Head TA Lectures

Julie, Angel, Noah, Taylor, Lucy
Table of Contents

1. Web Applications —Angel
3. Shell —Taylor
4. Struggling in CS —Julie
5. CS Past 15 —Lucy
How Do Web Applications Work?

Angel
Let’s look at an example webpage

Welcome to Courses @ Brown
Please visit our brief tutorial for information on using this site.
Supported systems

Have you Considered?

RUSS 0100 (Fall 2019)
Introductory Russian

LACA 1503Q (Fall 2019)
Politics of Indigenerity in Brazil

GRMN 0100 (Fall 2019)
Beginning German

TKSH 0300 (Fall 2019)
Intermediate Turkish

ARAB 0100 (Fall 2019)
First-Year Arabic

CHIN 0500 (Fall 2019)
Advanced Modern Chinese I
Two Main Parts

What the user sees & where the data comes from

Two common ways of describing this:

‘frontend’ vs. ‘backend’

‘client’ vs. ‘server’
Three major elements go into displaying many web pages:

1. **HTML** -- Content
2. **CSS** -- Style
3. **JavaScript** -- User Interaction
Frontend – Part 1 (HTML)

HTML: Hypertext Markup Language

● The most foundational part of a webpage

● Responsible for the base ‘content’ of the page
  ○ Structure, text, links, images

● Broken down into subdivisions
  ○ Like a lecture deck that has slides, with bullets, that have subbullets, each with subsubbullets
  ○ HTML relies on tags to describe what different parts of the page represent
    ■ e.g. ‘This is a header, <h1>’ or ‘This is a paragraph, <p>’
    ■ Tags act as identifiers for styling sheets to understand how to style the page
Frontend – Part 1 (HTML)

- Procedural Markup
  - had to explicitly describe every step
    - formatting codes (e.g., !p, !i5, !f24)

- Declarative Markup
  - use of tags allows us to separates the structure of a document (HTML) from its appearance (CSS)
Frontend – Part 1 (HTML)

You can display web pages with just HTML, and nothing else:

HelloWorld.html
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    <h1>Hello World!</h1>
</body>
</html>
CSS: Cascading Style Sheets

- Responsible for styling, makes each webpage look unique
- Can apply ‘styles’ to specific elements or types of elements on the page
  - takes advantage of tags:
    - ‘turn the text of all headers orange’
    - ‘add a red border around the element that I called #redBox’
- Connect HTML pages to style sheets to format them
- Webpages with just HTML and CSS only are meant to be ‘static,’ so generally don’t have much complicated user interaction
Let’s add some styling to our webpage from before:

```html
HelloWorld.html
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1>Hello World!</h1>
    <div id="redBox">
      <h1>In the box!</h1>
    </div>
  </body>
</html>
```

```css
style.css
h1 {
  color: orange;
}
#redBox{
  border-style: solid;
  border-color: red;
}
#redBox h1 {
  color: blue;
}
```

Attaches the CSS file to the HTML
Frontend – Part 2 (CSS)

Let’s add some styling to our webpage from before:
Frontend – Part 3 (Javascript)

- Code that is run in the user’s browser
- Controls the bulk of user interaction and data manipulation on the user end
  - One way of ‘scripting’ and writing more complicated code on the frontend
  - Many frameworks (such as Angular.js or React.js) are built on top of JavaScript

- Example uses:
  - Respond to user clicking
  - Sorting functionality on websites (e.g. order by price)
  - The Google Sheets TA Hours signups

- JavaScript often asks the server for data, and organizes what it gets back for the user

- Entirely unrelated to the Java that we know
  - JavaScript is partially OOP, but is messy and easy to get wrong
  - More flexibility than Java, but much more finicky
Let's add some Javascript to our webpage from before:

```javascript
window.addEventListener('load', function(){
  document.getElementById("redBox").onclick = function(event){
    alert("Successful Click!");
  };
});
```

```html
HelloWorld.html
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="style.css">
    <script src="demo.js"></script>
  </head>
  <body>
    <h1>Hello World!</h1>
    <div id="redBox">
      <h1>In the box!</h1>
    </div>
  </body>
</html>
```
Frontend – Part 3 (Javascript)
Let’s add some Javascript to our webpage from before:
Frontend – One last thing

- Because all frontend code (HTML, CSS, JavaScript) happens in your browser, you can see the code for any website that you want!
Frontend – One last thing

- Because all frontend code (HTML, CSS, JavaScript) happens in your browser, you can see the code for any website that you want!
Making pretty websites is great, but how does the website do data-related things (e.g. passwords, searching a movie database, etc.)?

