Intro to Programming

Clarissa Littler

March 28, 2017

What we'll cover

- What is programming?

What we'll cover

- What is programming?
- What are programming languages

What we'll cover

- What is programming?
- What are programming languages
  - . . . and why do we need them?

What we'll cover

- What is programming?
- What are programming languages
  - . . . and why do we need them?
- Basic programming in JavaScript

How this class works

- Lecture

How this class works

- Lecture
- Demonstration

How this class works

- Lecture
- Demonstration
- In-class exercises

How this class works

- Lecture
- Demonstration
- In-class exercises
- Take home supplements

What you'll get out of it

- An introduction to programming

What you'll get out of it

- An introduction to programming
- Enough knowledge to keep going on your own

What you'll get out of it

- An introduction to programming
- Enough knowledge to keep going on your own
- Code you can build off of

Programming is speaking a language

- All language is communication

Programming is speaking a language

- All language is communication
- Programming languages are special languages

Programming is speaking a language

- All language is communication
- Programming languages are special languages
- Computers need precision

Programming is speaking a language

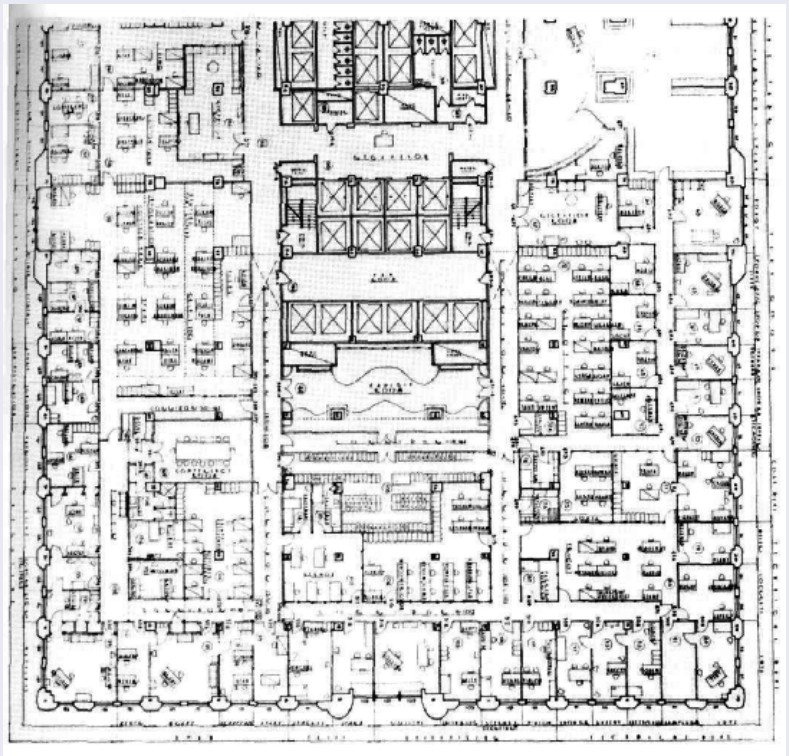
- All language is communication
- Programming languages are special languages
- Computers need precision they're not as smart as us

Why programming is hard

The precision of instructions computers need is unnatural for the human mind

Why programming is hard

Why programming is hard



Why programming is learnable

- Precise thinking may be unnatural

Why programming is learnable

- Precise thinking may be unnatural
- But it's not impossible

Why programming is learnable

- Precise thinking may be unnatural
- But it's not impossible
- It takes time

Why programming is learnable

- Precise thinking may be unnatural
- But it's not impossible
- It takes time and practice

Why programming is learnable

- Precise thinking may be unnatural
- But it's not impossible
- It takes time and practice
- Like learning any language

JavaScript is a programming language

- Many different languages

JavaScript is a programming language

- Many different languages
- One of the most common:

JavaScript is a programming language

- Many different languages
- One of the most common: JavaScript

Why JavaScript?

