Impact Lab 2025: Programming Fundamentals
Lecture 1: Introduction

Summer 2025

School of Computing and Data Science

Wentworth Institute of Technology

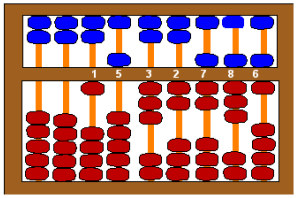
Me

- Prof. Micah Schuster
- Email: schusterm@wit.edu
  - Start with [<insert course name/number] in subject
- Office: Dobbs 204
- Formal Training: Nuclear Physics, HPC

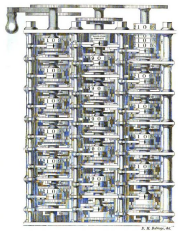
A Bit of History

What do you think was the first computing device?

Charles Babbage and Ada Lovelace, in the 1800's are considered the pioneers of computing

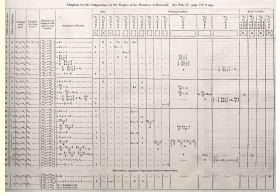


Abacus: 2700-2300 BCE



Babbage's Difference Engine

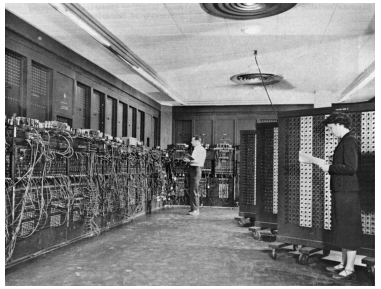
Lovelace's First Published Computer Algorithm



A Bit of History

Computers from the 1920's:

From the 40's on, there were many architectures and styles of computers.

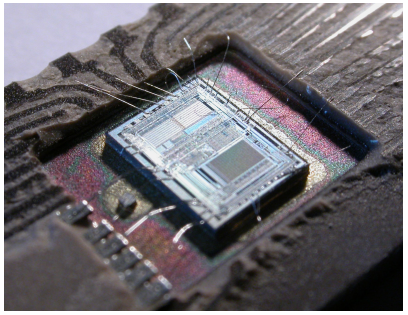


ENIAC - 1946

Programmed using patch cables and switches

A Bit of History

The development of the integrated circuit by Jack Kirby and Robert Noyce in the 1960's led to an explosion in computing for general purposes.



Intel eight-bit micro controller IC

Nowadays, we have very powerful, multicore, CPUs that can perform billions of floating point operations every second.

What Makes a Computer?

Hardware

Physical components → CPU, RAM, HDD

Wide variety of manufactures → Dell, Apple, Seagate

Abstracted to a simple set of ideas for computer science → Memory, Processor, Screen, Read/Write

Software

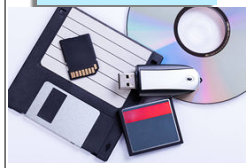
Programs (Instructions) → Operating Systems, Keynote, Mario Kart

Focus of the course

High Level Hardware Overview

Input Devices →

Storage Devices →



Processor (CPU)

Main Memory (RAM)

→ **Output Devices**

Topics for Today

- **P5.js**
- Shapes and Drawing
- Color
- Errors and the Console
- Comments
- Variables

What is p5.js (and JavaScript)

Not that easy to separate (at this point)

JavaScript is the language we will be using. It gives us the **syntax** for how we should write code.

p5.js has a lot of tools that we can use to draw on the screen. It is a **library**.

You can view what p5.js can do by looking at the **documentation**:

https://p5js.org/reference/

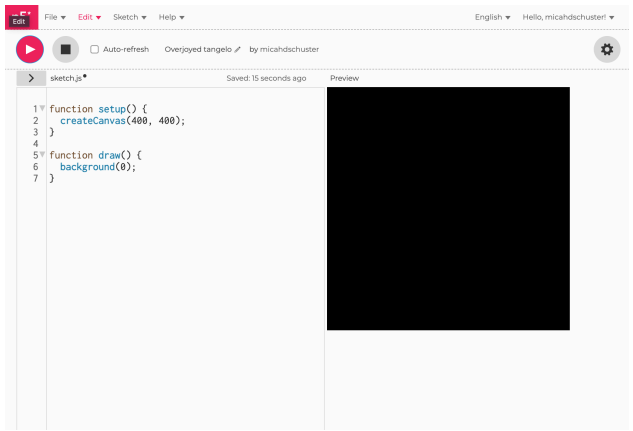
Wentworth
Computing & Data Science

p5.js Web Editor

editor.p5js.org

You'll need to signup for the website to save your code.