- The backend is where the bulk of data processing happens
- Backend is what interacts directly with the database
- **Java** is often used as a primary backend language, as is **Python** (which is taught in CS16)
  - but there are many dozens more that are commonly used
Databases store information in an organized manner
- Makes it easy and quick to query for information
- Can be queried in many different ways to get different subsets of information

‘Relational databases’ are most common, such as SQL (‘Structured Query Language’)
- Consists of many tables that relate to each other
- For example, might have an accounts table and a passwords table, so a user in the accounts table also has a row in the passwords table
Example SQL query:
- Get the first name of every user from the users table who has the last name van Dam

```sql
select first_name from users where last_name = 'van Dam';
```

returns

```
[“Andy”, “Django”]
```
Getting to the Webpage

- Websites have names that are easy to remember, like www.brown.edu

- When you type a website name to your browser, your Internet Service Provider (ISP, like Verizon or Cox) looks at a Domain Name Server (DNS)
  - DNS returns an IP ('Internet Protocol') address of the server to connect to
  - DNS is often called the 'phonebook of the internet,' since you can look up IP addresses from a website name

- Can connect to http://www.brown.edu or http://128.148.252.151/ in your browser and get to the same page
Combination of backend and frontend technologies is often referred to as ‘technology stack’ or ‘full stack’.
Putting It All Together (2/3)

Let’s walk through one way the cab.brown.edu site might work:

1. You type cab.brown.edu into your browser

2. Your Internet Service Provider gets the IP address of the brown.edu web servers and connects you

3. Your browser gets HTML, CSS, and JavaScript from the server to load in your browser

4. You search for ‘Computer Science’ courses by clicking on the drop-down, created by the HTML and styled by CSS
Putting It All Together (3/3)

5. The JavaScript registers the event, and sends a request to the servers (previously located in the basement of the CIT), asking for information.

6. The backend code registers the request, and queries the database for all Computer Science courses.

7. The database returns a list of all CS courses for the backend code to process.

8. The backend code returns the list, after any data processing, to the frontend (your browser).

9. The JavaScript in your browser updates what courses are displayed on the screen.
Interested in Learning More?

Great Learning Resource:
W3 Schools
https://www.w3schools.com/html/default.asp
Popular Apps for Web Creation

WordPress
Weebly
Squarespace
Wix
How Does a Computer Run My Code?

Noah
How do computers understand our language?

- You may have heard that computers can only understand 1s and 0s
  ○ Don’t worry, this is true!
- How does our Java code become those 1s and 0s?
- What does our computer do with those 1s and 0s?
From Java to 1s and 0s

```java
public class cs15 {
    public cs15() {
    }
    public int code(int crab) {
        int rave = 5;
        return crab + rave;
    }
}
```
From Java to 1s and 0s

Source Code (.java)

```
public class cs15 {
    
    public cs15() {
    }
    
    public int code(int crab) {
        int rave = 5;
        return crab + rave;
    }
}
```

Object Code (.class)

```
javac *.java
```

Java Virtual Machine (JVM)

```
java cs15.App
```
So That’s What My .class Files Are?

- In Java, this is called **bytecode**
- In other languages, this is roughly equivalent to **assembly code**

```java
public class cs15 {
    public cs15() {
        Code:
        0: aload_0
        1: invokespecial #1 // Method java/lang/Object."<init>":
    }
    public int code(int); {
        Code:
        0: iconst_5
        1: istore_2
        2: iload_1
        3: iload_2
        4: iadd
        5: ireturn
    }
```
From Java to 1s and 0s

Source Code (.java)
```
public class cs15 {
    public cs15() {
    }

    public int code(int crab) {
        int rave = 5;
        return crab + rave;
    }
}
```

Object Code (.class)

javac *.java

JVM
(Java Virtual Machine)

java cs15.App
So That’s What My .class Files Are?

