Variables

Declaration, Initialization, Assignment

Memory

Chapter 4, Examples 4-1, 4-2, 4-3, 4-4, Robot 2 (p. 54-56).
Chapter 8, Examples 8-1, 8-2, 8-3, 8-4, 8-5, 8-7, 8-8, 8-9 8-10, 8-11.

variables make code easier to change and easier to read

Starter: https://editor.p5js.org/cs105/sketches/-DpiDsI5s
https://editor.p5js.org/cs105/sketches/Y48n3yKis
Variable

- A symbolic **name** used to refer to a **value**.
  - The value may change, but the symbolic name doesn’t.

**Spreadsheet Analogy**
Spreadsheet Analogy

![Spreadsheet Image]

Computer Memory and Variables

![Diagram Image]
We're not “solving for” variables ...

\[
x + 3 + \frac{(2x - 12) - 1}{3} = 1
\]

\[
12 \left[ \frac{x + 3}{4} + \frac{(2x - 12) - 1}{3} \right] = 12 \[1\]
\]

\[
3(x + 3) + 4(2x - 13) = 12
\]

\[
11x = 55
\]

... we’re “assigning values to” variables (and then using variables in place of numbers).

Using Variables

- We don't always know the value (or even care what the exact value is), we just refer to it with a name.

  TWOPY

  mouseX

  donutSize
**Variable Types**

1. **Built-in (or “System”)**
   - e.g. width, mouseX
2. **Constants**
   - e.g. TWO_PI, BEVEL
3. **User-defined**
   - e.g. donutSize, bgShade

**Math**

“Math can be an important aspect of programming, but it’s not necessary to be good at math to understand or enjoy programming.

There are as many styles of programming as there are people who program, and it’s the decision of the individual to utilize or ignore math as they prefer. People who enjoy math often write programs to visualize equations or take delight in exploring phenomena such as fractals. People who struggled with math in school sometimes find they enjoy and understand it better when it is applied to form and motion."

(from Reas and Fry, *Processing Handbook*)
(math expression demos)

+ add

- subtract
  ... negative numbers

* multiply

() brackets

/ divide
  ... divide by 0 makes Infinity

% modulo
  (remainder after division)

Order of operations:
B E DM AS

(squares1)

using math expressions with built-in variables

https://editor.p5js.org/cs105/sketches/IJTBdu4rj
What does this code draw?

```javascript
function setup() {
  createCanvas(100, 50);
}

function draw() {
  rect(width - 5, height - 5, 10, 10);
}
```

(squares2) using user-defined variables

https://editor.p5js.org/cs105/sketches/wUnLfEI9Q
Creating (i.e. “Declaring”) User-defined Variables

### JavaScript Data Types

#### Number
- whole numbers like -5, 0, 12, ...
- decimal numbers like 0.123, -10.1, 32.0, 0.0

#### Boolean
- true or false

#### String
- any literal text, e.g. "hello", "123", "yoko ono", etc.

#### Object
- an Array
- a collection of information you define
- a p5 image
- a p5 colour
  ... (many more)
Declaring versus Assigning Variables

- Declaring:
  
  ```javascript
  let donutSize;
  ```

- Assigning:
  
  ```javascript
  donutSize = 80;
  ```

- Declaring and Assigning (aka "Initializing"):
  
  ```javascript
  let donutSize = 80;
  ```

let, const, and var

- JavaScript has three keywords to declare variables:
- **let** are for "user-defined" variables
  - variables declared with let can be re-assigned
  - In this course, we will almost always use let
- **const** are for "constants"
  - These kinds of variables can only be assigned once
  - Useful to hold a value that will never change in your program
    (so you make sure you don’t re-assign it by accident)

**var**

- **Do not use var**
  - Most developers agree var should be avoided
  - In this course, you'll lose marks if you use var
let vs const
where to declare
assigning multiple times
++ -- += -= *= /= shortcuts
implicit "conversion" of types

What does this code draw?
What does this code draw?

```javascript
let a = 50;
function setup() {
  createCanvas(100, 50);
  strokeWeight(20);
a = 100;
}
function draw() {
  point(a, height / 2);
}
```

Functions that Return a Value

- Some function calls **return a value**: `random`, `min`, `max`, (and many more we'll see later)

- Assign the "result" of these functions to a variable:
  ```javascript
  // assign a random number from 0 to 99.99999 to x
  x = random(100);
  // assign whichever is smallest, 50 or mouseX
  x = min(50, mouseX);
  ```

- Use the “result” of these functions as a function argument:
  ```javascript
  // draw a point at a random x and y location
  point(random(width), random(height));
  // mouseY changes weight, but only up to 25 pixels
  strokeWeight(min(mouseY, 25));
  ```
**min(a, b)**

Determines the smallest value in a sequence of numbers, and then returns that value. `min()` accepts any number of `Number` parameters.

