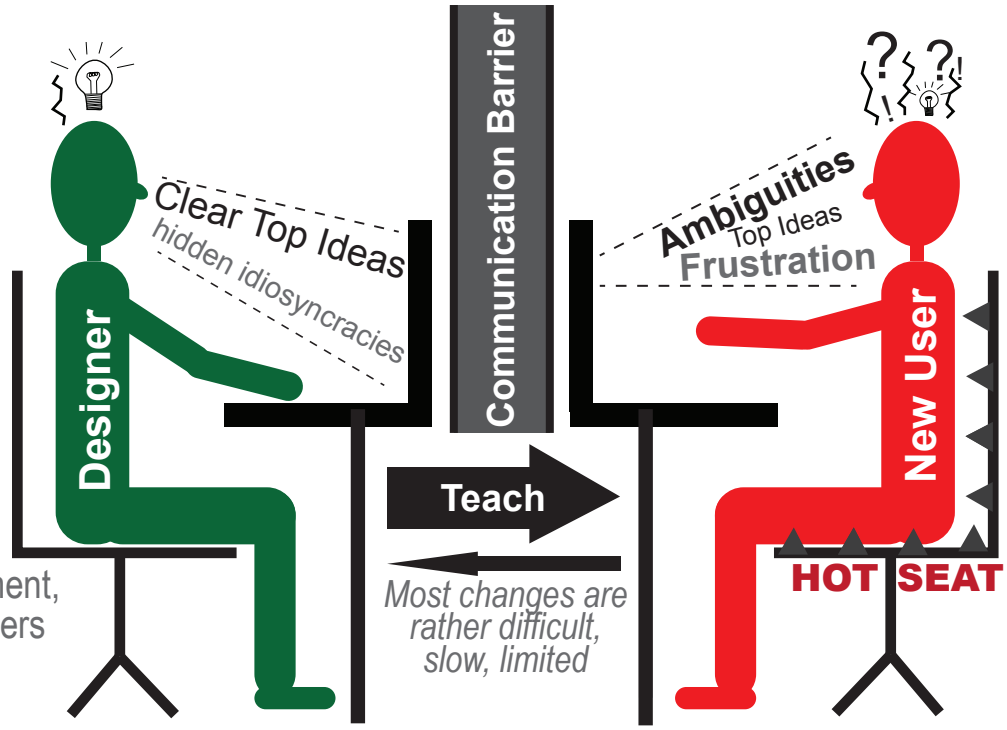
A. Usual Programming Language Design

Life as Designer …

Easier to Design
Harder to Evolve

The designer accepts some avoidable complexity, does not see it as a core problem, or is unaware of how it can slowly poison a language.

Avoidable complexity creeps in with each ambiguity, idiosyncracy, oversimplification, missing key feature, confusing name, extra rule, legacy requirement, or other nuisance that expects users to remember or do things they do not need for their research.



Life as User …

Harder to Learn

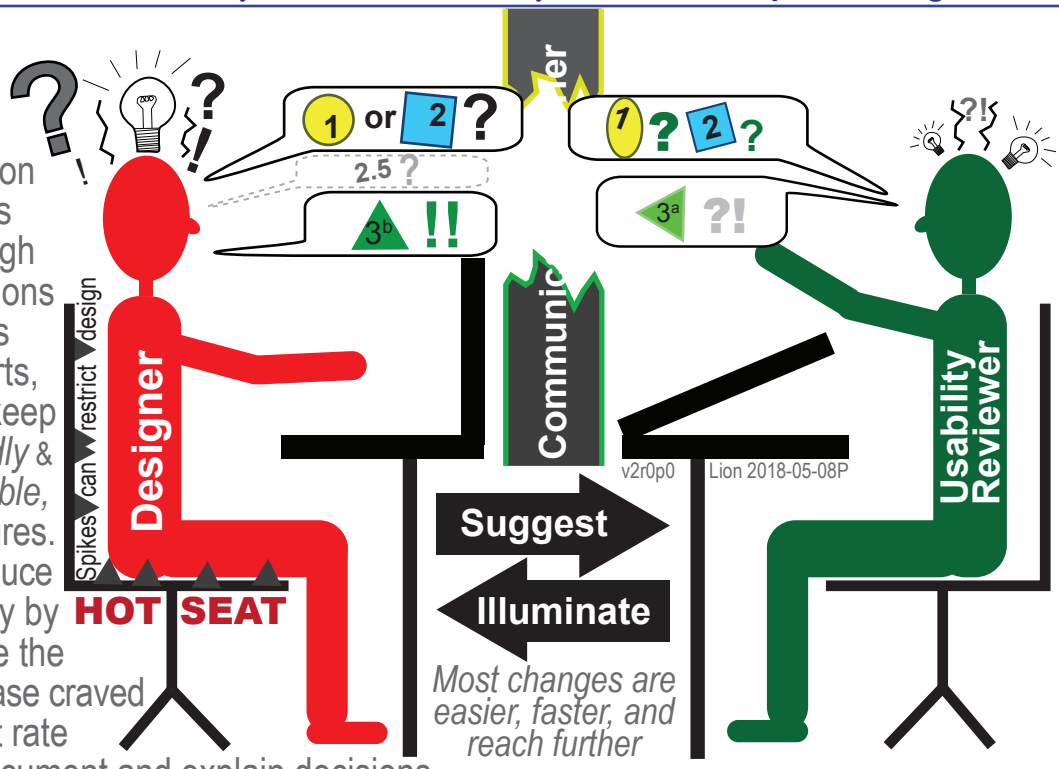
New users are hurled into the Hot Seat when they have to struggle with complexity that designers could have avoided by asking users. Facing avoidable complexity is usually frustrating and maybe even overwhelming; some users never recover and withdraw. Spikes symbolize rules that unexpectedly hurt users or delete data when ignored.

A design flip aims to avoid that learners have to bear the brunt of avoidable complexity in a language.

Flipped language design **asks users** how they would use a new syntax **before implementing** it.

Harder to Design
Easier to Evolve

The designer decides to take on the Hot Seat: find and address all avoidable complexity through sufficiently many design revisions based on in-depth discussions with usability reviewers, experts, and new users. The aim is to keep the language most *user-friendly* & *long-term backwards compatible,* while responsibly adding features. A designer's main job is to reduce long-term language complexity by strategic choices that combine the features for experts with the ease craved by beginners. Designers must rate feature stability accurately, document and explain decisions, organize review discussions, recruit reviewers, prioritize, etc.



Discuss to Design
Easier to Learn

Some potential users enjoy discussing feature and syntax design in advance. These users can excel as usability reviewers, who catch avoidable complexity prior to implementation. Their work is pivotal for reducing the long-term poison of avoidable complexity, which can quickly become invisible to a designer who often fails to predict where new users struggle. To reduce this blindspot, users share how they may interpret a given syntax. The designer takes it from there.

B. Flipped Programming Language Design

Takes a bit **longer.**
Lasts a lot **longer.**