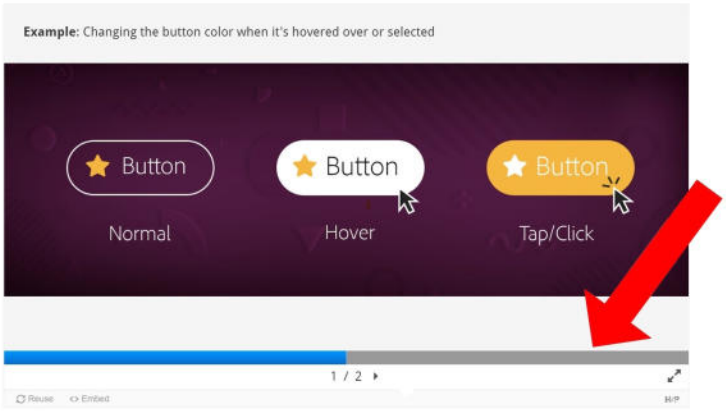
# Exploration: Cognitive Style Heuristics

## How to Complete This Module

This module will teach you about the **9 Cognitive Style Heuristics**. For each heuristic, there's a text description and example images. **Note**: The images are presented as multiple slides. You can **use the arrows or toolbar at the bottom of the image widget to navigate** between images.




Additionally, there's an end quiz with multiple-choice questions for you to complete at the end of the module.

Continue reading below to start learning!

## Introduction: What are Software Usability Heuristics?

Software usability heuristics are guidelines for making software user interface (UI) designs more usable. Software designers go through heuristics one-by-one when evaluating the usability of a UI. Software usability heuristics can also guide the creation of a new UI.

**An advantage** of software usability heuristics is they are an inexpensive way to make a UI more usable: they **do not require testing with actual users**.

## What are the Cognitive Style Heuristics?

The Cognitive Style Heuristics are one set of software usability heuristics. They were **created by a research team** headed by Margaret Burnett, a leader in usability research.

The Cognitive Style Heuristics are **used to find and fix usability bugs in software**. They are **based on research about different ways people problem-solve in software** they're using for

the first time.

At the core of the Cognitive Style Heuristics are five **cognitive problem-solving facets**:

- a user's **motivations** for using software (possible **facet values**: task completion vs. interest)
- their **information processing style** (comprehensive vs. selective)
- their **computer self-efficacy** (low vs. high)
- their **attitude toward risk** (risk-averse vs. risk-tolerant)
- their **style of learning** new software (by process vs. by tinkering)

The Cognitive Style Heuristics are guided by **cognitive style personas**: Abi, Pat, and Tim. Abi's cognitive style consists of a set of facet values at one end of the cognitive style spectrum. Tim is at the other end. Pat is somewhere in between. **Most people are like Pat**---a mixture of facet values. In addition, **an individual's facet values can change with time and circumstances**.

*Note: The blue text is customizable; the blue text shown is only an example. Abi, Pat, and Tim may be any gender.*

Researchers have shown that **designing with cognitive styles in mind can make software less gender-biased** [Vorvoreanu 2019].

To summarize, the Cognitive Style Heuristics are a set of principles based on the cognitive style personas and cognitive problem-solving facets. They allow developers to quickly identify parts of a UI that potentially have usability bugs. Designing for cognitive styles can make software less gender-biased. Keep reading below to learn about each of the heuristics.

## Heuristic #1 (of 9): Explain what ***new*** features do, and why they are useful

Allow Abi to quickly assess new features so they can choose whether to start using them. Allow Tim to quickly assess what a feature is for so they can move on to finding out what the rest of the software's features do.

Abi uses software only as needed for their task. They prefer familiar features to keep focused on the task and may be wary of new features.

Tim likes using software to learn what new features can help them accomplish.

## ▸ euristic #2 (of 9): Explain what ***existing*** features do, and why they are useful

Allow Abi to determine whether existing features are familiar to them. Allow Tim to more efficiently determine which features are known and which are new and unique.

Abi is risk-averse and so may avoid using features that have an unknown time cost and an unknown benefit.

Tim is risk-tolerant so may use features without knowing their cost or even what they do. Tim is also motivated to investigate new, cutting-edge features.

## Heuristic #3 (of 9): Let people gather as much information as they want, and no more than they want

Allow Abi to find the information they want, but don't force them to spend excessive time or effort gathering that information. Allow Tim to find correct information immediately, so that they don't go down the wrong path or get distracted from their task.

Abi gathers and reads relevant information comprehensively before acting.

Tim likes to delve into the first option and pursue it, backtracking if need be.

## ▶ euristic #4 (of 9): Keep familiar features available

To encourage Abi and Tim to continue using the software, allow them to interact with it in ways they expect.

Abi has low computer self-efficacy, so often takes the blame if a problem arises in the software, such as features having changed.

Tim has high computer self-efficacy, so often blames the software if a problem arises, such as features having changed.

## Heuristic #5 (of 9): Make undo/redo and backtracking available

Provide undo/redo and backtracking to allow Abi to feel comfortable proceeding, and to allow Tim to recover from mistakes.

Abi is risk-averse, so prefers not to take actions in software that might not be easy to reverse.

Tim is risk-tolerant, so is willing to take actions in software that might be incorrect and need to be reversed.

