06 Interaction Design

36-721 Statistical Graphics and Visualization

Jerzy Wieczorek

9/17/15

Last time

- Graphic Design overview, elements, principles
- Inkscape example
- HW 2

Today

- Examples of interactive graphics
- Interaction and navigation methods
- Interaction Design principles:
  consistency, visibility, constraints, feedback
- Designing an interactive graphic
- Creating interactive apps with R and Shiny
- Creating interactive graphics with D3.js

Today

Follow along:

- R code: `ui.R`, `server.R`
- D3.js code: `06_IxnDesign_D3_Bars.html`, `06_IxnDesign_D3_BarsLoadCsvData.html`, `06_IxnDesign_D3_Dataset.csv`

Examples of interactive graphics

- New York Times, **Why Is Her Paycheck Smaller?**
- Washington Post, **A global look at cardiac risk factors**
- Forbes, **American Migration**
- War on Ice, **Player Hextally**

Shneiderman's mantra

Like the Graphic Design examples, these contain:

- Headings for structure, body text for detail
- Richly informative, complementary graphs
- Annotations: interesting features; how to read the graph; data sources

but also tend to:

- Start with sensible defaults
- Make use of Shneiderman's mantra

"Overview first, zoom and filter, then details-on-demand."

—**Ben Shneiderman (1996)**

Interaction and navigation methods

- Search, input, set, filter
- Sort, arrange
- Zoom
- Scroll, pan
- Open and close windows, change tabs
- Hover, mouseover, tooltip
- Animation

Animation

Tversky, Morrison, and Betrancourt (2002):
animations should be

- Slow enough to track
- More schematic than realistic
- Annotated to direct the viewer's attention to crucial changes and relations

