7 Screen Reader Compatibility

1

Last Week we looked a bit at the technology which can be used to aid the accessibility of people who rely on Screen Readers, and this week, we will be taking a deeper dive into looking at the screen reader side of things.

# What are Screen Readers?

A screen reader is software that interprets what's on a screen—text, buttons, images, menus—and presents it to the user through:

Speech synthesis (spoken output)

Braille displays (tactile output)

Users navigate using keyboard shortcuts rather than a mouse. The screen reader announces content, metadata (like alt text), and structural elements (like headings or form fields).

Popular screen readers include:

JAWS (Job Access with Speech) – commercial, Windows-based

NVDA (NonVisual Desktop Access) – free and open-source for Windows

VoiceOver – built into macOS and iOS

Narrator – built into Windows

# Who Invented Screen Readers?

The story is as inspiring as it is technical.

Ted Henter, a motorcycle racer, lost his sight in a car accident in 1978. He retrained as a programmer, and co-developed JAWS, one of the first widely adopted screen readers.

Early screen readers like Versatile Speech in the 1980s were hardware-based, converting text to speech via synthesizers.

Many innovations came from blind programmers themselves, building tools their communities needed when mainstream tech lagged behind. This makes a lot of sense since, a blind person would truly understand the needs of another blind person much better than anyone else could.

How Do Screen Readers Work?

They act as a bridge between the operating system and the user, parsing:

UI elements (buttons, labels, menus)

Text content

Accessibility metadata (roles, states, descriptions)

They rely on accessibility APIs like:

MSAA/UIA on Windows

AT-SPI on Linux

AXAPI on macOS

These APIs expose widget information to assistive technologies.

# Implementing Screen Reader Support in Python + Tkinter

Best Practices

Use Label widgets with clear text: Screen readers can often detect these.

Associate labels with inputs using layout and naming conventions.

Avoid custom-drawn widgets unless you manually expose accessibility info.

What You Can Do in Your Own Tkinter Projects

Here are coding strategies to make your GUI more accessible:

1. Use Clear, Text-Based Widgets

Stick to standard widgets like **Label, Button, Entry**, and **Text**

Avoid custom canvas drawings unless you add narration manually.

2. Label Your Inputs Clearly

Use **Label** widgets placed near **Entry** fields.

Example:

Label = tk.Label(root, text="Username:")

entry = tk.Entry(root)

label.grid(row=0, column=0)

entry.grid(row=0, column=1)