# Bouncy Ball Coding Tutorials

Wojciech Golaszewski Cosmin Vladianu David Al Mjali Adris Khan

Tutorial 0 – Introduction to programming and the course

Coding Curriculum Group Project

UCL ENGINEERING
Change the world

UCL

# Getting started

## About this course

In this programming tutorial you will learn from scratch how to make a simplified physics simulation, your own version of Flappy Bird and a challenging maze game! No knowledge of programming is assumed, so don’t worry. Every bitty bit of the code we are going to write has been explained as intuitively as it can!

We are going to write our programs in nothing more than a language called JavaScript. The most convenient aspect of programs written in Java Script is that they can be run in every Internet browser. To run such a program we will only need a small HTML file, which would create a web page in which we will be able to run our JavaScript code. This tutorial does not require any knowledge of HTML, however. We will only focus on learning JavaScript.

We will start with programming a very simple physics simulation of a bouncing ball. Then we will use the knowledge we have learned to make a Flappy Ball game (just like Flappy Bird, but with a ball) and finally make another Maze Ball game with challenging obstacle course for the ball to complete. At this point it may sound incredibly ambitious for someone who has done none or little programming before, but do not worry. We will get through writing every program step by step, slowly and with every piece of a code explained.

Alright, let’s get started!

# Getting started

## Downloading stuff

1. Google Chrome

To get on with this tutorial you will need three things. First of all we need an internet browser to run our programs in. To us, Chrome is just the best for the task. You will be able to run your programs in any browser but for developing – yeah, Chrome is the one we recommend. We are going to use it throughout the tutorial.

2. JavaScript – p5.js library

Now we will have to download a JavaScript library called p5.js. Wait, but what is a library? It is nothing else but a set of pre-written programs which provide very useful tools for writing new, more complex, programs. For example if we want to draw a circle we create a function draw_circle(x, y, r) – we give it coordinates of the centre of our circle and its radius, and a circle with these centre coordinates and this radius is drawn on the screen. Functions like these do not exist in JavaScript itself, but many of such have been implemented in p5.js library using some other functions already existing in JavaScript. The great things is we do not have to worry about how JavaScript deals with drawing stuff etc. We just use functions of p5.js and treat them as a part of the JavaScript programming language. You will see how it really looks like when we start coding.