Tip: You can see your Object Code by running `javap -c <class_file>` if you want!

- In Java, this is called **bytecode**
- In other languages, this is roughly equivalent to **assembly code**
- Each line of code corresponds to one **instruction** that your computer can run
- Before Java (and other high level languages) engineers had to code in assembly!
The Java Virtual Machine (JVM)

- An extra intermediary step between bytecode and the computer processor
- Loads the bytecode then decodes it into the Machine Code
- It does so using the Instruction Set Architecture (ISA)
  - The ISA can be different for every processor!
- Executes at runtime, which is one of the reasons that Java is considered slow
Machine Code

- Machine code is a string of 1s and 0s (AKA bits) that represent instructions
  - Most computers use either 32-bit or 64-bit instructions
- Each line of assembly code directly corresponds to one instruction in machine code
- There are different types of instructions, depending on the processor
  - Trade-offs to having different instructions (ISA Design)
  - More complex instructions → slower processors
- Intel’s Core i9 runs at a clock speed of 5 GHz
  - This means it runs 5 Billion instructions every second!
Running Machine Code

- A computer does the same 3 things over and over again:
  - Fetch
  - Decode
  - Execute
- It does so using **circuits**
- In a circuit, a 1 is actually a wire at a high voltage
  - Traditionally 5V
- A 0 is a wire at ground, or 0V
- We can build complex circuits to perform different functions
Fetching

- Keep track of where we are in the program
  - Instruction Pointer or Program Counter
- Load the next instruction (stored in memory) in the program
Decoding

- An instruction carries pieces of information
- Decoding depends **COMPLETELY** on the ISA
- For example, a RISC-V* processor:

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>25</th>
<th>24</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>15</th>
<th>14</th>
<th>12</th>
<th>11</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>funct7</td>
<td>rs2</td>
<td>rs1</td>
<td>funct3</td>
<td>rd</td>
<td>opcode</td>
<td>R-type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imm[11:0]</td>
<td>rs1</td>
<td>funct3</td>
<td>rd</td>
<td>opcode</td>
<td>I-type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imm[31:12]</td>
<td>rd</td>
<td>opcode</td>
<td>U-type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

* RISC-V is an open-source project encouraging processor innovation and freedom in hardware architecture

How we know the instruction type
- Differentiates the operator within instructions (add, sub, mul)

Types of instructions

Figure 2.3: RISC-V base instruction formats showing immediate variants.

https://content.riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf
Executing

- Each instruction can be executed using a series of circuits
- For example, let’s build a 1-bit adder:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

This looks like XOR!
Executing

- However, we need to add numbers together that are more than 1 bit

\[
\begin{array}{c}
A \\
B
\end{array}
\begin{array}{c}
0110 \\
+1011 \\
01
\end{array}
\begin{array}{c}
1 \\
\text{carry in}
\end{array}
\begin{array}{c}
\text{--- 6} \\
\text{--- 11}
\end{array}
Executing

- However, we need to add numbers together that are more than 1 bit.
However, we need to add numbers together that are more than 1 bit.

Bits A and B go in.

We could have ‘carry in’ from earlier adding.

Bit S is the outcome.