Interaction Design principles

- Consistency
- Visibility
- Constraints
- Feedback

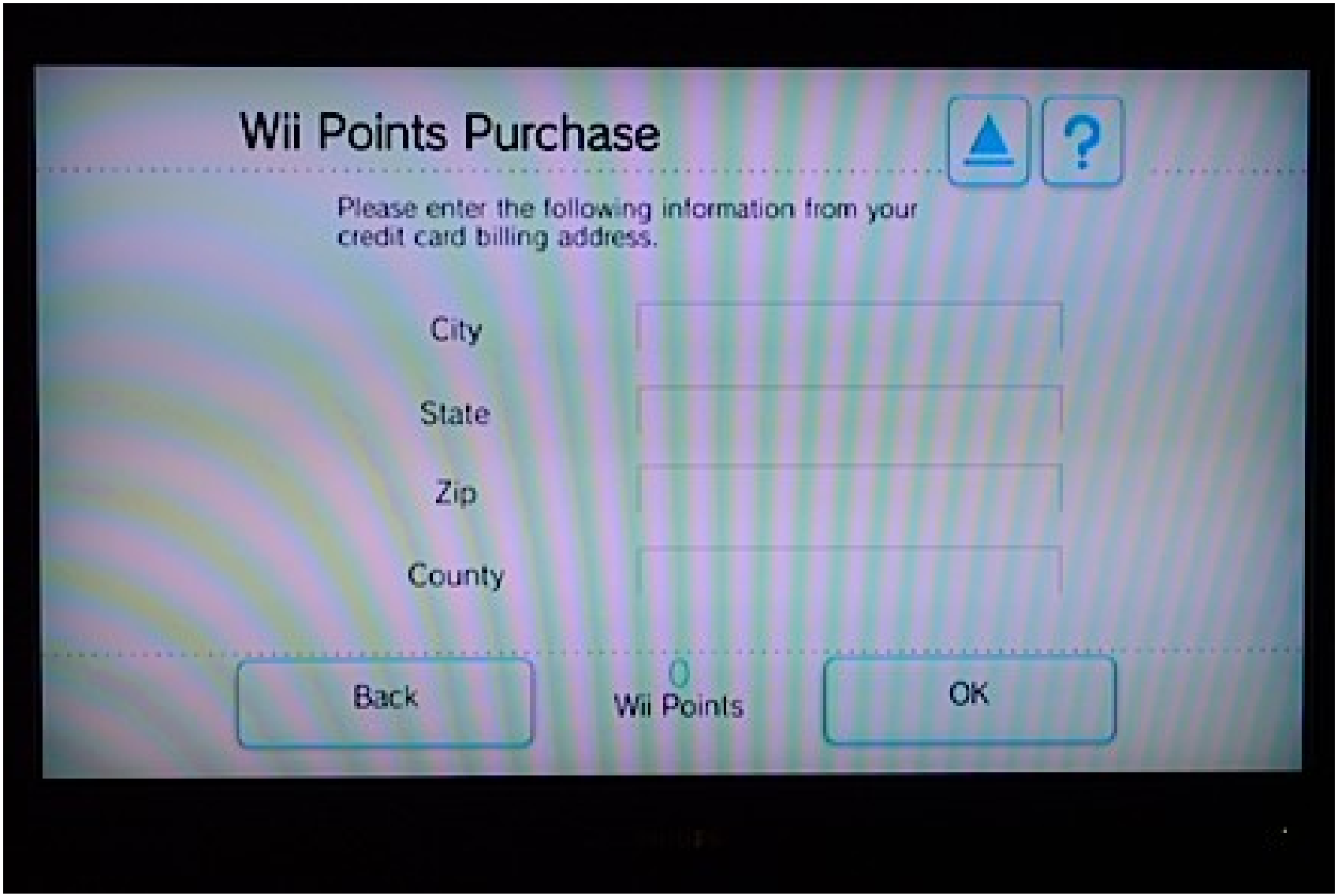
Consistency

Of navigation, and design, and graphics

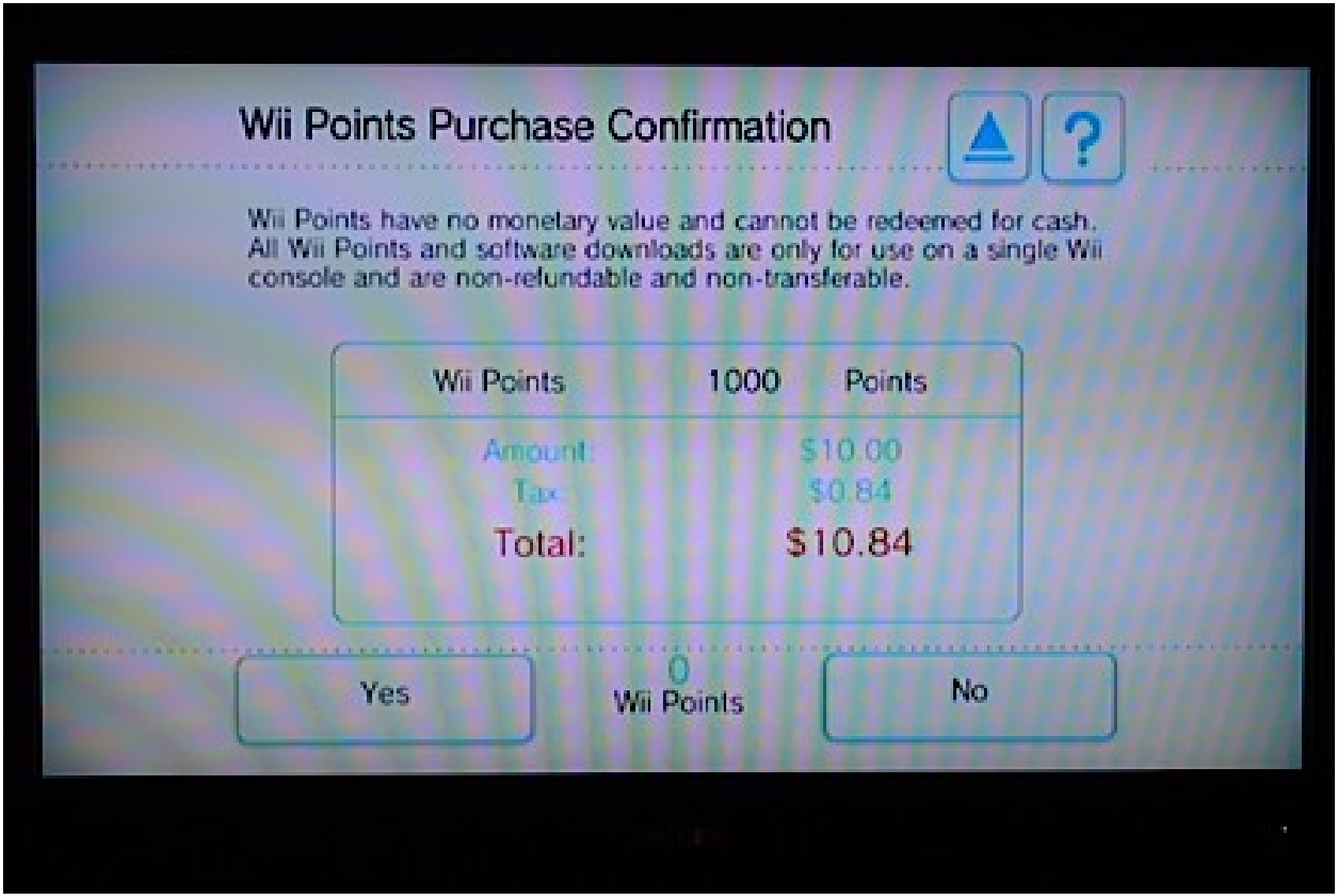
- Use same locations for analogous buttons from page to page
- Consistent visual style