- JavaScript runs in your browser

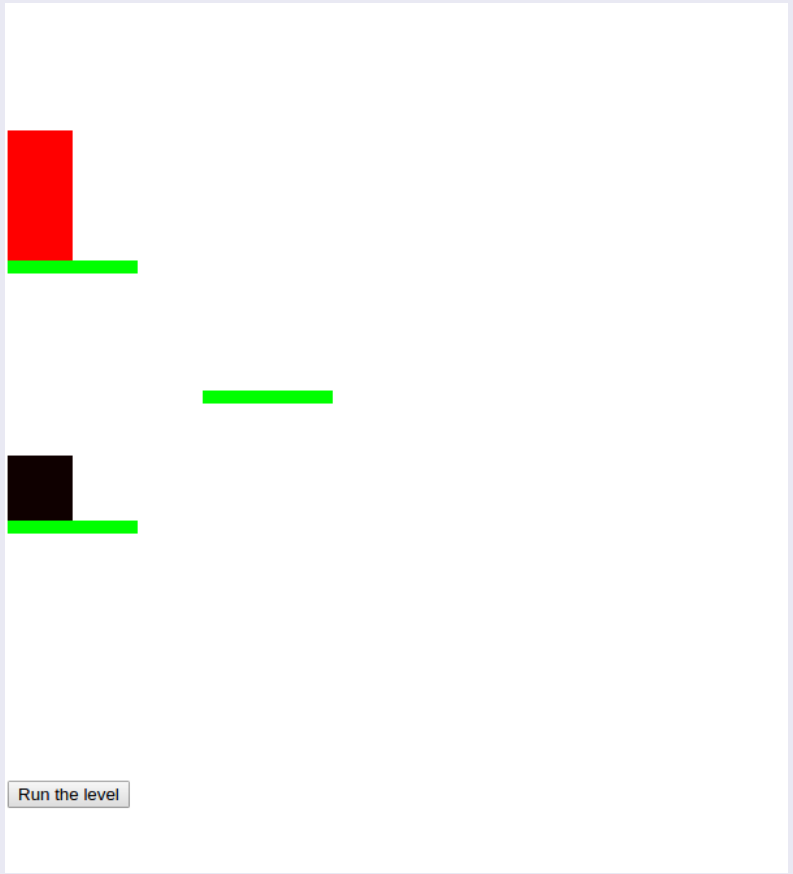
Why JavaScript?

- JavaScript runs in your browser
- JavaScript is a relatively simple language

Why JavaScript?

- JavaScript runs in your browser
- JavaScript is a relatively simple language
- JavaScript lets interact with web pages

How it looks



What it is

Simple puzzles where you need to write code to move your character (the black block) to the goal (the red pillar).

How do we play

Move our hero from the start

How do we play

Move our hero from the start to the finish

How do we play

Move our hero from the start to the finish navigating obstacles

Playing the game

We'll write the JavaScript code to move our hero around, using functions for

Playing the game

We'll write the JavaScript code to move our hero around, using functions for

- walking

Playing the game

We'll write the JavaScript code to move our hero around, using functions for

- walking
- turning

Playing the game

We'll write the JavaScript code to move our hero around, using functions for

- walking
- turning
- jumping

The JavaScript console

- Every browser can run JavaScript

The JavaScript console

- Every browser can run JavaScript
- The `console` allows you to test code

The JavaScript console

Let's try it!

Web pages

- Our code runs in the browser

Web pages

- Our code runs in the browser
- Crafty craftyjs.com

Web pages

- Our code runs in the browser
- Crafty craftyjs.com
- Take advantage of JavaScript's integration

What is a source file

- Code is read in from files

What is a source file

- Code is read in from files
- JavaScript code by convention ends in `.js`

What is a source file

- Code is read in from files
- JavaScript code by convention ends in `.js`
- Code you'll be changing in `pathn.js`

Loading a file

Easiest way to run JavaScript files is to load a page that calls them

Loading a file

```
<!doctype html>
<html>
  <head>
    <script src="testscript.js"></script>
  </head>
  <body>
    Check the console and see what happened!
  </body>
</html>
```

Syntax

- Syntax is the grammar of a language

Syntax

- Syntax is the grammar of a language
- Even stricter rules than human languages

Syntax

- Syntax is the grammar of a language
- Even stricter rules than human languages
- "Dog not can to ridebike nor can to cook"

Syntax

- Syntax is the grammar of a language
- Even stricter rules than human languages
- "Dog not can to ridebike nor can to cook"
- Computers can't guess

Evaluation of code

- Syntax doesn't do anything

Evaluation of code

- Syntax doesn't do anything
- Saying "I have a trillion dollars" doesn't make it so

Evaluation of code

- Syntax doesn't do anything
- Saying "I have a trillion dollars" doesn't make it so
- An *interpreter* runs (or *evaluates*) code

Arithmetic

Numbers

- 1
- 0.5
- -20
- ...

Operations

- +
- -
- *
- ...

Variables

I have a friend, let's call her "Cassandra"...

