

Programming Languages

Or: How to Sweet Talk A Computer

Computers: They're Everywhere!



You probably have 3 with you right now



How do we use them to creatively solve problems?

Goals for Programming Languages

- 1. Creatively solving the world's problems**
- 2. Work being done by students like you**
- 3. Real-world uses**

Computers: Bad Conversationalists

Computers bad at English

People bad at binary

1101011010110101?



Draw me some circles, slave!



We need a way to hide computer details and get work done

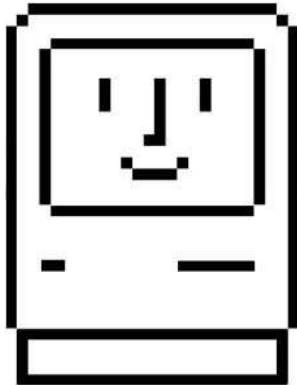
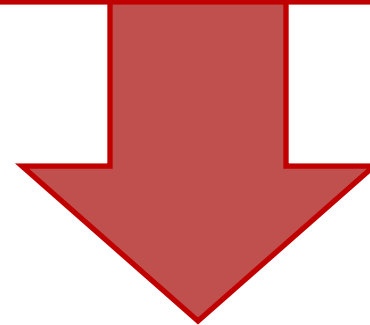
ABSTRACTION: Hiding irrelevant details

Hiding Details: Abstraction



Mouse clicks, typing, gesturing, drawing

ABSTRACTION

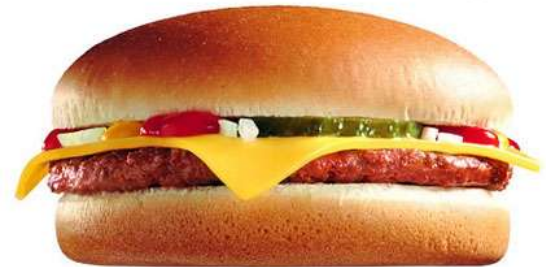


Loading web pages, playing silly videos

Abstractions = Languages

English: Words stand for (“abstract”) ideas

“cheeseburger”



Computers: Clicks, commands stand for (“abstract”) actions



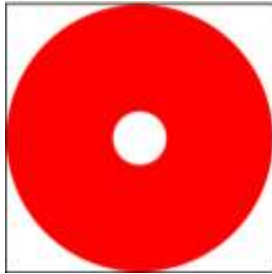
+ Enter



Load `www.facebook.com`
Display logo
Display photos
Display friends' statuses

Programming Languages

Programming languages let you communicate tasks to computers



circle color = red

circle color = white, size = 20



Different ones for different purposes (graphics, music, etc.)

Programming languages **ABSTRACT (hide)** the details

Example: Abstract Art

(A) What we want



(B) What the computer understands

```
setPixel(257, 249, 0, 0, 0)
setPixel(258, 249, 124, 73, 9)
setPixel(259, 249, 124, 73, 9)
setPixel(260, 249, 124, 73, 9)
setPixel(261, 249, 124, 73, 9)
setPixel(262, 249, 124, 73, 9)
setPixel(263, 249, 124, 73, 9)
```

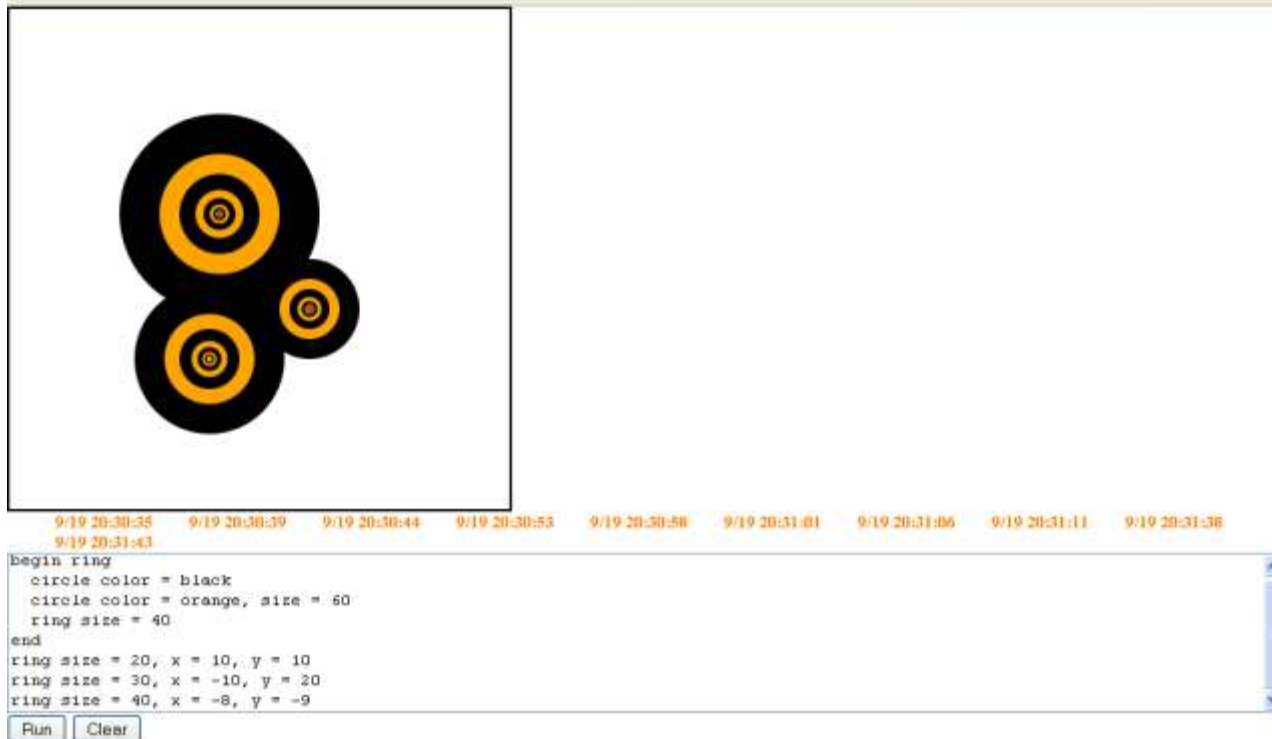
...

