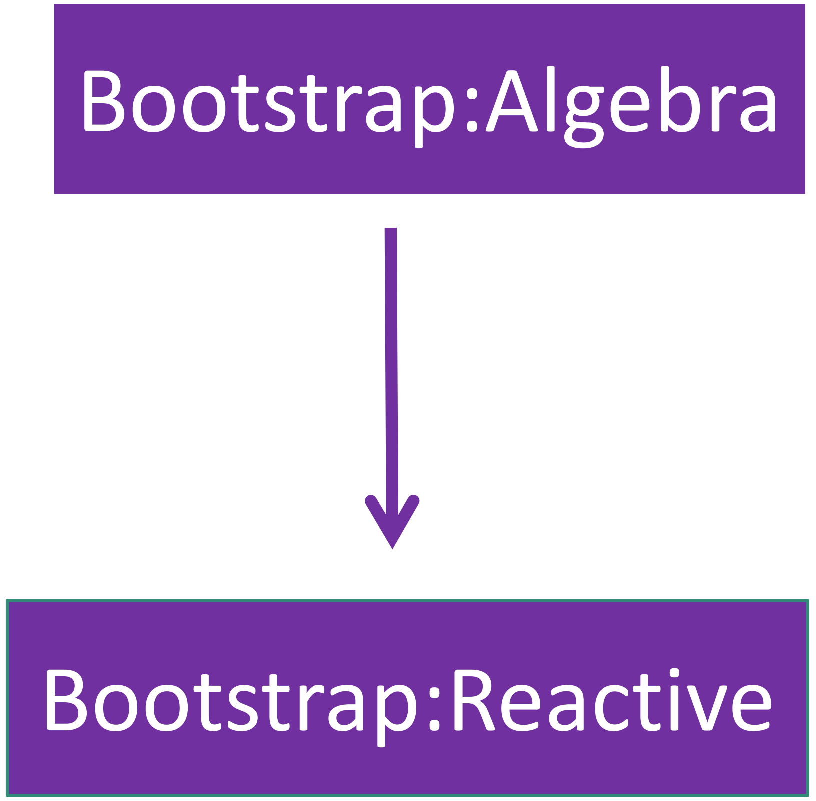
Engineering for Accessibility:
Screen Readers are not Enough

EMMA YOUNDTSMITH

MIDWEST REGIONAL MANAGER

BOOTSTRAP

Bootstrap: A Family of Computing Curricula



Students program a video game using Algebra

Integrated into existing math classes nationally

Students program reactive applications using data structures

Bootstrap: A Family of Computing Curricula

Bootstrap:Data Science

Students answer questions by programming analyses of real datasets

Bootstrap:Physics

Students program interactive simulations of physical scenarios

Bootstrap integrates into existing K-12 courses (Algebra, Physics, etc.) so that all students get exposed to computing