You can login with a Google account too!



Wentworth
Computing & Data Science

Sharing your p5.js sketch

You can share your sketch with anyone you want!

Go to File -> Share

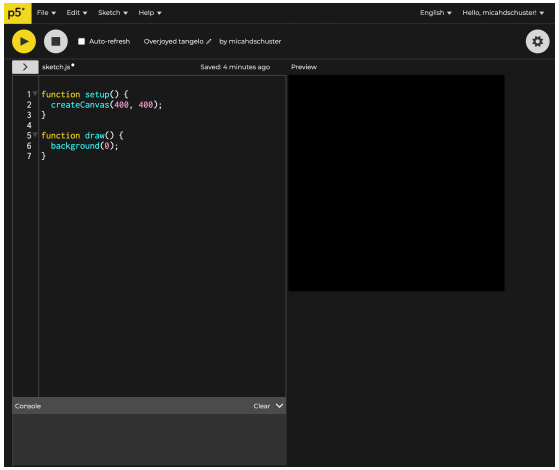
The Fullscreen link will display your sketch (program) in a browser to anyone you send it to.

Wentworth
Computing & Data Science

High Contrast Theme

Click on the gears on the right and choose High Contrast (or Dark) for the theme.

I'll keep mine as white so it is easier to read on the projector.



Wentworth
Computing & Data Science

Topics for Today

- P5.js
- **Shapes and Drawing**
- Color
- Errors and the Console
- Comments
- Variables

Program Instructions

Let's look at some of our starter code:

```
createCanvas(400,400);
```

Instruction (Function Name)

Arguments

Semicolon

What if we change the 400s in the argument list?

You'll need to rerun the code: ▶

Or check "Auto-refresh"

Program Instructions

What about the other stuff?

There's lots of:

- parenthesis
- curly brackets
- highlighted words

```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(0);
}
```

We're not going to worry about all of it yet, but it is worth knowing that there are some special words, called **keywords**, that mean something specific in the language.

Primitives

Let's start with some simple shapes. These are called **primitives**.

Anything we want to draw to the screen goes in the **`draw`** function.

```
rect(x,y,width,height);
```

Pick some values for the parameters and see what happens when you add this to your sketch.

What if you add another number to the rect arguments?

Shape

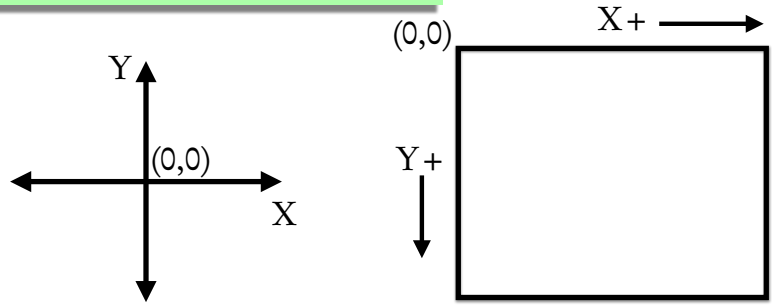
2D Primitives

```
arc()
ellipse()
circle()
line()
point()
quad()
rect()
square()
triangle()
```

These are in the documentation!

How `rect()` Works

What is our coordinate system?



Cartesian

p5.js Canvas

Start with this rect: `rect(100,50,25,100);`

Play around with the values.

Exercise: Drawing a Drawing

Use rectangles and other primitive shapes to create a drawing.

Use the documentation to help you figure out how to draw other shapes.