Variables function both as storage containers and pronouns

Creating Variables

```
var nameOfVariable = initialValueInIt;
var numberOfToes = 10;
```

Assigning variables

Follow along!

```
var musicalsThatShouldExist = "The Walking Dead on Ice";
musicalsThatShouldExist;
musicalsThatShouldExist = "Werner Herzog Sings The Blues";
musicalsThatShouldExist;
```

Example

You try it

Open the console, make a variable, and then try setting it to different values

Functions in math

LaTeX

$$f(x) = x + 10$$

Functions

Functions in JavaScript

```
function f(x) {
    return x + 10;
}
```

Using functions

First example of a function, a function that writes data to the console

```
console.log
```

Multi-argument functions

```
function moreFun (anArgument,anotherArgument) {
    console.log(anArgument + anotherArgument);
}

moreFun(10, 20);
```

Functions with no arguments

```
function noArgs () {
    return 10;
}
```

Example

Navigate to the file `consoleExample.html` and then check the console to see what happened

Example

```
<!doctype html>
<html>
  <head>
    <script>
      console.log("we're printing one message");
      console.log("and another message!");
    </script>
  </head>
  <body>
    Check your console!
  </body>
</html>
```

Moving left and right

- `step` moves forward a step

Moving left and right

- `step` moves forward a step
- `jump` jumps forward

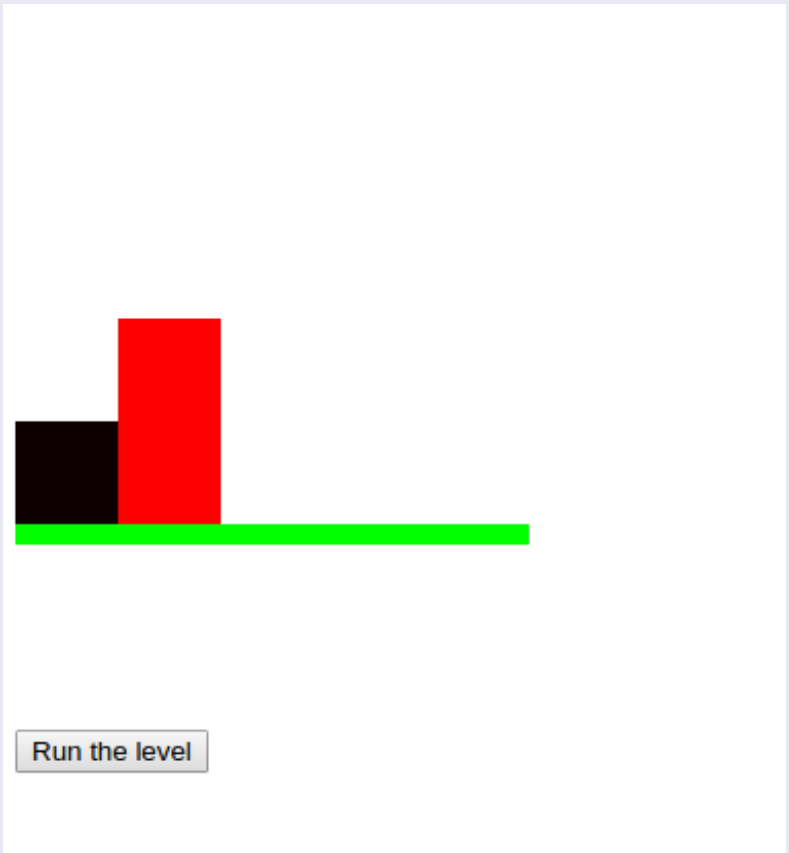
Moving left and right

- `step` moves forward a step
- `jump` jumps forward
- `turnAround` turns you around

Moving left and right

- `step` moves forward a step
- `jump` jumps forward
- `turnAround` turns you around
- steps need to be taken individually

Our first level



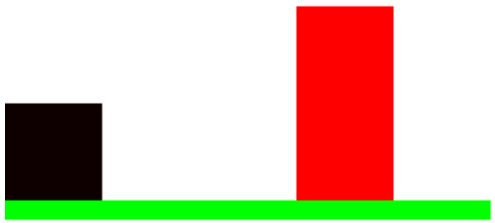
Try it out yourself

Add code to the function `solution` so that our hero moves to the exit

My solution

```
function solution(){
    step();
    finish();
}
```

Our second level



Run the level

Sequences

- Need to do more than a single step of code at a time

Sequences

- Need to do more than a single step of code at a time
- List the steps line by line

Sequences

- Need to do more than a single step of code at a time
- List the steps line by line separate by semicolons

Sequences

```
console.log(1);
console.log(10);
```

Taking multiple steps

How do we sequence actions in JavaScript?