Our Journey to Accessibility

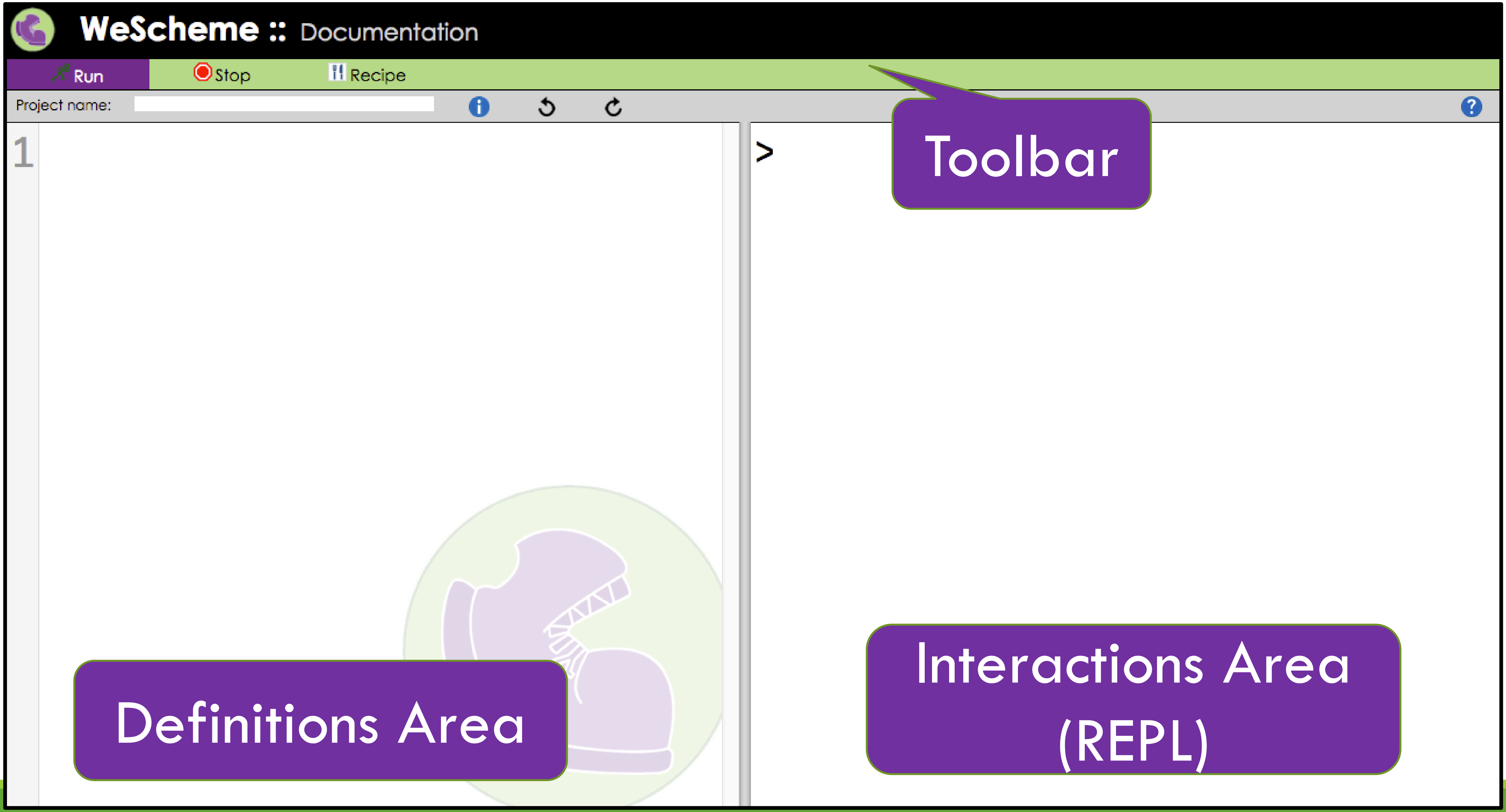
1. User Interface
2. Program Output
3. Writing Programs

Part 1: User Interface

- All functionality must be accessible
- All functionality must be logically organized
- All existing keyboard shortcuts and navigation conventions must be preserved

User Interface

- WAI-ARIA (Web Accessibility Initiative – Accessible Rich Internet Application Suite)


WeScheme :: Documentation
Run
Stop
Recipe
Project name:
1
>
Toolbar
Definitions Area
Interactions Area (REPL)

REPL Interactions

- REPL performs two tasks:
  - Gives you output based on your typed expression
  - Communicates a history of your input
- Need additional keyboard shortcuts so browser can read back previous interactions to the user
- Existing Chatzilla key conventions

Part 2: Describing Program Output

How should screen readers describe the following?