Let's look a some of the other documentation together.

Other than the primitives, functions like `rectMode()` can be very useful.

What if you change the order of the primitive shapes in your code?

Extension: What about Color?

Tidy Code

White space is any space in your code created by spaces, tabs, enters, etc.

It doesn't affect how your code runs!

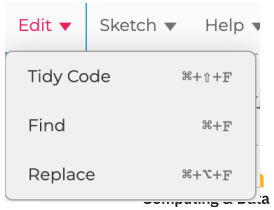
We do have style guides though.

The p5.js web editor has a handy feature to tidy up your style!

```
function draw()
{
background(  0  );
}
```

These are the same

```
function draw() {
  background(0);
}
```



Topics for Today

- P5.js
- Shapes and Drawing
- **Color**
- Errors and the Console
- Comments
- Variables

Color

Why is our background black?

How does color work on the computer?

R: Red

G: Green

B: Blue

To create a digital color, we mix different amounts of red, green, and blue.

background() takes a color as the argument!

```
background(220,0,200);
```

220 units of red
0 units of green
200 units of blue

Color

```
background(0,0,0);
```

Black

```
background(255,255,255);
```

White

One argument assumes grayscale

Unlike paint, mixing max amounts of all the colors gives us white and no color is black.

It's like mixing light, not paint.

All the color ranges are 0-255 (256 total)

Color Functions

```
background(0,0,0);
fill(0,0,0);
stroke(0,0,0:
```

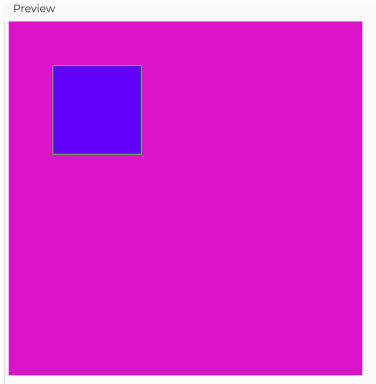
Our canvas is defined by `background()`.

`fill()` will fill the interior of our shapes.

`stroke()` will color the outline.

```
function draw() {
  background(220,0,200);

  fill(100,0,255);
  stroke(0,255,0);
  rect(50,50,100,100);
}
```



Order matters!

Set the color for what you are **about** to draw

Color Functions

```
background(0,0,0);
fill(0,0,0);
stroke(0,0,0:
```

Our canvas is defined by `background()`.

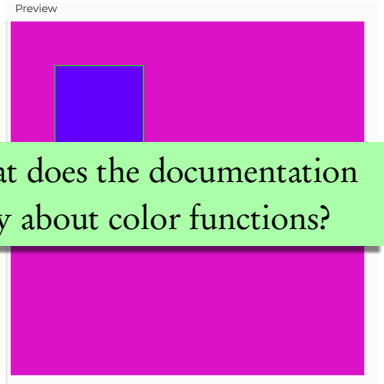
`fill()` will fill the interior of our shapes.

`stroke()` will color the outline.

```
function draw() {
  background(220,0,200);

  fill(100,0,255);
  stroke(0,255,0);
  rect(50,50,100,100);
}
```

What does the documentation say about color functions?



Order matters!

Set the color for what you are **about** to draw

Alpha

We can actually have four arguments for these functions:

```
function draw() {
  fill(r,g,b,a);
}
```

The fourth argument is the alpha (transparency or opacity)

Alpha uses the same range, (0-255), with 0 for transparent and 255 for opaque.

Exercise: Fill you drawing with Color!

Update your drawing with color.

Use the documentation to help you with the color functions.

Lookup `strokeWeight()`

What does it do?

How does it change your shapes?

Topics for Today

- P5.js
- Shapes and Drawing
- Color
- **Errors and the Console**
- Comments
- Variables

The Console

Below our code is the **console**

This show messages that we **`print()`** and errors in our code.

```
function setup() {
  print("hello world!");
}
```

We use the console to help us program.

If I need to know what my program is doing, I can print messages as it runs.