Consistency: bad example

Consistency: bad example



Consistency: bad example



Visibility

Of controls, and of data / annotation layers

- Use **affordances**: make it intuitively clear which elements are controls and what they do
- Don't hide important controls or annotations when the view changes

Visibility: bad example

Constraints

For error prevention and for clarity

- Prevent errors before they happen: consider slider (prevent entry of negative numbers) vs text box (allows entry but then gives error message)
- Avoid feature creep: limiting the possible interactions keeps your design from overwhelming users

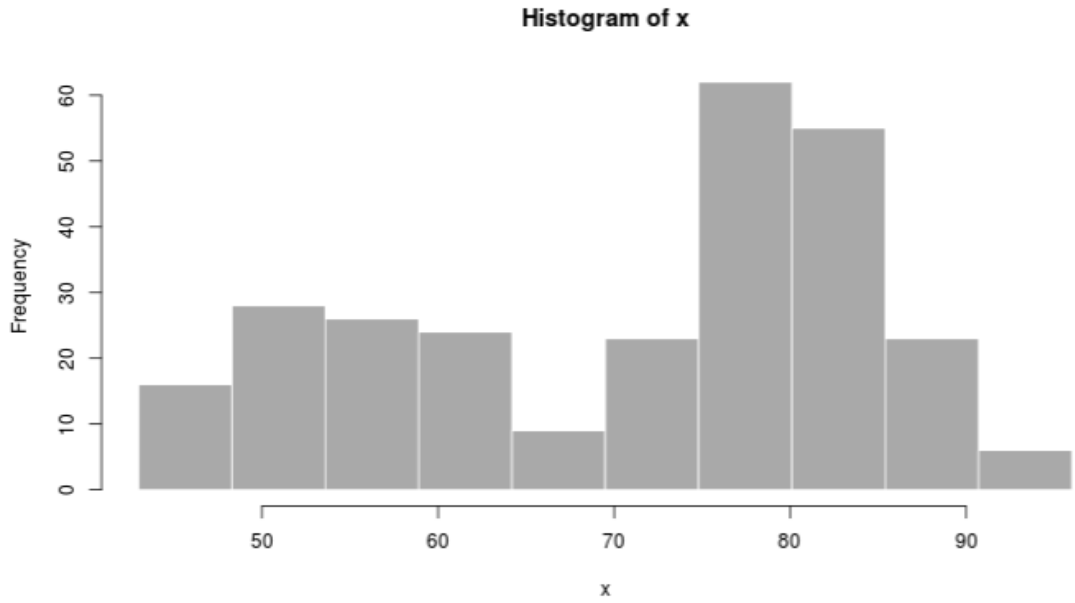
Constraints: good and bad examples

Hello Shiny!

Number of bins:

1 10 50

1 6 11 16 21 26 31 36 41 46 50



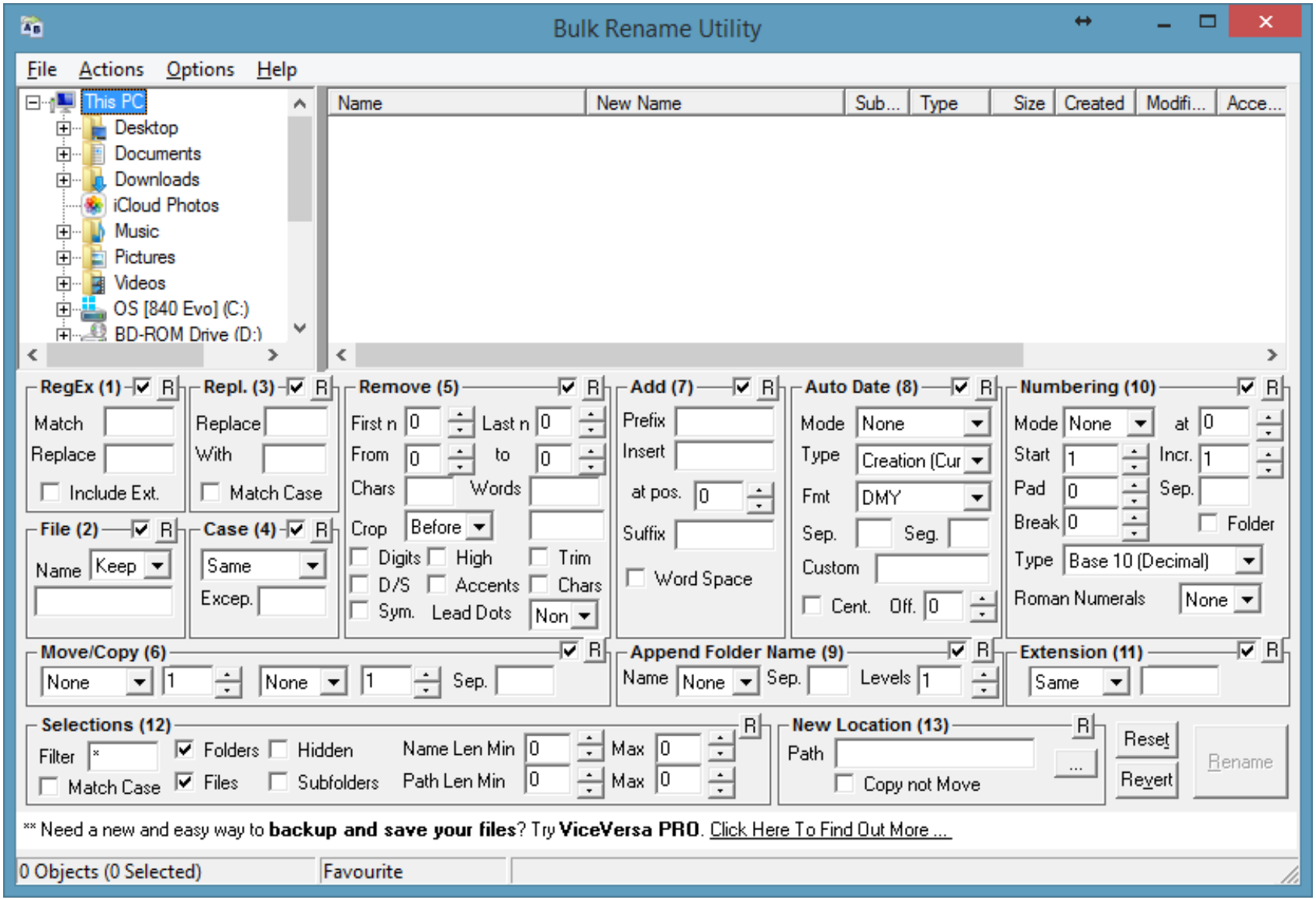
Hello Shiny!

Number of bins:

-5

Error: length must be non-negative number

Constraints: bad example



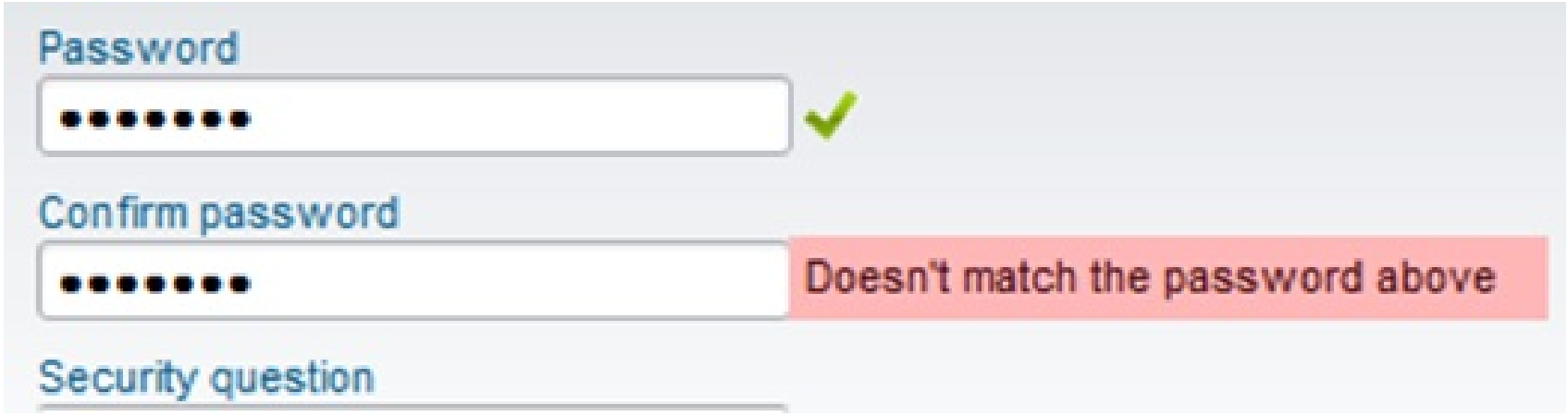
(via Twitter)