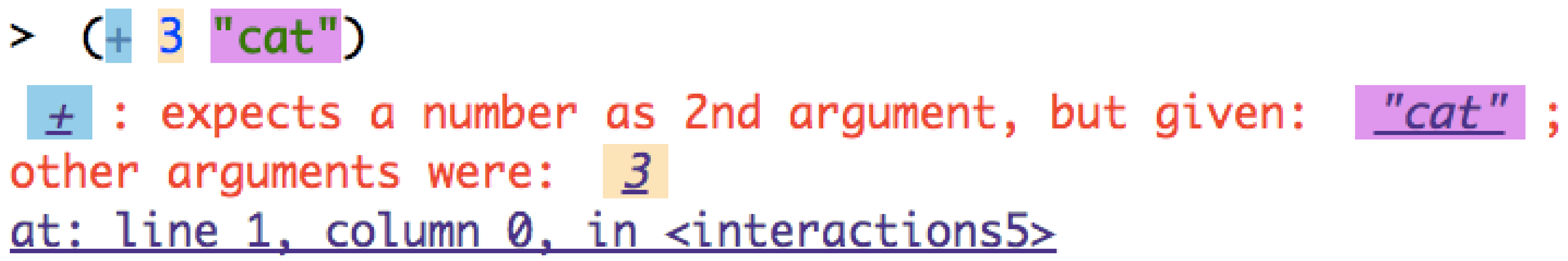
```
3          “3”          “three”
```

What about data structures? Lists? Trees?

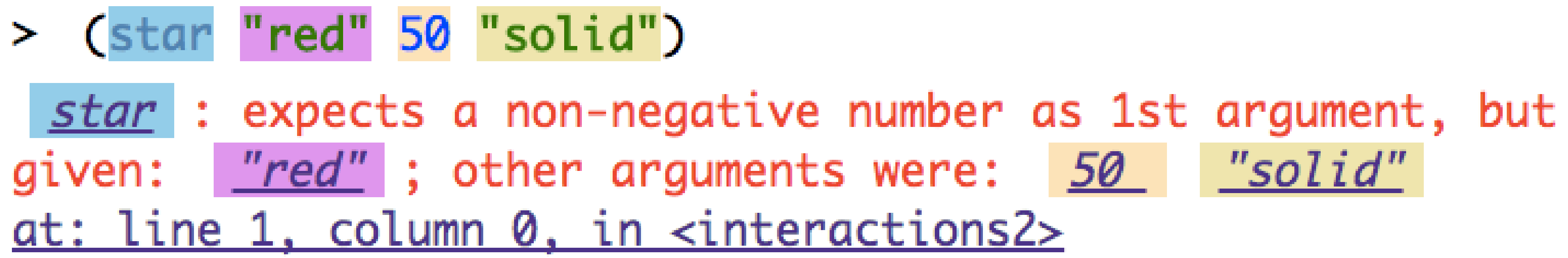
Accessibility at Runtime

- Had to rethink *every level* of our software, and build accessibility into the runtime
- Every data type must describe itself usefully
- Tagged with an ARIA-label
- Only possible because we build *our own* language, compiler, and runtime