How to get from (A) to (B)?

Use a language that **abstracts away** the details of drawing shapes

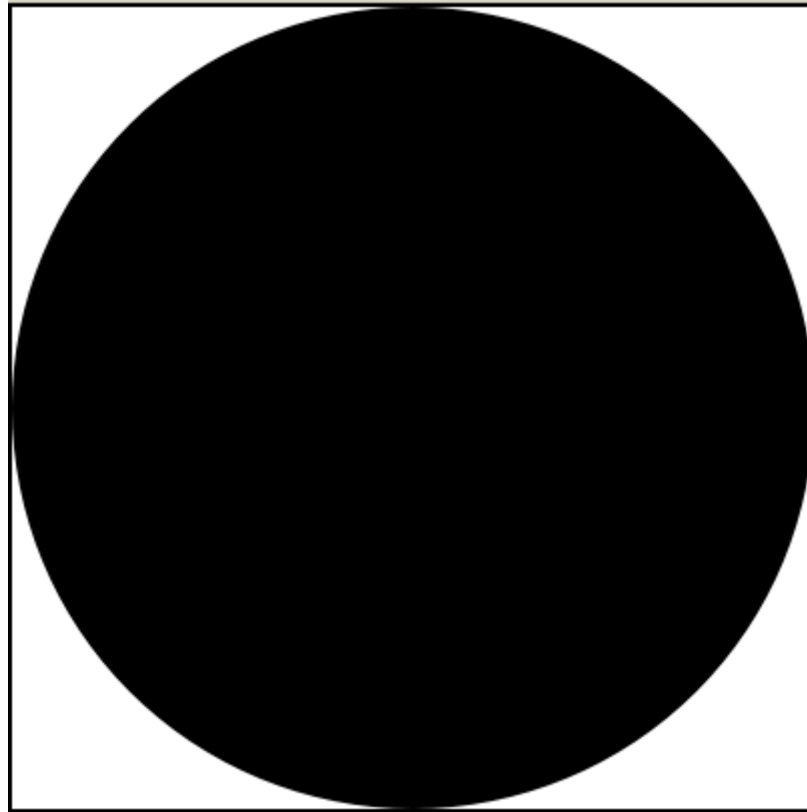
Programming With Shapes

We'll make our art in a programming language for drawing shapes



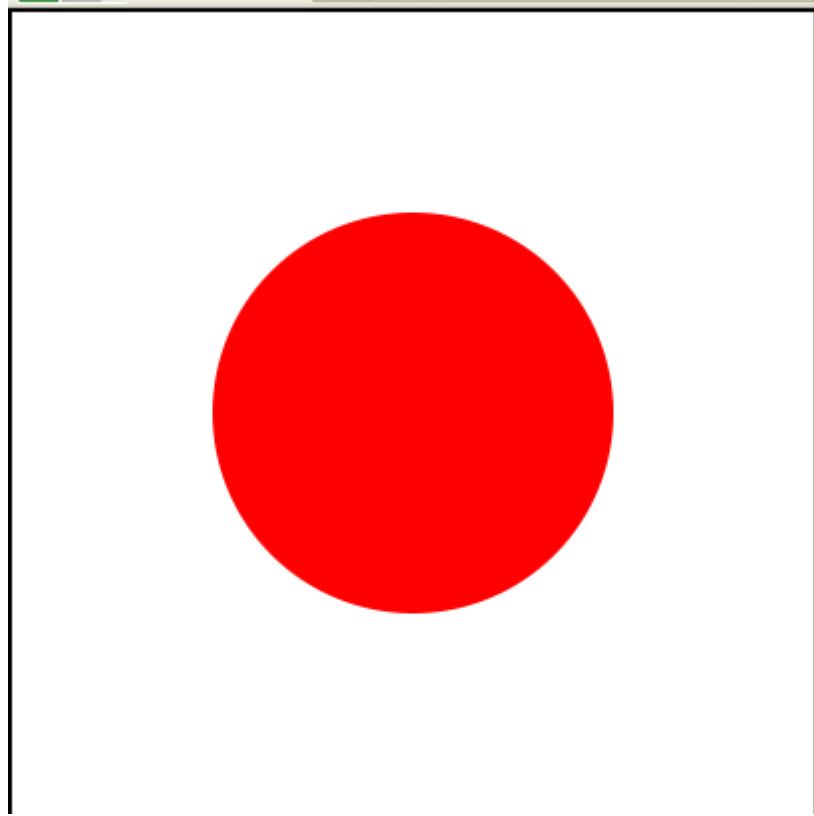