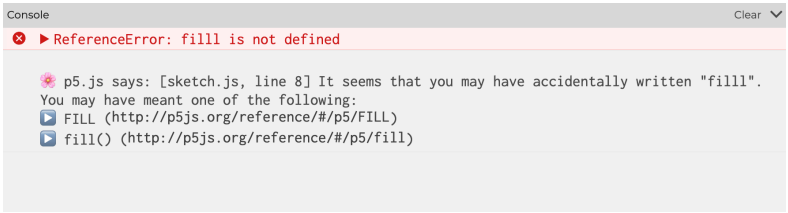
What happens?

Why did I put it in **`setup()`**?

What if we Make a Mistake?

What does the console say if you spell a command wrong?

Or delete one of the commas?



Our editor tries to be helpful, telling us where the issue is and what me may have meant.

Topics for Today

- P5.js
- Shapes and Drawing
- Color
- Errors and the Console
- **Comments**
- Variables

Comments

Comments are lines in our code that explain what is going on but don't affect the actual code itself.

```
function draw() {
  background(220,0,200);

  //set the fill and stroke
  fill(100,0,255);
  stroke(0,255,0);

  //draw the rectangle
  rect(50,50,100,100);
}
```

Comments start with **//**

Any text after is ignored by JavaScript.

Use comments to explain what is happening in your code!

Exercise: Comment your Code

Comment the lines in your code.

You don't need to comment every line, just blocks of code that achieve some goal.

Topics for Today

- P5.js
- Shapes and Drawing
- Color
- Errors and the Console
- Comments
- **Variables**

Variables: Fundamental Concepts

Each variable can hold only one value (kinda...we do have arrays and objects too).

Over time, as the program executes, **the value of the variable can change**.

So, the variable stores a value so that we can use it at any time throughout the program.

This is how programming variables are different from their algebraic ones, which generally only represent one unknown value.

Variables: Names

Programming languages have a few rules for variable names. For JavaScript:

Must start with a letter (upper or lower case) or an underscore.

May only contain letters, numbers and underscores.

Names are case sensitive

Good Examples:

```
count
x
user_input2
hit_points
```

Invalid Examples:

```
42
#yolo
a-b
```


Variables: Declarations

Every variable must be declared before you can use it.

```
let value = 0;
```

You may also see:

```
var otherValue = 0;
```

The difference between **`var`** and **`let`** has to do with scope.

For now, I'll use **`let`** whenever we declare a variable.

Style Note:

I'll use camelCase in the code I write. The uppercase helps read the variable name.

Variables: Initialization

Before you can use a variable, you **must** (should) give it a value!

This is not strictly true, some languages will just use a random value or zero (which may not good for your program).

Generally, JavaScript will just give the variable a value of **`undefined`**, which won't stop your program from running, but be very careful when using other languages.

Printing Variables

Printing variables to the console is just as easy as printing **hello world!**

```
function setup() {
  let value=10;
  print(value);

  print("my value: "+value);

  print(`my value is: ${value}`);
}
```

Remember, **`setup()`** runs only once, and **`draw()`** runs every "frame", so for now, we'll be printing in **`setup()`**.

There are many ways to create and print variables.

Mathematical Operators

Mathematical Operators are (generally) used with numeric types.

Addition (+):	`total = part1 + part2;`
Subtraction (–):	`left_over = total - used;`
Multiplication (*):	`force = mass * acceleration;`
Division (/):	`item_wt = total / num_items;`

Predefined Variables

p5.js has many predefined variables that store specific values related to p5.js functionality.

```
function draw() {
  background(0);
  circle(mouseX,200,50);
}
```

`mouseX` and **`mouseY`** constantly update to store the pixel position of the mouse on the canvas.

What happens if this is in **`draw()`**?

Drawing Program

Let's move background(0) to setup.

What happens now and why?

Use **`mouseX`** and **`mouseY`** as the position of the circle.

```
function setup(){
  background(0);
}

function draw() {
  circle(mouseX,mouseY,50);
}

function mousePressed(){

}
```

p5.js has other special functions like setup and draw. Lets add **`mousePressed()`** to our program.