arthur:
a great programming language
Ingredients for an arthur program

- 1 MP4 of ballers dunking
- 1 MP3 of a glass breaking sound
- 1 color: red
- ~35 lines of real live code
Let's show a sample!

BALL EM
What?

High level media manipulation language run primarily in Java that compiles to finished static canvas-y JavaScript sites.
Data Types: Primitives

```plaintext
num x = 5;
string x = “This language is great!”;
color x = <<255, 0, 255>>;
color x = CHARTREUSE;
```

All initialized with literals

Realtime manipulation
Data Types: Non-Primitives

Video \( x = \text{video}( \text{"arthurShow.mp4"} ) \);

Sound \( x = \text{sound}( \text{"inDaClub.mp3"} ) \);

Image \( x = \text{image}( \text{"starwars.jpg"} ) \);

Initialized with files

Java manipulation
What would happen if you...

- Added a color to a sound?
- Multiplied two strings?
- Turned a sound into a picture?

We figured it out!!!!
The main ideas

There are two sides to Arthur

- First: A creative process with unusual results
  - Morphology between medias
  - A space to experiment
  - Output with a wow factor
The main ideas, cont’d.

- Second: **A suite of editing possibilities in a single package**
  - Eliminates need for multiple software tools for different types of media
  - **Condenses heavy-duty media manipulation routines into very simple & very small code styles**
Casting & interoperability

All types can be cast (\(\rightarrow\)) to one another.

The operands of (\(+, -, /, *\)) can be of any two types!
Casting

- Video→Sound extracts sound from a video and saves it as an MP3
- Video→Image samples & combines frames from the video
- Image→color gets you the average color of all the pixels in the image
- string→Sound performs “text to speech”
- number→Sound ??? try it and find out
Image -> string

Video -> image
Interoperability

The result of an operation is the same type as its left operand.

- Video * number speeds up or slows down the video by a factor
- Sound + number raises the sound’s pitch by an amount
- Image / Image overlays two images
- Video / number tiles the video
* 3 / 4
sound * string

JUST BEING HONEST
Then what?

If you want to use arthur as a tool for editing pictures, sounds, and videos

-> Just scoop up the media files from the outputs folder

If you want to watch something crazy and cool

-> Open and deploy the target program, an HTML5 Canvas application
Putting it all together

Arthur programs have three main parts:

1. *Initialize* media variables from file names and literals and *manipulate* them as you please
2. *Choreograph the* presentation *of media variables within the canvas application*
3. *Set up event handlers for real-time* user interaction *with the canvas application*

*2 and 3 are optional, of course*
Program structure

```java
void init() {...} //initialize and manipulate media (backend - Java) & add it to the canvas
void loop() {...} //alter canvas in real-time (backend - JavaScript)
  void key() {...} //make canvas react to key events
  void click() {...} //and mouse click events
  void move() {...} //and mouse move events!
```
Language bits

```javascript
add(media, frame optional, num optional)
// adds media object to arthur canvas

ms() // easy call to current time in ms, returns num

frame(x, y, w optional, h optional)
// add media to specific location on canvas, w/ specific size

cooler() // return a pretty random color

num * { block } // intuitive for-loop
```
The making of arthur

meet a lot

github
social coding

write\LaTeX

# groupme

February 15th 2014 - April 19th 2014
Commits to master, excluding merge commits

spriiiing breaaaaaaaaak
THE TEAM
Translator Architecture

- Arthur source
- Lexer
- Parser
  - Java translation
  - Java translator
  - Whisperer
  - Middleman
  - Augmented JS translation
  - JS translation
  - JS Translator
  - JS library
  - Browserify
  - Java runtime
  - init()
  - add()
  - loop()
Sample time

http://kevin-roark.github.io/arthur/
What have we learned?

- Nothing
- Something
Just kidding!

- The state of media encoding is a mess
- There are lots of libraries out there -- don’t reinvent the wheel, make it better!!!! (but start with the right wheel)
- Demystification of a “compiler”
- Making stuff robust against failure is hard
- You can make whatever you want
Goodbye

Thanks for listening to us talk about arthur. We are proud of arthur and hope you enjoyed its styles.