www.cse.ucsd.edu/~prondon/canvas.html

The First Program



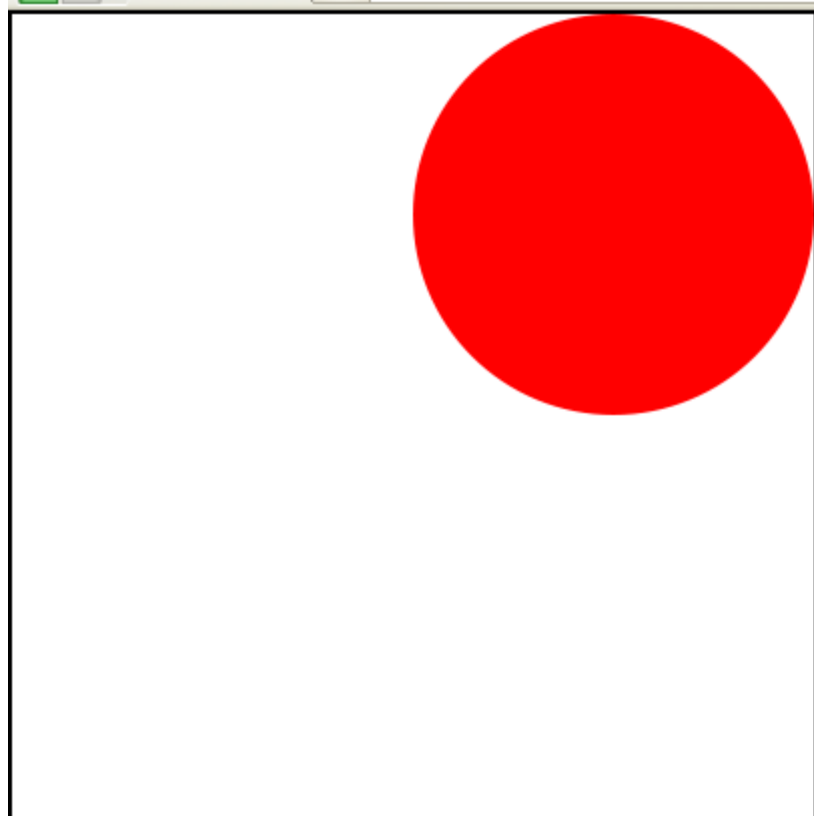
circle

Changing the Size and Color



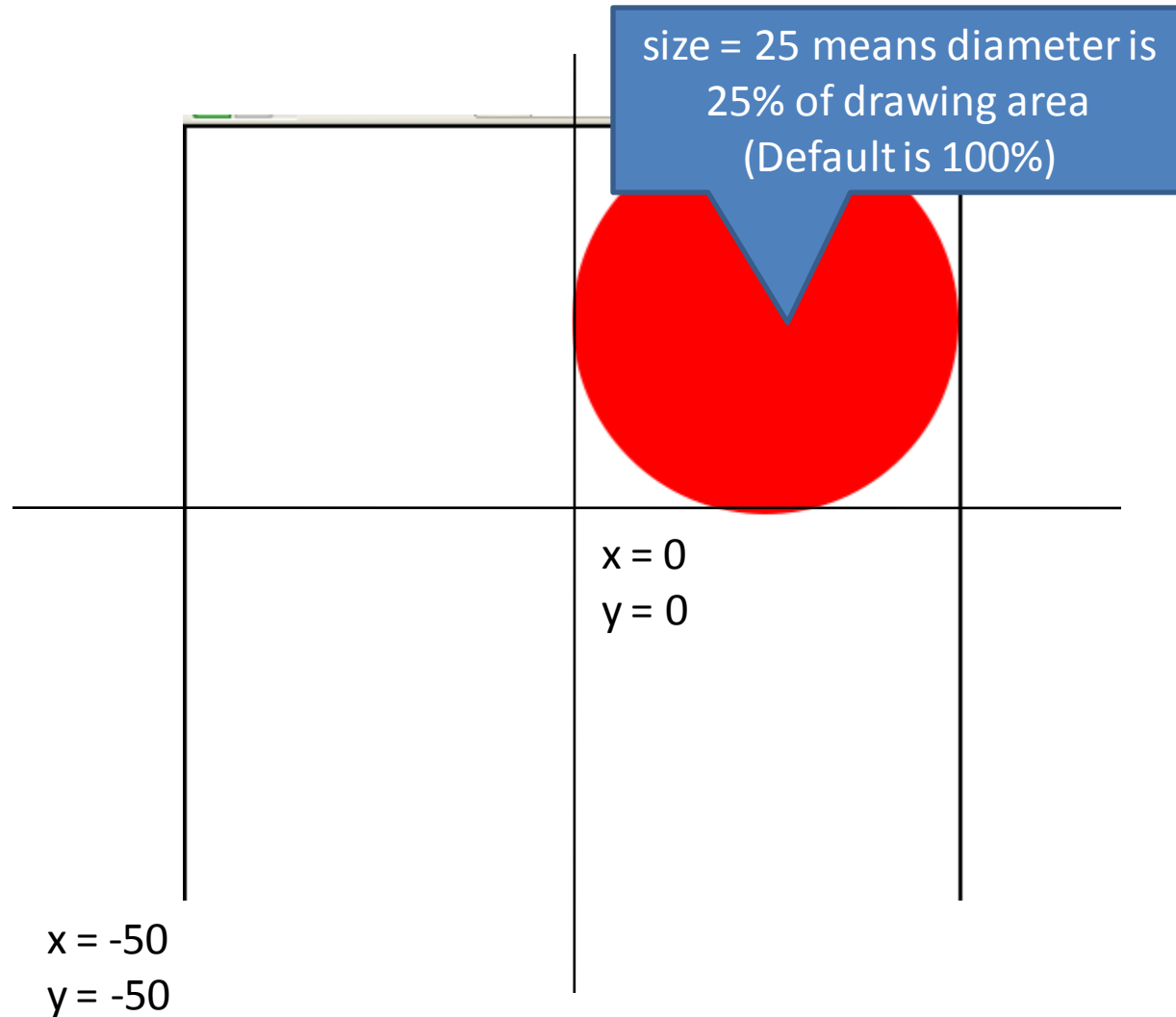
```
circle color = red, size = 50
```

Changing the Position



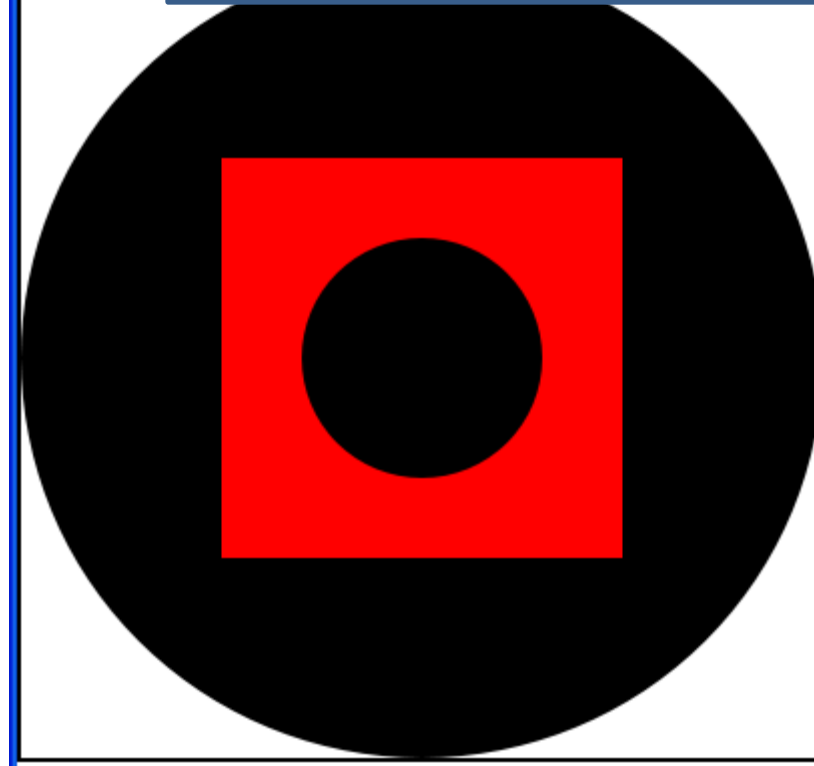
```
circle color = red, size = 50, x = 25, y = 25
```

Aside: Coordinates



Drawing Multiple Shapes

Later shapes drawn on top of earlier ones



```
circle  
square size = 50, color = red  
circle size = 30
```

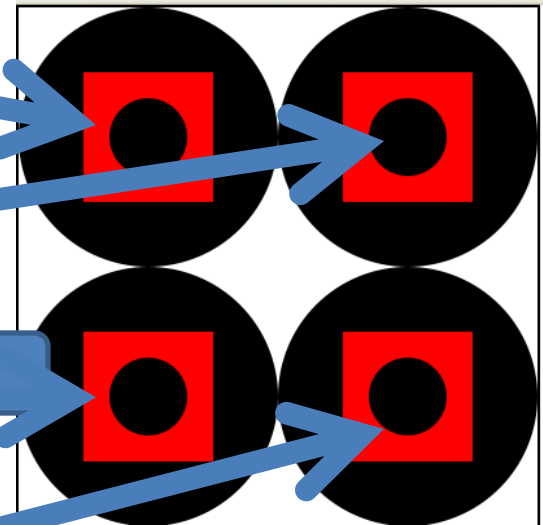
Drawing It Four Times...

