CS 105
What Is Computer?

Well, that's easy.

A computer is a device that uses algorithms to process data into information.
What Is An Algorithm?

Simply put, it's a recipe.

It provides abstract instructions on how to solve a problem.

You use algorithms every day
What Are Data?

Little pieces of "stuff"

Numbers for example

Or Strings
What Are Data?

There are many forms of data

Images

Sound Files

Videos

etc.
How are data stored?

Numbers
Not terribly hard

Everything Else
A bit more complex
A quick trip to the 80s
A quick trip to the 80s
OK, maybe the 60s

Capacity: 80 bytes
What are bytes?

One byte consists of 8 bits.

And what is a bit, then?

The smallest unit in a computer
What does a Bit look like?

Like This

Or This
What does a Bit look like?

Like This
Not punched

Or This
Punched
But why just 1 and 0?

Because deep down, each computer is just an electronic circuit.

And you can think of 0 as OFF and 1 as ON.
<table>
<thead>
<tr>
<th>But why just 1 and 0?</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>True</td>
</tr>
<tr>
<td>High Voltage</td>
</tr>
</tbody>
</table>
It's all about those electrons
What about all the other numbers?

Computers only know 0 and 1

We can use these two to represent all other numbers

... and just about everything else
Binary Numbers

Every decimal number can be converted into a binary and vice versa.
How?

Let's look at a decimal

21 \( = (1 \times 1) + (2 \times 10) \)
How?
Let's look at a decimal

21

Tens  Ones
How?

Let's look at a decimal

5921

Thousands

Hundreds

Tens

Ones
How?

Let's look at a decimal

Each digit can assume 10 different "states" (0,1,2,...,9)

Each digit is worth ten times as much as its right neighbor
How?

In a binary number

each digit can assume 2 different states (0 and 1)

each digit is worth twice as much as its right neighbor
How?

In a binary number

10

Twos  Ones
How?

In a binary number

10

Twos

Ones

(0*1)
In a binary number

10

(0\times 1) + (1\times 2)
How?

In a binary number

10

\((0 \times 1) + (1 \times 2) = 2\)

Twos

Ones
Can you guess?

What would this binary number look like in decimal?

101011

(1*1)+(1*2)+(0*4)+(1*8)+(0*16)+(1*32)
Now let's try the opposite

Convert this decimal number into its binary equivalent

\[99-64-32-0-0-0-2-1=0\]

<table>
<thead>
<tr>
<th></th>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
But this isn't 1964

Capacity: 80 bytes
A typical DVD: 4.7 Gigabytes
Cards needed for your movie:
470,000,000 / 8 = 58,750,000
That's a lot of cards!

So a stack of 58,750,000 cards should be approximately 6.18 miles high.
Back to the Future...

How does a tape work?
Back to the Future...

How does a tape work?
Back to the Future...
How does a tape work?
Back to the Future...

What about music cassettes?
Back to the Future...

Well, they were analog
Back to the Future...
How about a hard disk?
Back to the Future...

How about a hard disk?

It's still all zeroes and ones!
How do we get the data?

We'll have to tell the drive where it is like giving somebody your address.
### Addresses in Memory

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>00020480</td>
<td>c0 66 fb 81 ff ff ff ff ff 40 67 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>00020490</td>
<td>c0 67 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>000204a0</td>
<td>c0 68 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>000204b0</td>
<td>c0 69 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>000204c0</td>
<td>c0 6a fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>000204d0</td>
<td>c0 6b fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>000204e0</td>
<td>c0 6c fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>000204f0</td>
<td>c0 6d fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>00020500</td>
<td>c0 6e fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>00020510</td>
<td>c0 6f fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>00020520</td>
<td>c0 70 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>00020530</td>
<td>c0 71 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>00020540</td>
<td>c0 72 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>00020550</td>
<td>c0 73 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>00020560</td>
<td>c0 74 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>00020570</td>
<td>c0 75 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>00020580</td>
<td>c0 76 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>00020590</td>
<td>c0 77 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>000205a0</td>
<td>c0 78 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>000205b0</td>
<td>c0 79 fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>000205c0</td>
<td>c0 7a fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>000205d0</td>
<td>c0 7b fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>000205e0</td>
<td>c0 7c fb 81 ff ff ff ff ff</td>
</tr>
<tr>
<td>000205f0</td>
<td>c0 7d fb 81 ff ff ff ff ff</td>
</tr>
</tbody>
</table>
Addresses in Memory

When we know the address
We can readily access data

We can also write to an address
Well, not just any address...
... at least not safely
Why Should You Care?

As you begin programming, you will want to store some data.

And you'll want to get it back later.
How do you do that?

Thankfully JavaScript does the hardest part for you.

You just tell it that you want to create a "variable" (like a box for stuff) and it will find space in memory for you.

It will even keep track of the address and retrieve your stuff for you.
The Takeaway

Data are little pieces of "stuff" that can represent numbers, texts, images and a lot more.

Using a little "magic" we can represent all data in binary numbers.

Information is something that is useful to the user.
An algorithm is a strategy or a "plan" to perform a task or solve a problem.

We can store data on many media and retrieve them by pointing to their address.

ALF was awesome.
Whew!

That was a lot of stuff!

But we're done for the day!