Error Messages

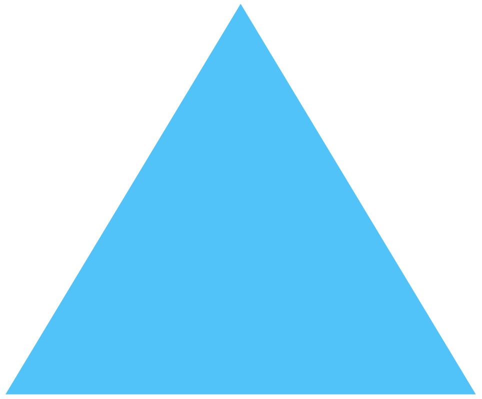


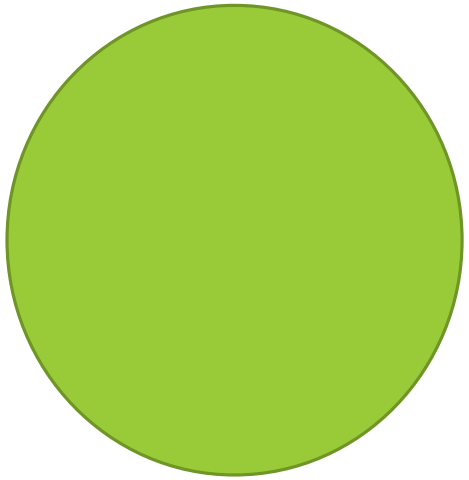
```
> (+ 3 "cat")
+ : expects a number as 2nd argument, but given: "cat" ;
other arguments were: 3
at: line 1, column 0, in <interactions5>
```

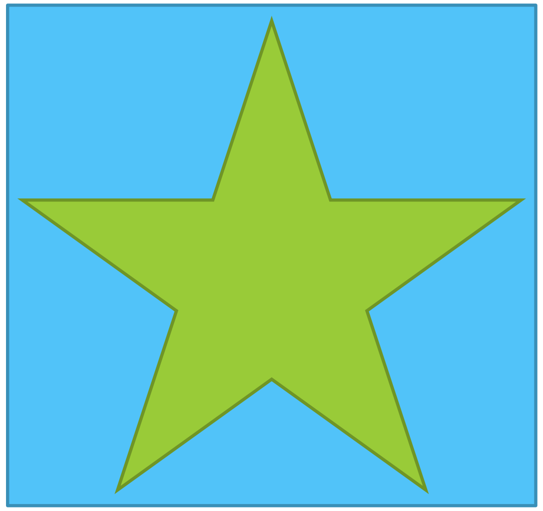


```
> (star "red" 50 "solid")
star : expects a non-negative number as 1st argument, but
given: "red" ; other arguments were: 50 "solid"
at: line 1, column 0, in <interactions2>
```

What about Images?