**Syntax**

```javascript
min(a, b)
```

**Parameters**

- `a` *Number*: Number to compare
- `b` *Number*: Number to compare

**Returns**

- *Number*: minimum Number

---

**p5**

```javascript
function draw() {
  ...
  // set w to be the smallest of
  // mouseY and 25
  let w = min(mouseY, 25);
  strokeWeight(w);
  ...
}

function draw() {
  ...
  // set to be the smallest of
  // mouseY and 25
  strokeWeight(min(mouseY, 25));
  ...
}
```

Starter: [https://editor.p5js.org/cs105/sketches/0QZM7wHns](https://editor.p5js.org/cs105/sketches/0QZM7wHns)
What does this code draw?

```javascript
let w = 50;

function setup() {
  createCanvas(100, 50);
  strokeWeight(20);
  w = 0;
}

function draw() {
  background(200);
  w = 100;
  point(w, height / 2);
}
```

What does this code draw?

```javascript
let z = 200;

function setup() {
  createCanvas(100, 50);
  strokeWeight(20);
}

function draw() {
  background(200);
  z = min(z, 50);
  point(a, height / 2);
}
```
Review of Variables (So Far)

- Definition of a variable
- Relationship to computer memory
- Variable types
  - JavaScript Data types
- Variable Declaration vs. Variable Assignment
- let, const, and var
- Mathematical operators
- BEDMAS

Why User-defined Variables?

- To reuse the same value in your program
  - e.g. draw many squares of same size
- To make it easy to change a value later
  - e.g. make squares larger
- To make your code easier to understand
  - e.g. easier to read since “squareSize” means more than "10"
- Keep track of something that changes
  - e.g. click number, animated object position
Program State

- What line of the program is executed next?  
  (i.e. Sequential Control Flow)

- Current settings of all drawing attributes 
  (e.g. fill, stroke, strokeWeight, colorMode, etc.)

- Current values of all variables
  (use variables to “save” and “update” state of things)

Using Variables to “Save State”

- Example: What kinds of state are there in a game?
use mouse and key presses to control circle sizes, all using user-defined variables

```javascript
function draw() {
    ...
    ellipse(width / 2, height / 2, redSize, redSize);
    ...
}

function keyPressed() {
    redSize = redSize + 3;
}
```

Starter: https://editor.p5js.org/cs105/sketches/7pCJvLk2

https://editor.p5js.org/cs105/sketches/Lf7UwAxj7

The Animation Process

- http://youtu.be/M2ORkIrHUbg
animation

adding a bit to a variable each frame

assigning variable when mouse button event

Starter: https://editor.p5js.org/cs105/sketches/0a9XqYHo

https://editor.p5js.org/cs105/sketches/Mcj4bsUNJ

Trace of variables

- see: “03b Animation (trace).pdf”
How Assignment (=) Works with Variables

A single equals sign = means:
“assign calculated value on the right to the variable on the left”

- Only values of variables on right side are used.
- No variables are “linked.”
- Two variables do not share a location in memory.

```javascript
let a = 10;
let b = a;
b = b + 1;
```
Numerical Representation

- Binary Numbers: 1 and 0
- How numbers are represented in a computer
  - Bits and Bytes
- Limits of storage:
  - numeric overflow
  - precision

What is this binary number in decimal?
Numerical Limits

- There are limits to what numbers can be stored in a computer.

**Numeric Overflow**
https://editor.p5js.org/cs105/sketches/iPADgbyJh

```javascript
let a = 9007199254740982;

function setup() {
    frameRate(2);
}

function draw() {
    a = a + 1;
    // watch the output!
    print(a);
}
```

**Precision Errors**
https://editor.p5js.org/cs105/sketches/MJevdLhI3

```javascript
let a = 1;

function setup() {
    frameRate(1);
}

function draw() {
    a = a + 0.1;
    // watch the output!
    print(a);
}
```