Do we really have to write it four times?

```
circle size = 50, x = -25, y = 25  
square size = 25, color = red, x = -25, y = 25  
circle size = 15, x = -25, y = 25  
  
circle size = 50, x = 25, y = 25  
square size = 25, color = red, x = 25, y = 25  
circle size = 15, x = 25, y = 25  
  
circle size = 50, x = -25, y = -25  
square size = 25, color = red, x = -25, y = -25  
circle size = 15, x = -25, y = -25  
  
circle size = 50, x = 25, y = -25  
square size = 25, color = red, x = 25, y = -25  
circle size = 15, x = 25, y = -25
```

Had to resize each shape manually!

Had to position each shape manually!



We want to hide the details of drawing each set of shapes

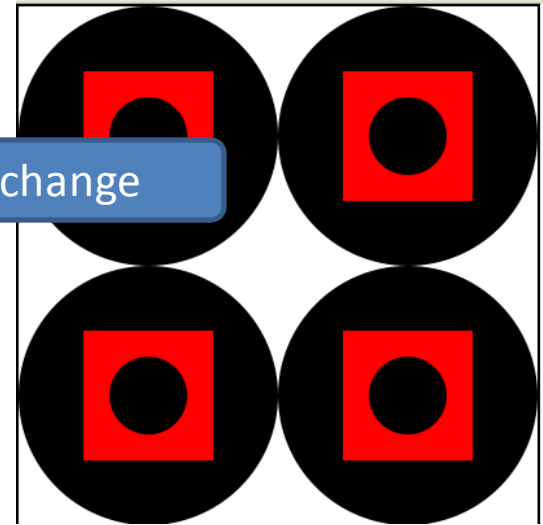
Programming languages let you **define** things and **reuse** them

...But Writing It Once

You can **define** the shape once...

```
begin circlesquare
  circle
  square size = 50, c
  circle size = 30
end
circlesquare size = 50, x = 25, y = 25
circlesquare size = 50, x = -25, y = 25
circlesquare size = 50, x = 25, y = -25
circlesquare size = 50, x = -25, y = -25
```

Only specify the parts that change



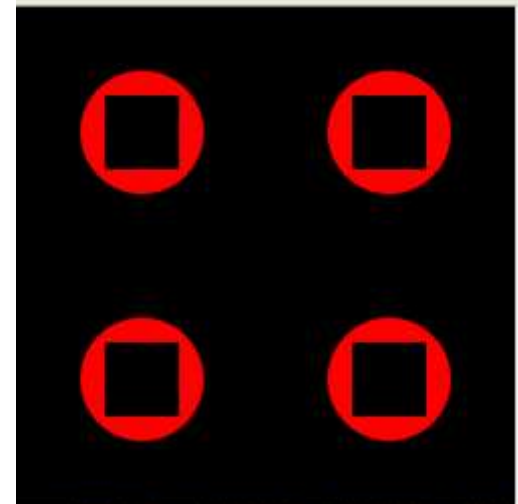
...and then use it four times

Another case of **abstraction!**

Changes Are Easy!

Changing the **one** definition...

```
begin circlesquare
  square
  circle size = 50, color = red
  square size = 30
end
circlesquare size = 50, x = 25, y = 25
circlesquare size = 50, x = -25, y = 25
circlesquare size = 50, x = 25, y = -25
circlesquare size = 50, x = -25, y = -25
```



...changes **all** four uses

What's Under The Hood?