We might have ‘carry out’.

\[ \begin{array}{c}
11 \\
0110 \\
+1011 \\
101
\end{array} \]

\[ \text{carry out} \]
Executing

For a 32-bit adder, we’d need to stick 32 of these in a row!
Executing

- However, we need to add numbers together that are more than 1 bit

These are ‘Logic Gates’

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Ci</th>
<th>S</th>
<th>Co</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

 Bits A and B go in

 We could have ‘carry in’ from earlier adding

 Bit $S$ is the outcome

 We might have ‘carry out’

 $\begin{array}{c}
 0110 \\
 +1011 \\
 \hline
 1001 \\
\end{array}$

$S = 1 \text{ carry out}$
Logic Gates (we must go deeper)

- Take 1-2 inputs and give 1 output
- Basic building blocks of an electronic circuit
- AND, OR, NAND, NOR, XOR, XNOR, NOT
- Example: this is a NAND gate (not both of them)

These are Transistors

Traditionally 5 V, AKA ‘1’

Ground, AKA ‘0’
I Wanna Go Faster!

- There are optimizations made at **every step** of this process!
- Software engineers work to optimize their code (remember Big O?)
- We can compile code better so that we run less instructions
  - Or design the ISA better!
- Improve the processor circuit
- Improve logic gates
- Make transistors smaller → fit more!
  - Shorter paths in the wires
  - Processors have > 10 billion transistors
  - Remember Moore’s Law?
What’s Next In Computer Engineering?

- Gordon Moore (Intel)
  - Predicted in 1965 that the # of transistors on a chip would double every 2 years
- You might have heard that Moore’s Law is ending
  - Unfortunately, probably true
- Companies like Intel and AMD are adding more cores
  - Modern processors are actually made up of multiple processors that work in parallel
- If we can’t pack more transistors on a chip, we might have to look to new technologies
  - MTM (More-Than-Moore)
- For now: we don’t know
  - A really exciting field!

https://www.youtube.com/watch?v=7uvUiq_jTLM
Questions?

- Interested? Classes with pre-reqs that you’ve already taken:
  - ENGN 0500
  - CSCI 0330
$ toilet -f bigmono12 --gay 'SHELL'

SHELL

$ toilet -f smblock --metal "Taylor"

Taylor

cslab00c ~ $
GUI vs CLI

- GUI
  - Graphical User Interface
  - User interacts with the window (left-click, right-click, drag)
  - WIMP
  - Visual flow of elements
  - Highlighted items
GUI vs CLI

- CLI
  - Command Line Interface
  - User types text commands the program will run
    - REPL
  - ex. Xfce-terminal
  - Shell specific scripting languages
    - ex. bash
CLI: Good Idea or Bad Idea?

- Possible negatives
  - Knowledge of a script language
    - cd, ls, touch
  - Reduced interaction for user
    - No second click
    - No dragging
    - No trackpad swipes
  - Reduced design for user
    - Little/no prompts to users for intuitive control flow
    - And...
CLI: Good Idea or Bad Idea?

- Positives
  - Faster
    - GUI takes a lot to load
    - Bypass tedious user input and just type commands to do an equivalent action - practice provides fluidity
  - Get minimum functionality
    - Modularize commands
  - Accessible!
Common Commands

cd         - change directory
ls          - list directory contents
touch      - create a new file or “touches” existing file
rm         - remove a file
mkdir      - create a directory
rmdir      - remove a directory
mv          - move one file to another location
cp         - copy one file
cat        - concatenate and print file
chmod      - change file modes/permissions
grep       - file pattern searcher
clear      - clear the terminal
Piping and Filters

- **Piping** redirects the output of one command to be the input of another command
  - The output is transferred from left to right
  - `command_1 | command_2 | command_3 | .... | command_N`
  - Ex. `ls -l | wc -l` (will show on next slide) will take the output of `ls` and pass it into `wc`

- A **filter** takes input from one program or file, reads it, performs an operation on it, and prints the result to the terminal
  - Ex. `cat, grep, sort, wc`
Flags and Combining Commands

ls
  • ls -l
  • ls -a
  • ls -l | wc -l

rm
  • rm -rf temp/

javac
  • javac *.java

grep
  • find "*" | xargs grep "string to search for"

man
  • man <command name>
Useful Tips and Tricks

- Retyping a recent command?
  - Press the up arrow
  - Try running history

- ~/.environment (for department machines)
  - Loads upon opening a shell
  - Use: path, unmask, aliases, variables
  - Alias
    - alias andyBot='cd ~/course/cs0150/AndyBot'