Taking multiple steps

Taking three steps $\Longrightarrow$

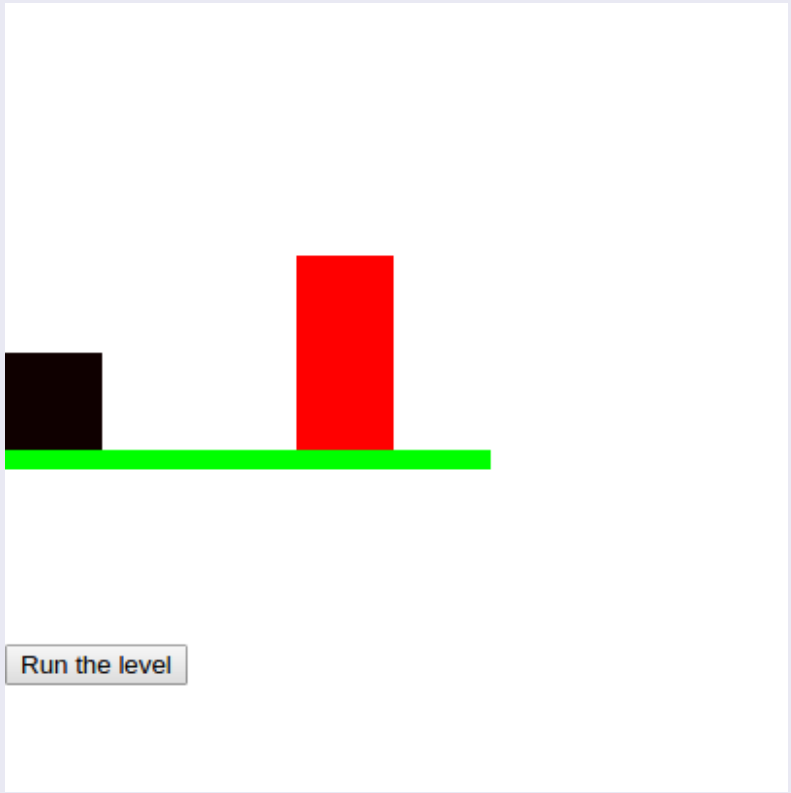
Taking multiple steps

Taking three steps $\Longrightarrow$

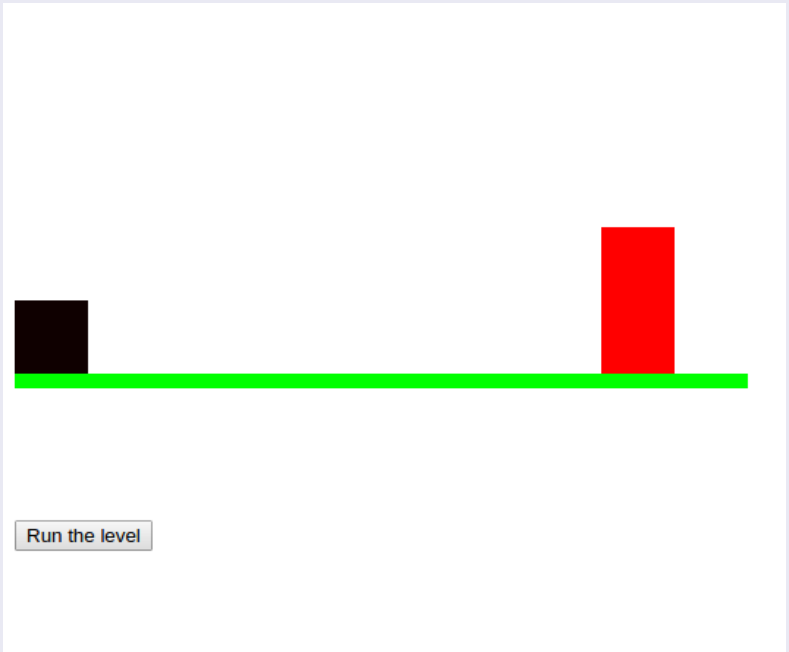
```
step();
step();
step();
```

Our second level

Try solving it!



Our third level



Taking as many steps as needed

- Previous solution isn't helpful

Taking as many steps as needed

- Previous solution isn't helpful
- Who wants to type `step()` again and again?

Taking as many steps as needed

- Previous solution isn't helpful
- Who wants to type `step()` again and again?
- Need a way to repeat, *iterate*, steps

Iteration

- Iteration

Iteration

- Iteration general term

Iteration

- Iteration general term
- Two kinds

Iteration

- Iteration general term
- Two kinds
  - for

Iteration

- Iteration general term
- Two kinds
  - for
  - while

For loops

Definite iteration

Do something a set *number of times*

Chop three onions

Walk five blocks

What is truth?