JavaScript

C++

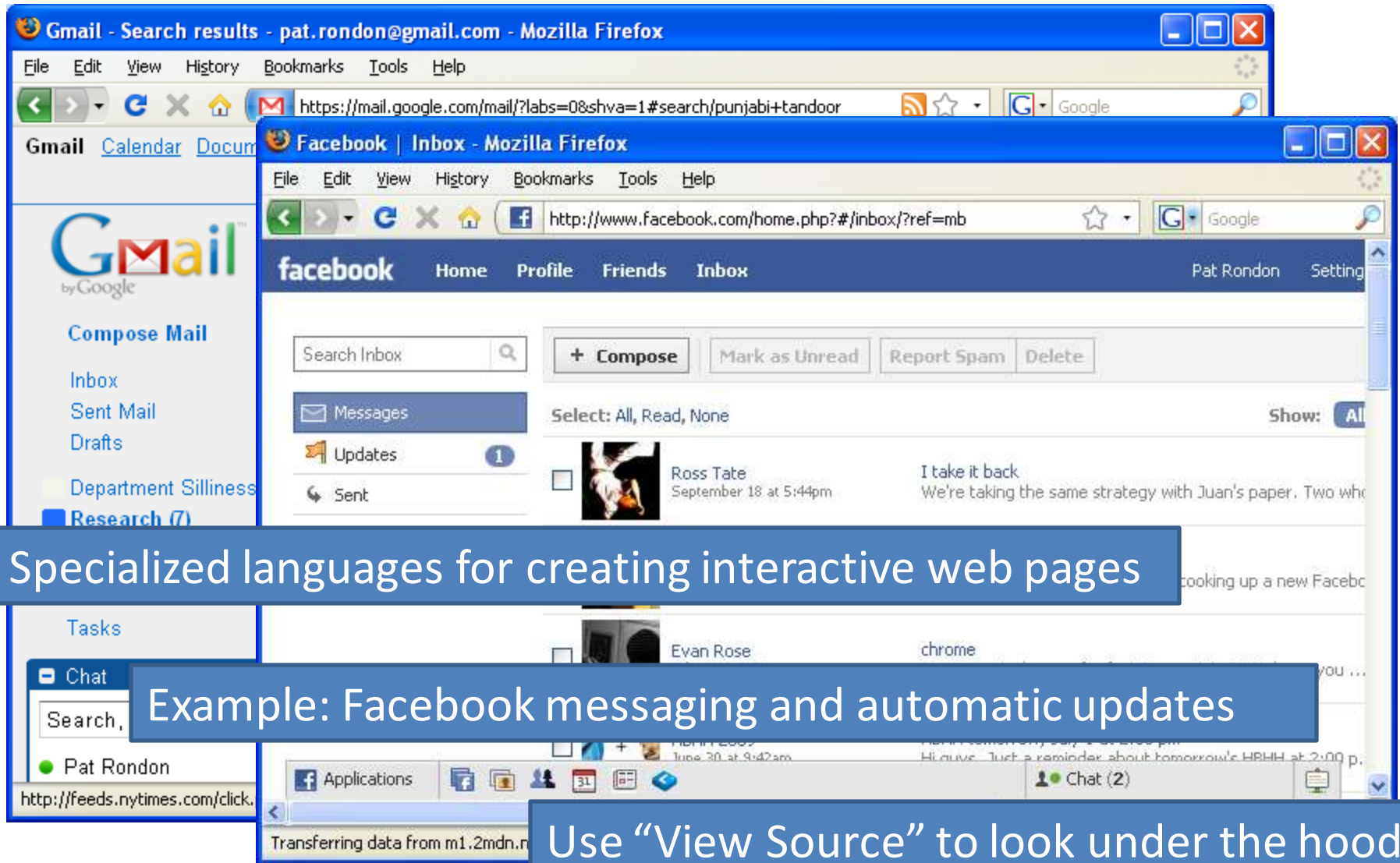
Assembly

11010110
11001001

Each abstraction is built on top of other abstractions

Go to “View Source” and see!

JavaScript and HTML



The screenshot shows a Mozilla Firefox browser window with two tabs. The first tab is titled "Gmail - Search results - pat.rondon@gmail.com" and displays the Gmail interface with a search bar and a list of messages. The second tab is titled "Facebook | Inbox - Mozilla Firefox" and displays the Facebook inbox interface with a search bar, a list of messages, and a chat window. The chat window shows a conversation with "Pat Rondon" and a message from "Evan Rose". The browser's address bar shows the URL "http://www.facebook.com/home.php?#/inbox/?ref=mb".

Specialized languages for creating interactive web pages

Example: Facebook messaging and automatic updates

Use "View Source" to look under the hood

Review: Programming Languages

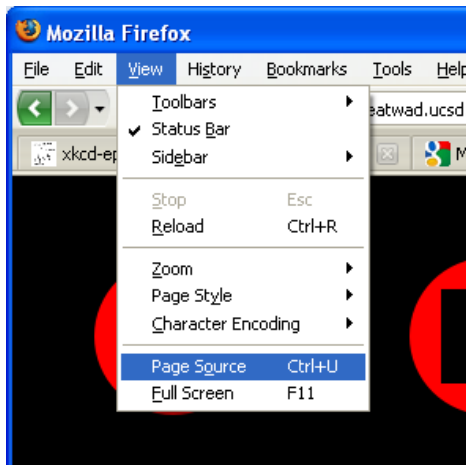
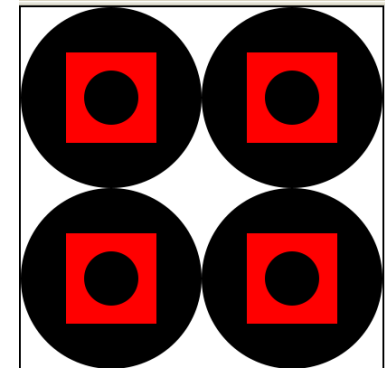
Abstract away computer details