- Don’t remember the filepath or command?
  - Press Tab once for tab completion
  - Press Tab twice for a list of everything in that directory

- Want to save terminal output to review later? Use >
  - 1s -l > output.txt
Useful Tips and Tricks

● Want to stop something you accidentally ran?
  ○ Ctrl-C to interrupt the process
  ○ Ctrl-Z to pause a process
    ■ Can then send to foreground (fg) or background (bg)
      using the process ID

● Really quick code fix and don’t want to wait for eclipse to load?
  ○ vim or nano (though nano is clearly better)

● Are things broken?
  ○ ncdu (if you went over quota)
    ■ cache is safe to delete
  ○ fix-eclipse
  ○ fixscreens
~FUN~ Commands

- Bash
  - `cal`
  - `date`
  - `su <login>`
  - `banner <text>`
  - `telnet towel.blinkenlights.nl`
    - `nc towel.blinkenlights.nl 23` (if MacOS Sierra or above)

- CS_FUN*
  - `food`
  - `cowsay <text>`
  - `sl`
  - `tetris`
  - `lsd`

*commands on the department machines
Interested?

Introduction to Computer Systems

This course covers the organization of computer systems (in terms of storage units, caches, processors, and I/O controllers) and teaches you assembly and C language programming.
Struggling in CS
& Why You Belong Here
Julie
CS is hard.
You are not expected to do things right 100% of the time.

(not even close)
learn how to do everything right

gain experience in fixing what’s wrong
Process: com.example.myapp, PID: 17295

java.lang.RuntimeException: Unable to start activity ComponentInfo{com.example.myapp/com.example.myapp.MainActivity}: java.lang.NullPointerException
  at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2195)
  at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2245)
  at android.app.ActivityThread.access$800(ActivityThread.java:135)
  at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1196)
  at android.os.Handler.dispatchMessage(Handler.java:102)
  at android.os.Looper.loop(Looper.java:136)
  at android.app.ActivityThread.main(ActivityThread.java:5017) {1 internal calls}
  at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:779)
  at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:595)

  at com.example.myapp.NameUtil.toFullName(NameUtil.java:6)
  at com.example.myapp.MainActivity.onCreate(MainActivity.java:19)
  at android.app.Activity.performCreate(Activity.java:5231)
  at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1087)
  at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2159)
  at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2245)
  at android.app.ActivityThread.access$800(ActivityThread.java:135)
  at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1196)
  at android.os.Handler.dispatchMessage(Handler.java:102)
  at android.os.Looper.loop(Looper.java:136)
  at android.app.ActivityThread.main(ActivityThread.java:5017)
  at java.lang.reflect.Method.invoke(Native Method) {2 more...}
You’ve already learned so much.

If you don’t believe me, try this:

cs0150_install AndyBot
*disclaimer: NOT my Tetris
Square[] _activeSquares
Square[] _activeSquares

Checking Move Validity:
- for every active square:
  - see if square underneath it is black

... and then nothing moved
private boolean legal(int row, int col) {
    if (row < 0 || row > (Constants.NUM_ROWS - 1) || col < 0
        || col > (Constants.NUM_COLUMNS - 1)) {
        return false;
    } else if (!(_matrix[row][col].getColor().equals(
        Constants.UNOCCUPIED_COLOR) || _matrix[row][col].getColor()
            .equals(Constants.SHADOW_COLOR))
            && !this.isActive(_matrix[row][col]))) {
        return false;
    } else {
        return true;
    }
}
In practice, programmers rarely “get it right the first time.”
Doctors: Googling stuff online does not make you a doctor.

Programmers:
When you delete a block of code that you thought was useless

me: let's rewrite the CSS
my website:

junior dev: "i found the bug"
senior dev: "i found a bug"
Problems become familiar problems.
Struggling is *not* a sign of failure.
"I did something wrong..."
"Now it doesn't work..."

"Where have I seen this before?"
"How can I figure out what happened?"
"Where can I find more information?"
Struggling + Frustration => Learning
**Bulette**

A bulette is a massive predator that terrorizes any lands it inhabits. Also called a "land shark," it lives only to feed. Irascible and rapacious, bulettes fear no other creature, and they attack with no regard for superior numbers or strength.