# Getting started

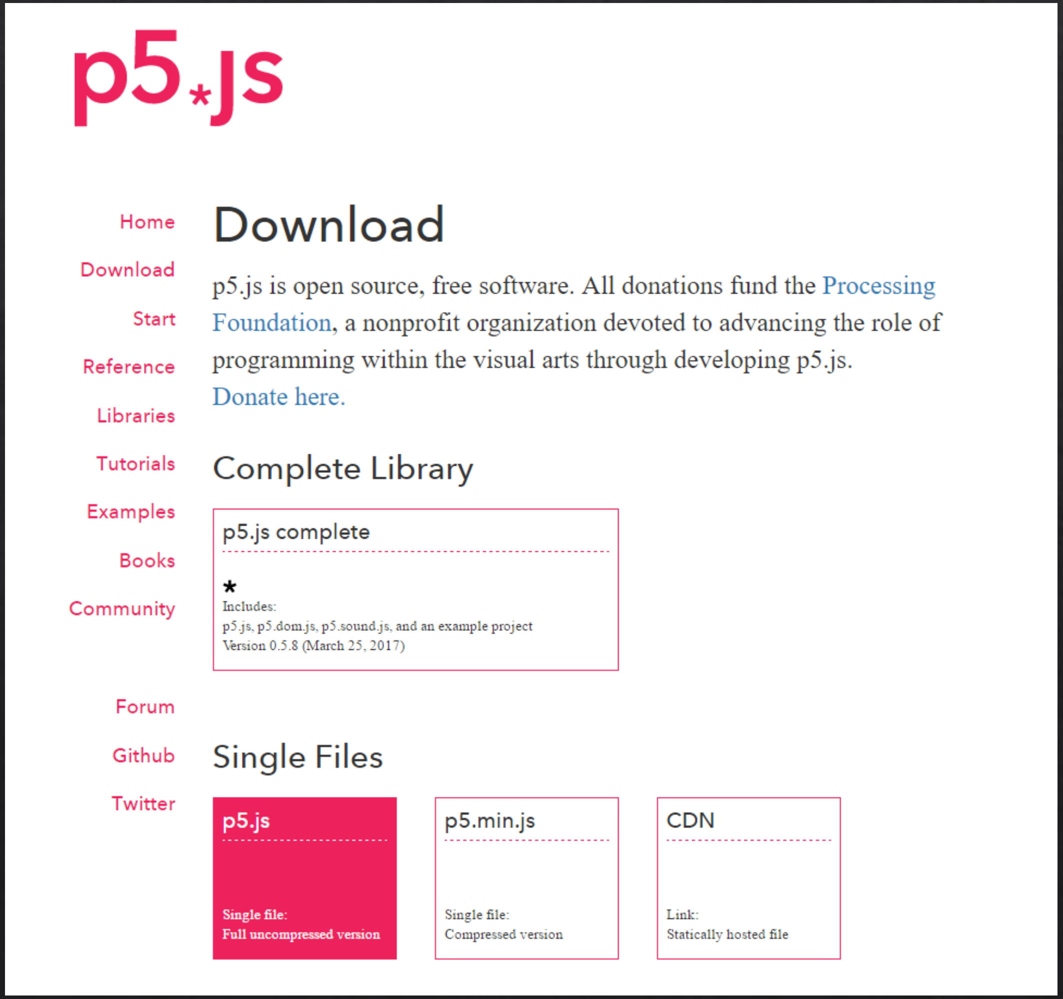
## Downloading p5.js

To download p5.js visit this website:

https://p5js.org/

Then go to Download page and download p5.js as a single file. Just click on the p5.js single file box, the one we have highlighted here in pink.

Create a new empty folder (let's say, call it Bouncy Ball Projects) where you would like to write your programs in. As p5.js is only a library, which means it's just a long piece of JavaScript code, you do not have to install anything! Just keep p5 there.



Download

p5.js is open source, free software. All donations fund the Processing Foundation, a nonprofit organization devoted to advancing the role of programming within the visual arts through developing p5.js. Donate here.

Complete Library

p5.js complete

Includes:
p5.js, p5.dom.js, p5.sound.js, and an example project
Version 0.5.8 (March 25, 2017)