- `true`
- `false`

Arithmetic comparison

- <
- >
- ==

For-loop syntax

```
for(var i = 0; i < 10; i = i + 1){
    console.log(i);
}
```

Level 3: for-loops

Run the level

While loops

Do *something* while *something* is true.

While loop syntax

```
var i = 100;
while (i > 1) {
    console.log("looped");
    i = i / 10;
}
```

isAtExit predicate

`isAtExit` returns `true` if you're at the exit and `false` if you're not

Negation

- `!  false == true`
- `!  true == false`

Level 3: while-loops

Run the level

Level 3: while-loops

```
while(!isAtExit()){
        ...
}
```

Level 4: jumping

Run the level

Level 5: a lot of jumping

Run the level

Code reuse and redundancy

- Don't want to write jumping code every time

Code reuse and redundancy

- Don't want to write jumping code every time
- We often have things we want to repeat

Code reuse and redundancy

- Don't want to write jumping code every time
- We often have things we want to repeat
- Recall:

Code reuse and redundancy

- Don't want to write jumping code every time
- We often have things we want to repeat
- Recall: functions are chunks of code

Code reuse and redundancy

- Don't want to write jumping code every time
- We often have things we want to repeat
- Recall: functions are chunks of code
- We've seen how to *call* functions

Code reuse and redundancy

- Don't want to write jumping code every time
- We often have things we want to repeat
- Recall: functions are chunks of code
- We've seen how to *call* functions now write them

Writing a function

```
function myFunc () {
        ...
}
```

Writing a function

Function to walk two steps

```
function twostep () {
    step();
    step();
    finish();
}
```

Making a function for jumping

Put it into a function called `platformJump`

platformJump

```
function platformJump(){
    step();
    jump();
}
```

Re-solve level 5

Run the level

Simplifying code

What if we wanted a single function that would step if you're not on a ledge and jump if you're on a ledge?

If-statements

If-statements are how you *choose* what to do based on whether something is *true*

If-statement syntax

```
if (10 < 20){
    console.log("ten");
}
else {
    console.log("twenty");
}
```

atEdge predicate

`atEdge` returns `true` if the player is near an edge and `false` if they are not

safeStep

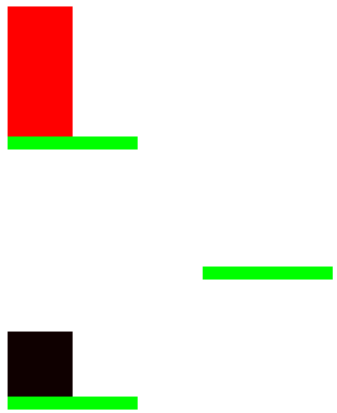
Create a safeStep function

```
function safeStep () {
        ...
}
```

Rewriting level 5

Run the level

Bonus: sixth level



Where do you go from here?

Where do you go from here?

- Continue learning programming!

Where do you go from here?

- Continue learning programming!
- Study guides available

Where do you go from here?

- Continue learning programming!
- Study guides available
  - General progrmaming study guide

Where do you go from here?

- Continue learning programming!
- Study guides available
  - General progrmaming study guide
  - Guide to making a game with Crafty

Where do you go from here?

- Continue learning programming!
- Study guides available
  - General progrmaming study guide
  - Guide to making a game with Crafty
  - Both are continually updated

Where do you go from here?

- Continue learning programming!
- Study guides available
  - General progrmaming study guide
  - Guide to making a game with Crafty
  - Both are continually updated so keep checking back!

What topics are left?

- Objects in JavaScript

What topics are left?

- Objects in JavaScript
- Arrays in JavaScript

What topics are left?

- Objects in JavaScript
- Arrays in JavaScript
- Higher-order functions/closures

What topics are left?

- Objects in JavaScript
- Arrays in JavaScript
- Higher-order functions/closures
- Interacting with the browser

What topics are left?

- Objects in JavaScript
- Arrays in JavaScript
- Higher-order functions/closures
- Interacting with the browser
  - DOM

What topics are left?

- Objects in JavaScript
- Arrays in JavaScript
- Higher-order functions/closures
- Interacting with the browser
  - DOM
  - Events

What topics are left?

- Objects in JavaScript
- Arrays in JavaScript
- Higher-order functions/closures
- Interacting with the browser
  - DOM
  - Events

Thanks

Thanks for attending this course!

References

Study guide is available at:
`https://github.com/clarissalittler/multcolib-lectures/blob/master/BeginnerProgrammingReference.pdf`

There will also be a study guide for learning about making small games with Crafty and a longer tutorial explaining all the code in this repository. So stay tuned!