**Underground Hunters.** Bulettes use their powerful claws to tunnel through the earth when they hunt. Heedless of obstruction, they uproot trees, cause landslides in loose slopes, and leave sinkholes behind them. When vibrations in the soil and rock alert a bulette to movement, it shoots to the surface, its jaws spread wide as it attacks.

**Wandering Monster.** A bulette ranges across temperate lands, feeding on any animals and humanoids it comes across. These creatures dislike dwarf and elf flesh, although they often kill them before realizing what they are. A bulette loves halfling meat the most, and it is never happier than when chasing plump halflings across an open field.

A bulette has no lair, but roams a hunting territory up to thirty miles wide. Its sole criterion for territory is availability of food, and when it has eaten everything in an area, a bulette moves on. These creatures often home in on humanoid settlements, terrorizing them until their panicked residents have fled, or until the bulette is slain.

All creatures shun bulettes, which treat anything approaching them as prey. They detect an untold number of these enemies and select a target at random. To them, any humanoid is food, and they will attack without hesitation or provocation.

**Armor Class** 17 (natural armor)

**Hit Points** 94 (9d10 + 45)

**Speed** 40 ft., burrow 40 ft.

<table>
<thead>
<tr>
<th>STR</th>
<th>DEX</th>
<th>CON</th>
<th>INT</th>
<th>WIS</th>
<th>CHA</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>+4</td>
<td>21</td>
<td>+5</td>
<td>2</td>
<td>-4</td>
</tr>
<tr>
<td>11</td>
<td>+0</td>
<td></td>
<td></td>
<td>10</td>
<td>+0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Skills** Perception +6

**Senses** darkvision 60 ft., tremorsense 60 ft., passive Perception 16

**Languages** —

**Challenge** 5 (1,800 XP)

**Actions**

**Bite.** Melee Weapon Attack: +7 to hit, reach 5 ft., one target.

- Damage: 14 (2d6 + 4) piercing damage.
‘So this is what I’m using…’

‘Yeah, but haven’t you heard of these?’

. CSV  JSON  SQL  React
CS rests upon foundational principles.
You are not behind. At all.
TETRIS IS THE MOST SOLD GAME EVER WITH 495 MILLION COPIES SOLD
The Goal of CS

Know how to do everything

Know how to figure out how to do what you want to do.
Part of that process includes not knowing what to do.

AND THAT’S COMPLETELY FINE.
Go get ‘em, tiger.
CS Past 15

Research | Industry | Involvement at Brown | Culture
Research
Why Research?

- Often different from classwork, opens up the black box
- Get to know professors and undergraduates, M.Sc. and Ph.D. students better
- Lots of interdisciplinary opportunities
- Path to Graduate School
- Be independent and creative
- Learn how to read the latest research literature
- FUN!
What can you do with research?

- Get into Grad School! Get awesome jobs!
- On the path to becoming an Academic
- Work on cutting-edge problems in industry
- Ex: Microsoft Research (est. in 1991) is one of the largest, most respected software research organizations
  - They work in a variety of areas like systems, machine learning, security, data mining and big data, computer graphics and computer vision, speech and natural language understanding, UI and UX, algorithms, quantum computing...
  - Typically need a Ph.D. to work there, though some researchers do have developers working with them
  - Research components of Google, Facebook, Snap, IBM, Adobe...
How to Get Involved with Research (1/2)

● Ask specific professors about projects
  ○ The best way to find interesting options is by talking to professors
  ○ Be prepared to talk about your interests and background

● Take a 1000-level course
  ○ Taking upper-level CS courses is a great way to explore your interests and discover what you like

