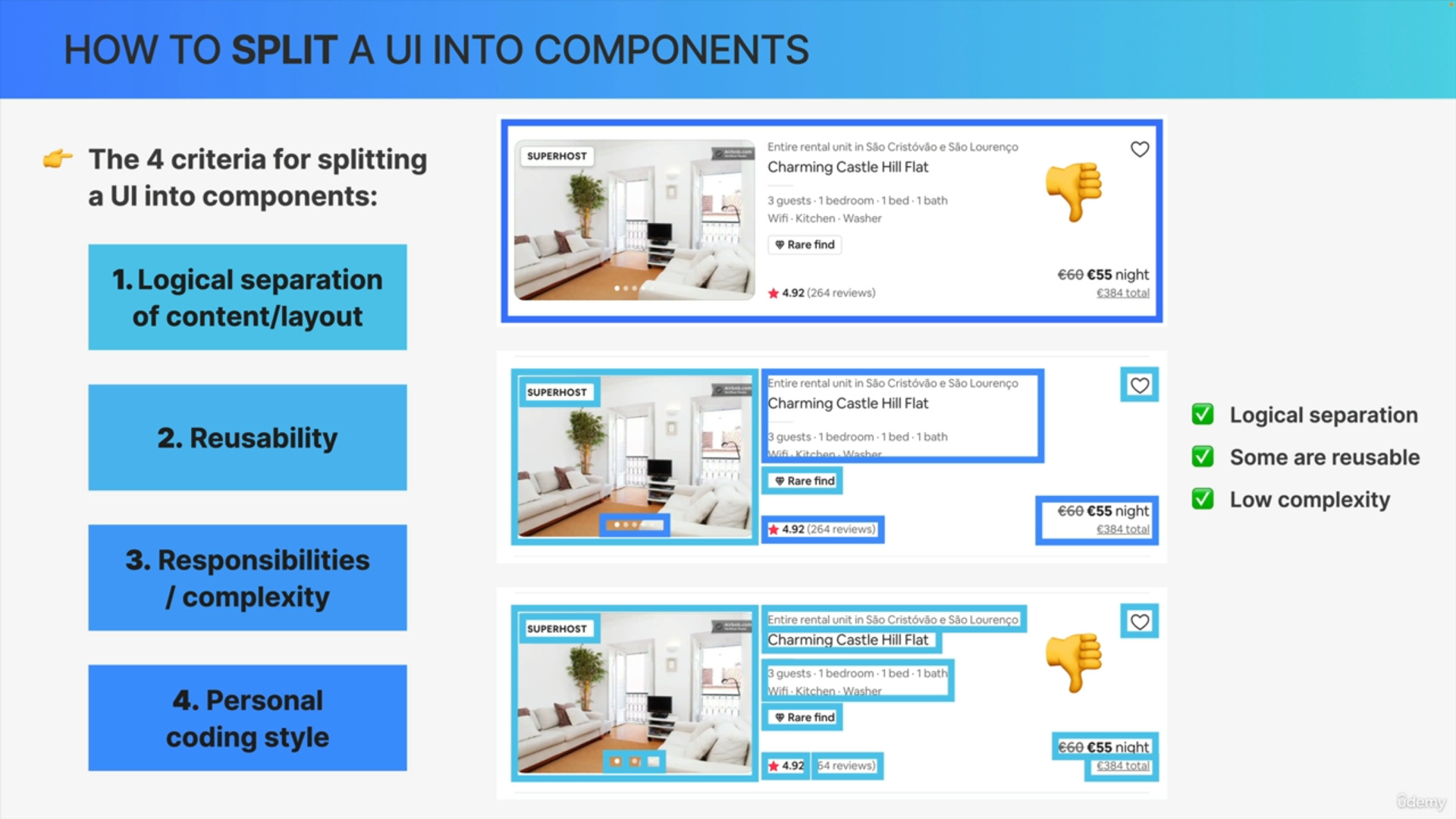
* Balance between too specific and too broad.
* Ensure logical separation of content and layout.
* Aim for reusability and single responsibility.