## Heuristic #6 (of 9): Provide ways to try out different approaches

Allow Abi to try a different approach when they feel unable to proceed with the current one. Allow Tim to try multiple ways of solving a problem.

Abi has low computer self-efficacy, so often takes the blame if a problem arises in the software. This can discourage Abi from persevering in the software.

Tim has high computer self-efficacy, so often blames the software if a problem arises but is willing to try numerous ways of addressing the problem.

## Heuristic #7 (of 9): Communicate the amount of effort that will be required to use a feature

Allow Abi to decide whether or not a feature will require too much effort to use. Allow Tim to understand that a feature may take extra effort, and thus more time, to help them stay on track with their task.

Abi is risk-averse, so may want to avoid features with high effort costs.

Tim is risk-tolerant, so may begin using features that require extra effort and time, and that are unrelated to the task at hand.

## Heuristic #8 (of 9): Provide a path through the task

Provide Abi a way to go through tasks using a clear process. Provide Tim a way to bypass step-by-step processes and tutorials if those are not required for learning the software.

Abi is a process-oriented learner, so prefers to proceed through tasks step-by-step.

Tim learns by tinkering, so prefers not to be constrained by rigid, pre-determined processes.

## Heuristic #9 (of 9): Encourage mindful tinkering

Encourage Abi to tinker in ways that lead to them discovering task-relevant functionality. Encourage Tim not to over-tinker (e.g., by adding an extra click), so that they make fewer mistakes, have time to absorb important information, and stay on-task.

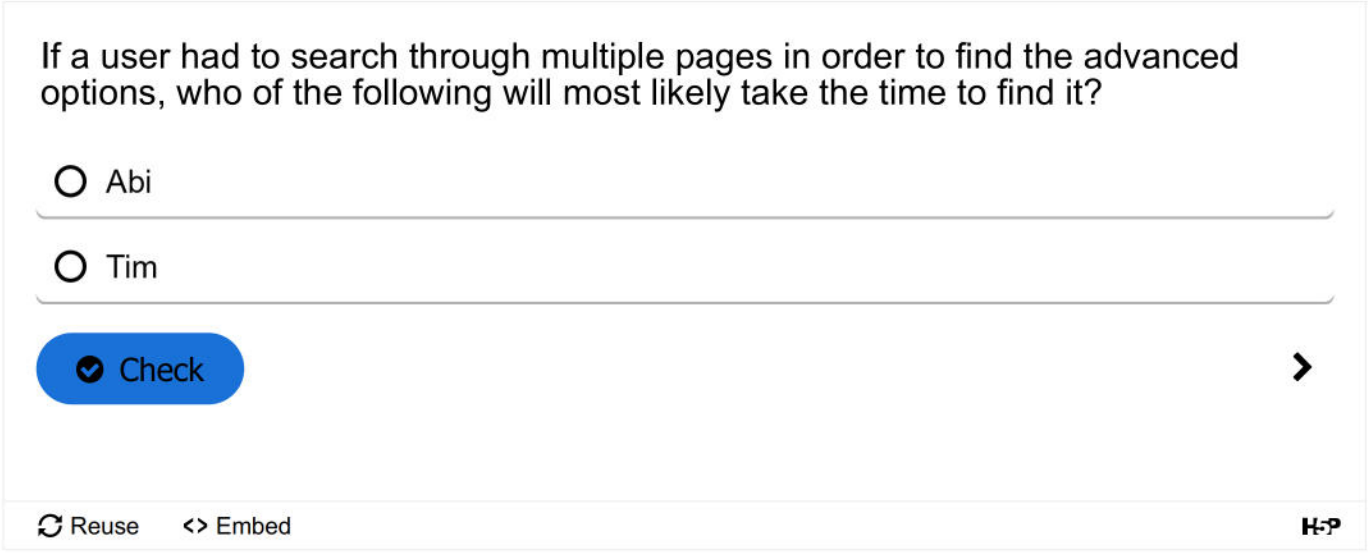
Abi is a process-oriented learner so usually prefers not to tinker. Because of this, they might miss useful or important parts of the software.

Tim learns by tinkering, but sometimes tinkers addictively and gets distracted from their task.

## Wrap Up

The Cognitive Style Heuristics are a set a 9 software usability heuristics for evaluating and improving the usability of UIs for people with different cognitive styles.

## Quiz




## References

Margaret Burnett, Anita Sarma, Claudia Hilderbrand, Zoe Steine-Hanson, Christopher Mendez, Christopher Perdriau. July 2018. *The GenderMag Heuristics (Beta Version)*. https://gendermag.org/flyers_handouts.php.

Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephann Makri, Laura Beckwith, Irwin Kwan, Anicia Peters, William Jernigan. 2016. *GenderMag: A Method for Evaluating Software's Gender Inclusiveness*. Interacting with Computers, Volume 28, Number 6, October 2016, pp. 760-787

Mihaela Vorvoreanu, Lingyi Zhang, Yun-Han Huang, Claudia Hilderbrand, Zoe Steine-Hanson, Margaret Burnett. 2019. *From Gender Biases to Gender-Inclusive Design: An Empirical Investigation*. CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4-9, 2019, Glasgow, Scotland, UK. ACM, New York, NY, USA.

Return to Extra Credit