● Take a graduate course
  ○ Often have a research component to it
Many open opportunities for undergrads!
- Attend the Town Hall meetings every semester
- Browse cs.brown.edu for info about specific professors and projects
  - Undergrad CS research overview: cs.brown.edu/degrees/undergrad/research
  - Undergrad CS research opportunities
    https://cs.brown.edu/degrees/undergrad/research/research-opportunities/
  - UTRAs https://www.brown.edu/academics/college/fellowships/utra/
- Meta-URAs (Undergrad Research Assistants) help coordinate research
  - Any undergrad research questions? Contact mura@cs.brown.edu

Typically start after a few years of CS, but some faculty (e.g., Andy, Stephanie Tellex) take students after 15/16
Some Brown Research Areas

- Brown Laboratory for Linguistic Information Processing (BLLIP)
- Data Management and Data Science
- Computational Vision
- Human-Computer Interaction
- Human to Robots Laboratory
- Programming Languages
- Systems (Brown Systems Research Group)
- Theory
- Visualization Research Lab
Industry
What “Industry” Looks Like

Government
- NASA
- NSF
- HealthCare.gov

Non-Profits
- Khan Academy
- techsoup
- change.org
- Kiva

For-Profits
- Adobe
- Facebook
- Apple
- Pixar
- Microsoft
- Google
- Dropbox
- Flatiron
- Activision
- Teespring
- Figma
- MongoDB

(Founded by Brown Alums!)
What “Industry” Looks Like

- **Education Tech (‘EdTech’)**
  - Coursera, Khan Academy, Knewton, Canvas
- **Health Tech**
  - athenaHealth, HealthCare.gov, Clover, Flatiron
- **Financial Tech (‘FinTech’)**
  - NerdWallet, Jane Street, Square, Bloomberg
- **Consumer Tech**
  - Google, Facebook, Apple, DropBox
- **Animation**
  - Pixar, Dreamworks
- **Video Games**
  - Activision, Blizzard, Bungie, EA, Valve
- **Bio Tech**
- **Ad Tech**
- **Robotics**
- **Virtual and Augmented Reality**
- **Security**
- **Databases**
- **Gaming**
- **Lots and Lots More...**
Types of Careers in Industry (1/3)

- **Software Engineer/Developer**
  - Focus on creating and coding the software
  - Variety of specialties: Test/Quality Assurance (QA), etc.
  - **Not a code monkey, not coding 10-12 hours a day**
    - Often in meetings collaborating on design, setting requirements, and talking to prospective customers
    - Depends on company/job, so research/ask about it during process
  - Can work on different parts of applications:
    - Specialists: Frontend, Backend, Databases
    - Generalist: ‘Full-Stack’
Types of Careers in Industry (2/3)

- Project/Program/Product Manager
  - Some of our best HTAs have gone into Program Management
  - Focus defining what the product should be and what features it should have
  - Includes some level of project management/coordination
  - Work with both prospective users and software developers
  - Technical position
    - Some PMs code and make prototypes
  - Can’t just tell everyone what to do. Have to convince the engineers that your plans are the best for the product

- Being a PM doesn’t mean you can’t be a SWE after (and vice versa)
Types of Careers in Industry (3/3)

- UX (User Experience) Designer
- UI (User Interface) Designer
  - UX vs. UI
- Data Scientist
- Systems Programmer
- IT Architect
- And many more!
The Road to a Job/Internship*

- Research companies
- Submit resume (online, at Tech Fair, at Career Fair, by email, etc.)
- Technical interview (phone/skype)
- Onsite Interview
- Offer

*very approximate - many companies will have slightly different steps!
Aside: Technical Interviews (1/3)

- Many software development jobs & internships require some form of technical interviews
- On an online coding pad or on whiteboard, interviewer gives some problem that you have to solve
- In 45 minutes to an hour, you are expected to reason through different ideas and write code or pseudocode for a solution
  - Expect to talk aloud and show your work
  - Your solution doesn’t necessarily have to work or be totally right!
  - *Most importantly, they just want to see how you think*
● If coding, can normally use whatever language you are most comfortable in
● Problems are often algorithmic:
  ● ‘ How could you reverse a LinkedList?’
  ● ‘ How could you build a Queue using two Stacks?’
  ● ‘ Imagine <long scenario about some hypothetical game>. How would you account for <specific case or rule>?’
● Many of the foundational data structures and algorithms (and methods of problem-solving) needed for technical interviews are covered in lots of depth in CS16 :)

