Debugging

What is debugging?
Debugging steps
Debugging techniques

What is type of error is this?

A. Syntax Error
B. Runtime Error
C. Logic Error
D. Other Error
E. Not an Error
Finding Errors in your Code: Debugging

- A bug is an error in your program
  - it won’t run
  - It crashes
  - it doesn’t do what you want
- Debugging is the process of finding these errors
  - Some errors are easier to find
  - Others can be hard to isolate, even hard to recognize
Debugging Steps

1. Recognize that a bug exists
2. Isolate the source of the bug
3. Identify the cause of the bug
4. Determine a fix for the bug
5. Apply the fix and test it
Debugging Strategies

- Check for runtime warnings and errors
- Rule out common errors
- Add extra debugging output (like print)
- Force your program into certain states
- Isolate code might cause the problem
- Slow down code execution (change frameRate)
- Step through your code with an interactive debugger
- Clean up code, rewrite code
- Talk through the problem with someone
- Take a break

Check for Runtime Warnings and Errors

- Open the JavaScript console in your browser!

  **Chrome**
  - `Cmd + Option + J` (Mac) or `Ctrl + Shift + J` (Windows/Linux)
  - View > Developer > JavaScript Console

  **Firefox**
  - `Cmd + Shift + J` (Mac) or `Ctrl + Shift + J` (Windows/Linux)
  - Tools > Web Developer > Browser Console

  **Safari**
  - `Cmd + Option + C`
  - Develop > Show JavaScript Console
  - (need to enable “Show Develop Menu...” in Advanced tab of Preferences first)
Rule out Common Errors

accidental infinite loop

\[ \times \text{ for (let x = width; x > 0; x++)} \{ \]

precision problems with ===

\[ \times \text{ if (f === 5) } \{ \text{// f might not be whole number} \]

unintentional variable shadowing

\[ \times \text{ let x = 0; function draw() } \{ \text{let x = x + 1;}} \]

Rule out Common Errors: Missing/Extra Brackets

- Use the built-in autoformatting function frequently
  - CMD + T (in Processing IDE)
  - SHIFT + TAB (in p5 Web editor)
- If indentation doesn’t look right, you have a syntax error
Add Extra Debugging Output: print

- Print value of a variable to console
  
  ```
  print("x: ", x); // prints "x: 123" (if x is 123)
  ```

- Print a message when conditional is true
  
  ```
  if (mouseIsPressed) {
      print("mouse is pressed");
  }
  ```

- Use print to print a message at different places
  
  ```
  print("HERE");
  print("draw start");
  ```

---

```javascript
function draw() {
    print("draw start");
    background(220);

    if (mouseX < th) {
        print(mouseX, " T"); // debug output
        ...
    } else {
        print(mouseX, " F"); // debug output
        ...
    }

    print("HERE");
}
```
Add Extra Debugging Output: print for loops

- Very useful for debugging loops

```javascript
for (let x = 20; x < width; x += 10) {
    print("loop x:", x);
    ...
}
```

Add Extra Debugging Output: Shapes

- Use shapes to “draw” debugging output in the canvas

```javascript
line(th, 0, th, th); // debug: line shows threshold
if (mouseX < th) {
    ...
} else {
    ...
}
```
**Force your program into certain states**

- Set global variables to fixed values to verify expected behaviour

```javascript
mouseX = 49; // debug, force mouseX to be value
if (mouseX < d) {
    ...
} else {
    ...
}
```

**Isolate Code That Might Cause the Problem**

- Comment out sections of code and see if bug persists

```javascript
// draw and update the ship
function shipUpdate() {
    // move the ship
    shipX = mouseX;

    // keep the ship on canvas
    // shipX = constrain(shipX, shipWidth/2, width - shipWidth/2);

    // draw the ship
    shipDraw(shipX, shipY);
}
```

**Command + / to Comment/Uncomment**
Slow Down the Execution

- Make your program run artificially slow to see what happens on individual frames

```javascript
function setup() {
  ...

  // draw runs at only one frame-per-second
  frameRate(1);
}
```

Step through your code with an interactive debugger

- Guide to using Chrome Debugger
  - [https://javascript.info/debugging-chrome](https://javascript.info/debugging-chrome)
How to Avoid Bugs

- **Plan** your program on paper first
- Use **user-defined functions** to make your code modular
- Use **user-defined functions** to make your code testable
- Don’t guess, **think about why** you’re adding each statement
- Keep your code “clean”:
  - remove variables that aren’t used
  - remove conditionals or relational expressions that aren’t needed
  - look for ways to avoid duplicated code
- Keep your code understandable:
  - use good variable names
  - comments to explain what a section of code should be doing
- You should be able to **trace through your whole program** without running it: if you can’t, it will be very hard to debug