circle color = red
circle color = white, size = 20



Allow you to make your own abstractions



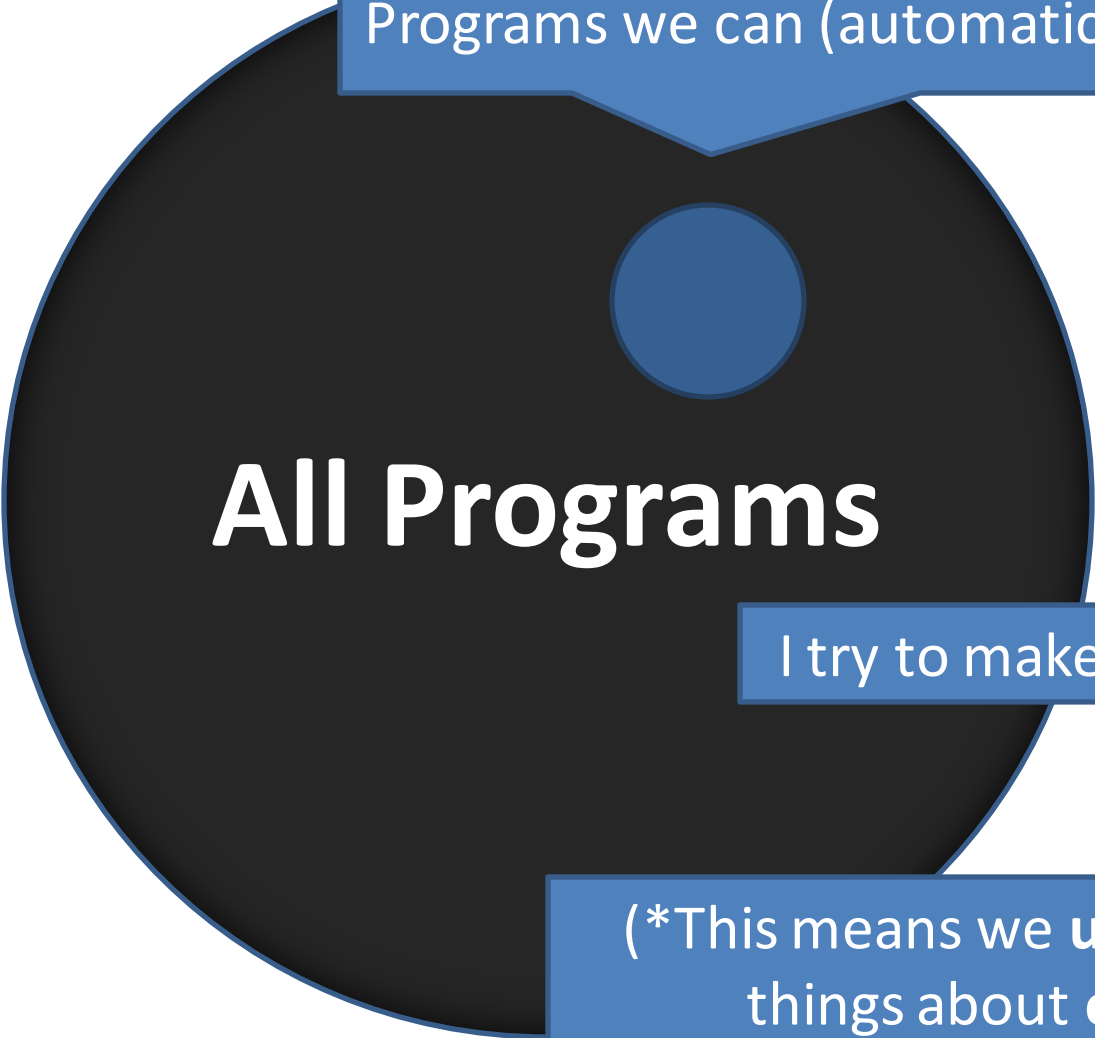
Are just programs you can write

Goals for Programming Languages

- 1. Creatively solving the world's problems**
- 2. Work being done by students like you**
- 3. Real-world uses**

What I Do: Program Verification

Programs we can (automatically) prove don't crash*



All Programs

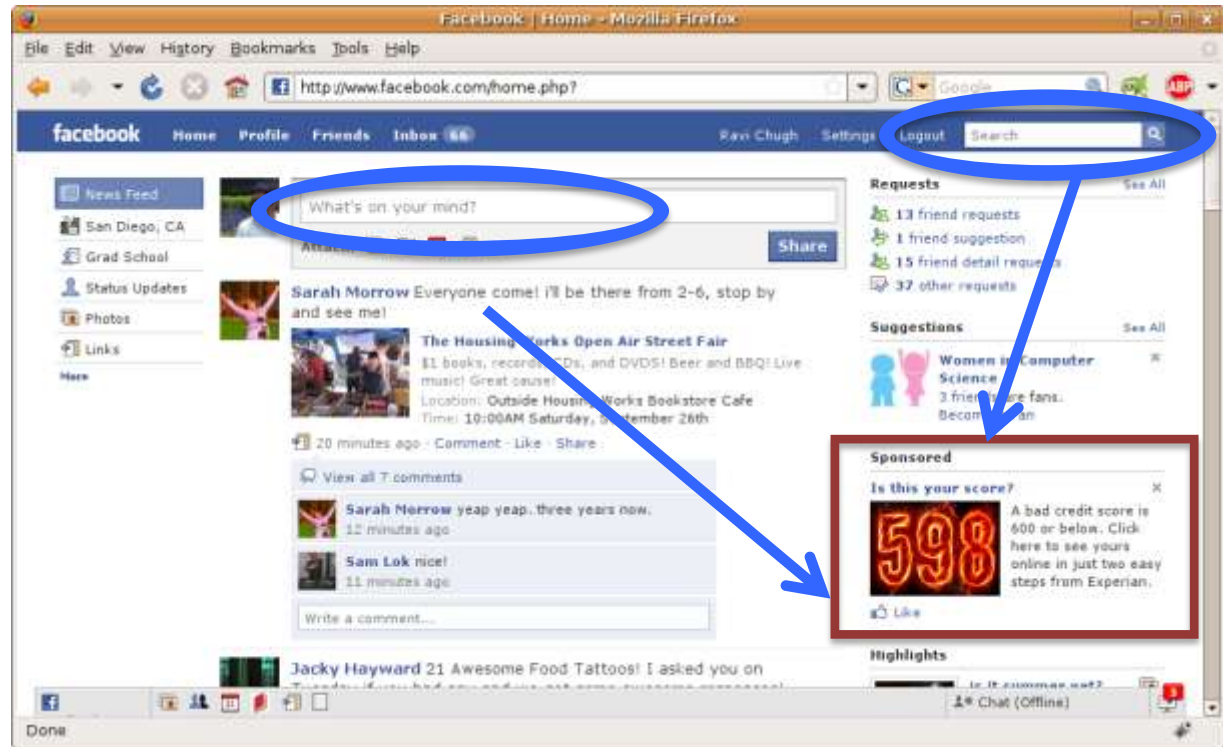
I try to make the blue circle **bigger**