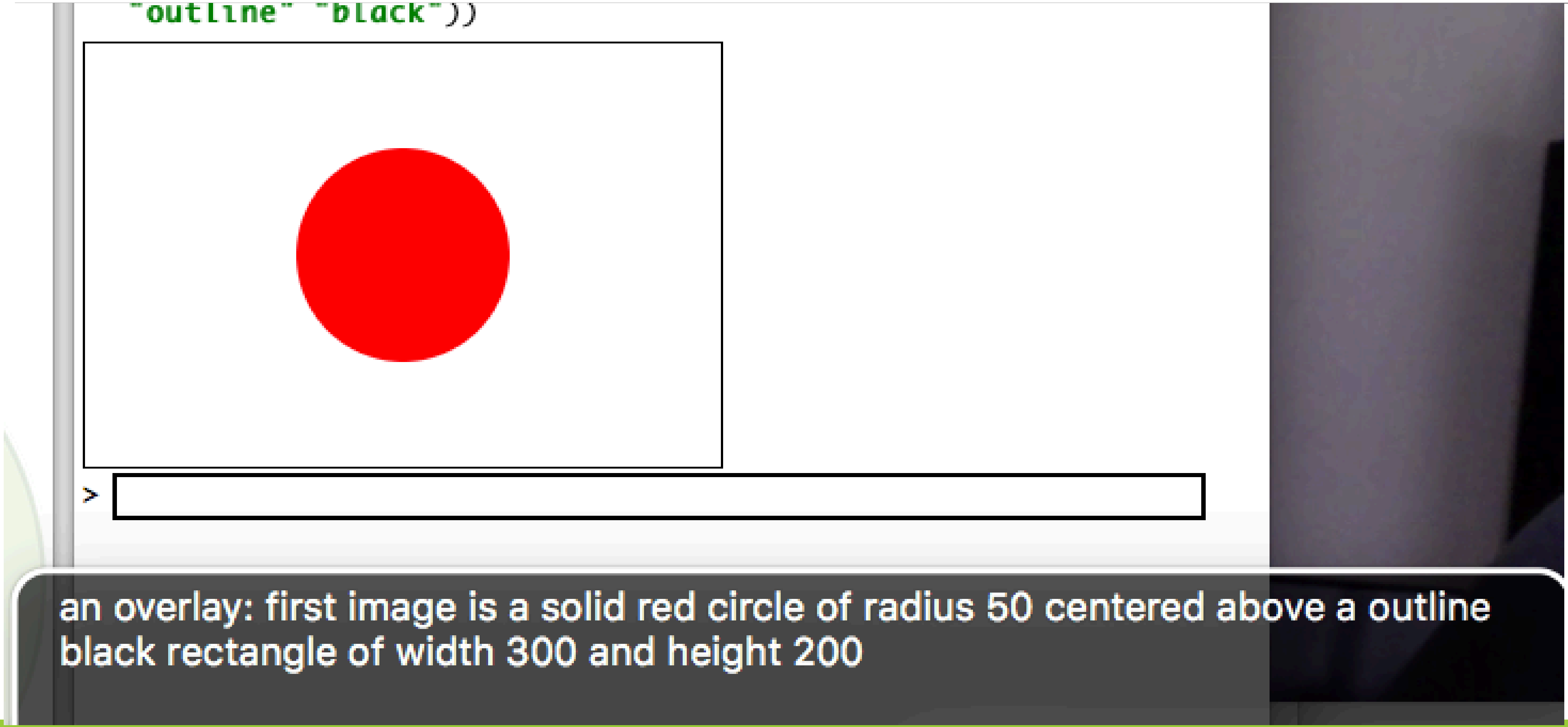






- Scene Graphs preserve the *structure* of images

What about Images?



```
"outline" "black"))
```

```
>
```

an overlay: first image is a solid red circle of radius 50 centered above a outline black rectangle of width 300 and height 200

What about Complex Images?

What about Animations?

- Animations can be lists of images!
- We can look at the structure of the image, the structure of the animation, to be smart about describing how they change

User Testing

- Auburn University - STEM Wars
- Students wrote basic expressions and made images
- Taking a risk: Making Images???
- Programming images provides a formalism for talking about data – engaging, empowering, and fun

Part 3: Writing and Editing Programs

- With 10,000 lines of source code, what is the structure?
- Problem: Screen readers will try and read every symbol on every line

Our Wish for Screen Readers:

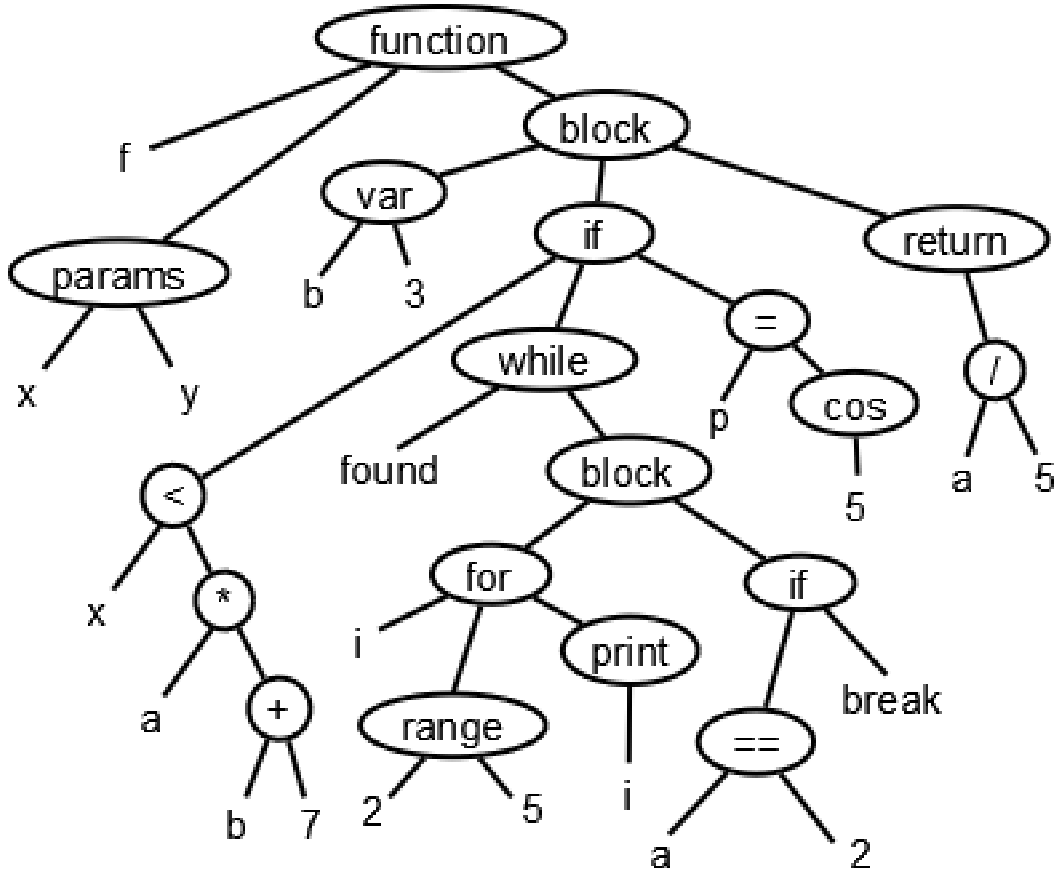
“There are 8 class definitions and 3 function definitions in this file”

“BACKGROUND, a value definition…”

“update-player, a function definition…”

Structure of ALL well-formed programs

- Abstract Syntax Tree
- Parser builds an AST, passes it to the compiler
- We **draw** the AST as a Collection of nodes in an HTML webpage



Exposing the AST as HTML

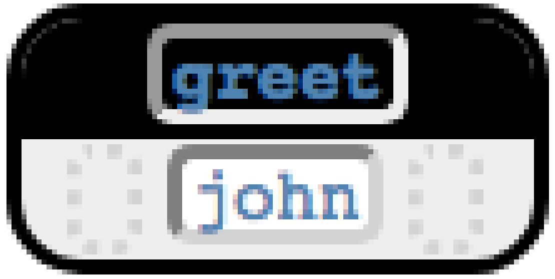
- Screen readers understand structure of a webpage
- Can separate how code is read aloud from the syntax: now we can ***explain*** the code, instead of just reading it

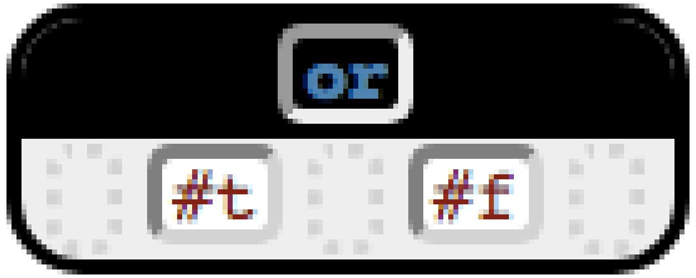
"plus, a function with two arguments…"

"a conditional with four branches…"

Surprise Blocks!

- Unintentionally created a fully accessible, block-based language
- Can collapse, toggle between blocks and text, or cut/paste to nest blocks within other blocks – all with keyboard shortcuts!





Lessons Learned

- Accessibility doesn’t end or begin with a screen reader
- Structure is key
- Don’t reinvent the wheel!

Special Thank You

- Richard Ladner
- Sina Bahram
- Daniela Marghitu
- Paul Carduner

NSF

esa entertainment software association FOUNDATION

AUBURN UNIVERSITY

Google

Thank You

emma@bootstrapworld.org

facebook.com/BootstrapWorld

@BootstrapWorld

@youndtsmith