Aside: Technical Interviews (2/3)
Aside: Technical Interviews (3/3)

- Common practice book is ‘Cracking the Coding Interview’
  - Coding in Java
  - Can get in PDF online or in print
  - Sibling version exists for product management roles
What an Internship Might Look Like (1/2)

- Working at a small company/start up (< 50 people)
  - Typically only designing 1-2 products
  - Small user base → take more risks in project features
    - Have a larger say in the direction of the project
  - Fast paced → push out features as fast as you can build them (every couple of days)
What an Internship Might Look Like (2/2)

- Working at a mid level - large company (> 200 people)
  - Building many products, can get exposed to multiple different technologies in one company
  - Larger user base → take less risks, work is heavily reviewed
    - Might not have as much freedom, but affect many more users
  - Slower paced → features pushed out every couple of weeks/once per month

- Personal Experience:
  - Worked as a software engineer for Grupo Siayec, Software Company in Mexico City
    - First internship
    - Company size: 100
  - Team was 3 engineers and me
  - Developed 4 projects from scratch for clients that are now used all over Mexico
Department & Culture
Future Classes

- Some options for next semester
  - CS16: Algorithms and Data Structures
  - CS22: Discrete Math and Probability
  - CS1800: Cybersecurity and International Relations
- After that, courses in our 10 tracks
Involvement at Brown (1/3)

- **WiCS - Women in Computer Science**
  - Mentorship program, meetings and events
  - Supports the Artemis Project, a free summer camp for rising ninth-grade girls from the Providence area who show interest in science and technology.

- **Mosaic+**
  - Advocate for diversity within Brown's CS community
  - Big-little system, workshops, group study
Involvement at Brown (2/3)

- **Hack@Brown**
  - Hackathon and year-round workshops, started and run by a team of undergrads
    - Run workshops and events year-round
The UTA Program

- 60% of Brown CS concentrators TA at least one semester
- Applicants open in March for the next Fall semester and October for the next Spring semester
- TAing is a great way to help students, reinforce class material for yourself, make a little money, and to meet other people in the department!
- Most past TAs and Head TAs have said TAing was one of the most important parts of their Brown education
Myth: You have to eat/sleep/breathe Computer Science

My experience:

- Take plenty of courses in other departments
  - Lots of people double-concentrate
- Don’t have to pull all nighters to be a ‘real’ CS student
  - Note: There is no such thing as a ‘real’ CS student
- Don’t have to spend all your time in the CIT!
Myth: An [ ScB / AB ] is the better option.

My experience:
- Really doesn’t matter
- Take classes that you find interesting!
- Could do the AB and end up taking extra classes above the minimum (Andy calls this ‘ the AB++’ )
Myth: As a computer science student, you should do a software engineering internship after freshman, sophomore, and junior years to be on the ‘right track’

My experience:

- Can take a few summers to explore your interests
- Totally possible to not intern freshman or sophomore year and build your resume in other ways: Research/TAing/other involvement/summer classes/non-CS jobs
Myth: If you’re in a computer science class, you probably want to work in the tech industry

My experience:

● It’s just a field of study - people come to CS from many backgrounds and graduate CS to do a variety of things

● Some options: engineering, product management, nonprofits, research/academia, completely unrelated work
Myth: Computer science is the best and/or hardest concentration

My experience:

- CS is undoubtedly hard but so is everything else! College is difficult and everyone works hard in their field.
- Take advantage of the Open Curriculum. Explore other fields to get a well-rounded experience. All concentrations are valuable and important!
In Conclusion...

• Good luck with your final project and the rest of your semester
  ○ You’ve got this!
  ○ Reflect on how much you’ve learned - go back and look at AndyBot, we promise it will be an enlightening experience

• We <3 Brown CS
  ○ The people in this department are awesome
  ○ This field is awesome
  ○ Computers are awesome
  ○ You are awesome!!!