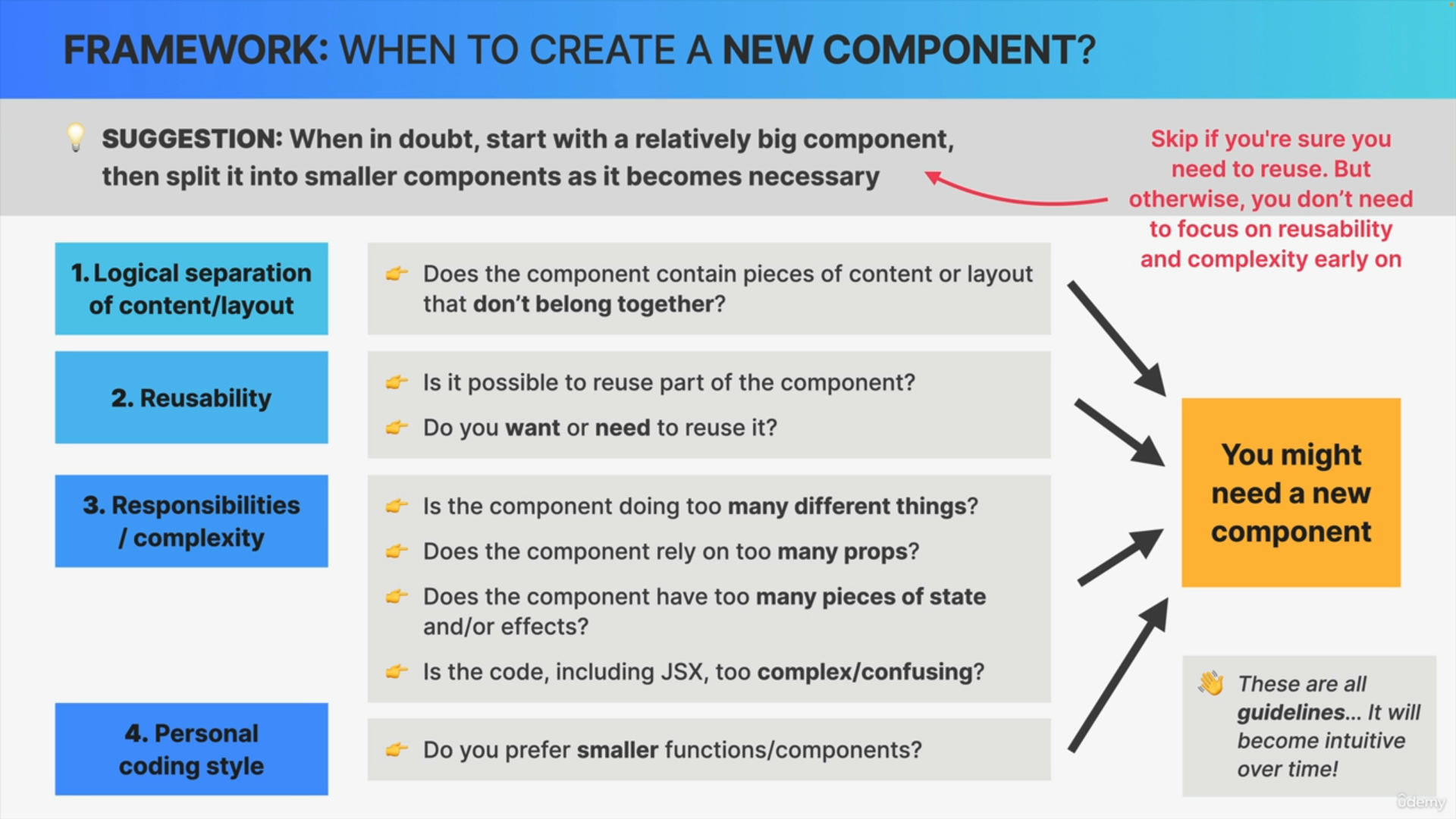


**Component Creation Framework**

* Start with a relatively big component.
* Split into smaller components as needed based on:

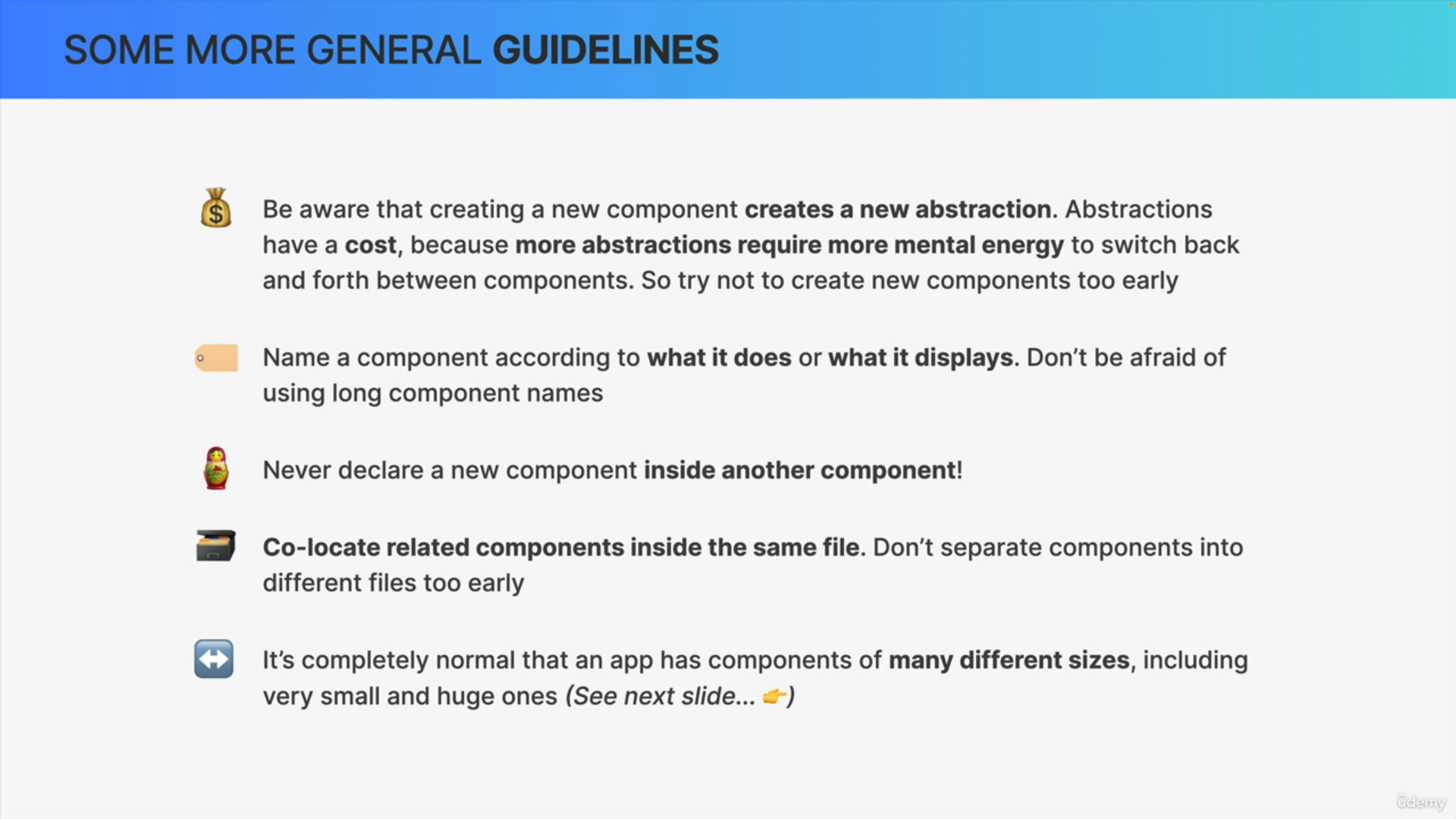
1. Logical separation.
2. Reusability.
3. Responsibility and complexity.
4. Personal coding style.





**General Guidelines**

* Creating new components = creating new abstractions.
* Name components according to their purpose.
* Do not declare new components inside other components.
* Co-locate related components in the same file.



**Component Size and Reusability**

* Applications will naturally have components of various sizes.
* Small components are more reusable.
* Large components may be less reusable but necessary for complex tasks.

**Conclusion**

* Building components will become intuitive over time.
* Use guidelines and best practices to help in the beginning.
* Balance component size, responsibility, and reusability for optimal React development.

**Action**

Apply these concepts to break up a large component into smaller ones in your code.