(*This means we **use programs** to prove things about **other programs!**)

Ravi Chugh: Web Security

Modern web sites
run JavaScript code

Ads/gadgets loaded
from other sites



Can code from other sites read what you type?

How to check?

- As the web site is running? (“Dynamically”)
- Before the web site starts running? (“Statically”)
- A combination of both?

See UCSD Programming Systems



<http://cseweb.ucsd.edu/groups/progsys/>

Goals for Programming Languages

- 1. Creatively solving the world's problems**
- 2. Work being done by students like you**
- 3. Real-world uses**

Programming Languages Play Nice

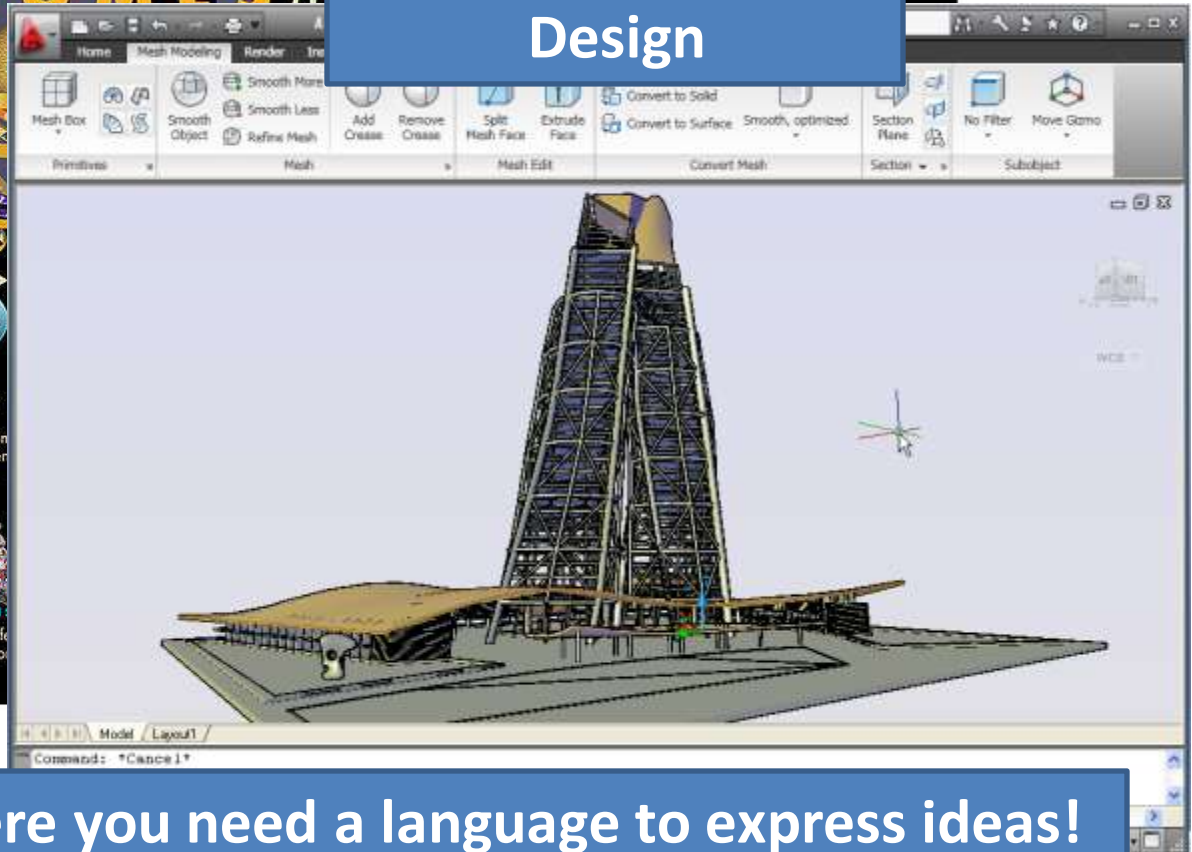
Music and Audio



Biology



Design



Use them anywhere you need a language to express ideas!

Where To Go From Here

CSE 8A: Introductory Computer Science

Programming in Java, creating graphics and other media

CSE 30: Computer Organization, Systems Programming

Going “all the way” under the hood to the machine code

CSE 130: Programming Languages

Studying different languages, their uses and limits

CSE 131: Compiler Construction

How to create your own programming languages