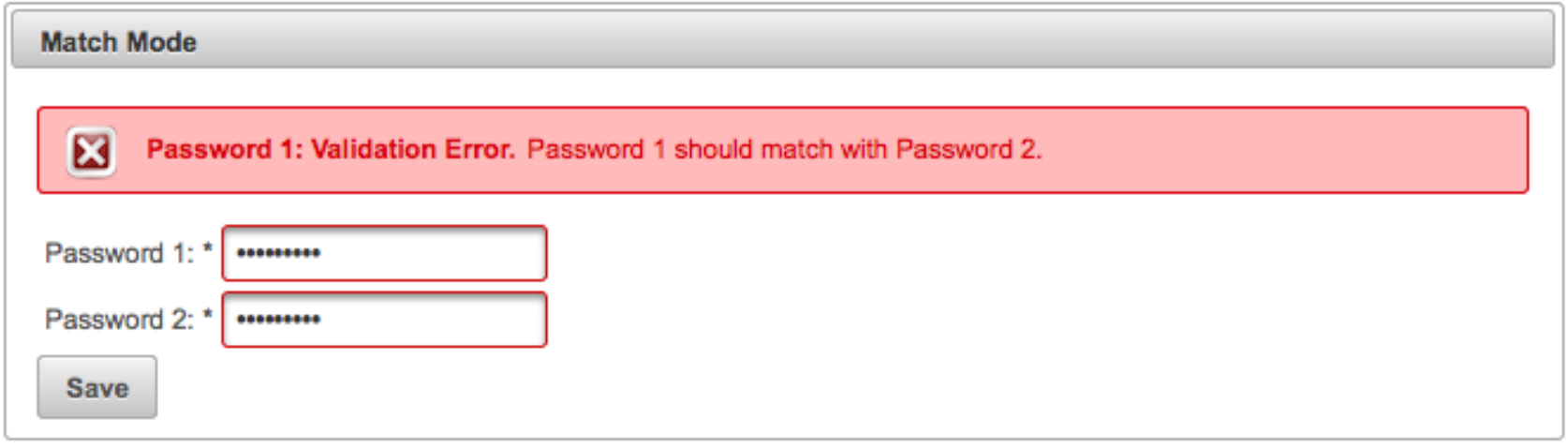
Feedback

Confirm that user's actions had an effect

- React to actions: button changes when clicked, points are highlighted on mouseover
- Complement, don't interrupt: nicer to see "Passwords do not match" as you type, rather than after submitting form

Feedback: good and bad examples





Interaction Design in practice

Again, like with Graphic Design:

- Find your story or message
- Choose the right graphic forms and interactions to support it
- Sketch layout ideas on paper
- Choose a visual style
- Try out on computer

Usability

For large / serious projects:

- Define client's needs, or study target audience; invent user personas and design for them
- Avoid feature creep (adding too many features, making the overall experience harder to navigate)
- Usability testing at many stages: paper prototypes, computer prototypes, full implementation

Shiny

Install `shiny` package from within R

Create `ui.R` and `server.R` files, for the front-end (what to display) and back-end (how to process it in R)

RStudio's **Shiny tutorial** is free online

Hosting Shiny apps

On your own computer:

- Put your app's `ui.R` and `server.R` files in the same folder
- Start R and load package with `library(shiny)`
- Run your app with `runApp("/Path/To/Folder")`

Online:

- **shinyapps.io**
- Host on a Shiny server, like **CMU stat department's**

D3.js

Download zip file from the **D3.js website**

D3 philosophy resembles `ggplot2`
(which data maps to which visual features?)
. . . but much more low-level: no default axes, legends, etc.
. . . and much more flexible, including interaction and transitions

D3 requires a good understanding of HTML, CSS, SVG, and JavaScript

Scott Murray's **website tutorial** and his book **Interactive Data Visualization for the Web** are excellent intros, free online

Hosting D3.js apps

On your own computer, may need to host a local HTTP server due to browser security restrictions.

Use Python as a HTTP server, if you have it. . .

Or set up R as a HTTP server with **`servr` package**:

- Put your app's HTML and other files in the same folder with `d3.js` file
- Start R and load package with `library(servr)`
- Run your app with `httd("/Path/To/Folder")`

For next time

- We'll discuss the field of data visualization research: some good people and journals to know, and open problems in the field
- HW 3 (Grammar of Graphics) due Saturday at 5pm, through Blackboard
- Projects 1 (Graphic Design) and 2 (Interaction Design) will be posted soon; due on future Saturdays