Single Files

p5.js
Single file:
Full uncompressed version

p5.min.js
Single file:
Compressed version

CDN
Link:
Statically hosted file

# Getting started

## Text Editor

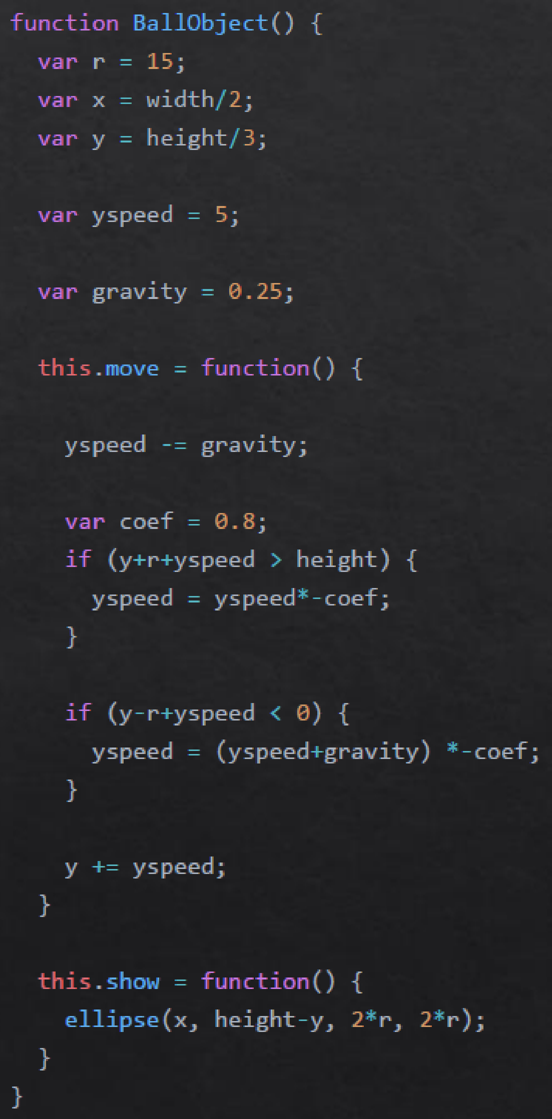
3. Atom – text editor

We also need a nice editor to write our code in. And indeed, whether it is nice or not is the most important aspect here. You see, we could actually write our programs in a simplest text editor like Notepad or text edit, but it is also very useful to have some handy tools.

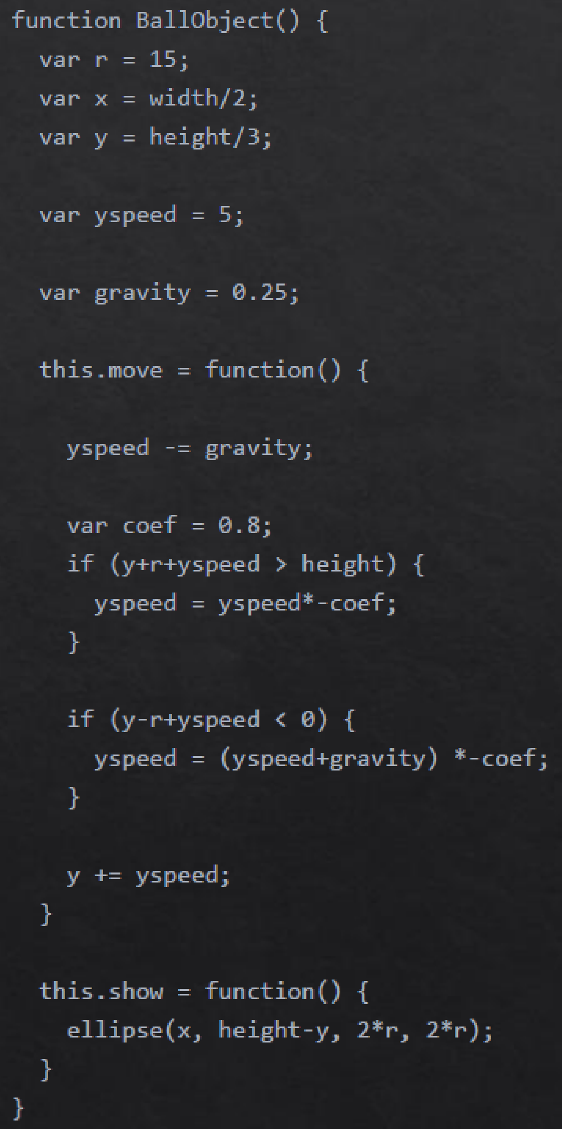
First of all – syntax highlighting. Have a look at the code right here. These are the exactly same piece of code, but with and without coloured syntax. See for yourself which one is more pleasant to look at! Colouring is one of the first few steps of making writing programming easier and more organised. If you decide to do some more programming in the future and do complex projects you will need to learn how to organise your code, for example dividing it into separate files when the code gets too long.

Among many others, text editors give you many great tools to make your coding life easier (as some point they are no longer simply called text editors, but rather working environments).

I think that Atom is an editor which you will find very useful and pleasant to work in! You can, of course, choose another editor, but if you don't have any favourite one yet – we recommend Atom!



```
function BallObject() {
  var r = 15;
  var x = width/2;
  var y = height/3;

  var yspeed = 5;

  var gravity = 0.25;

  this.move = function() {

    yspeed -= gravity;

    var coef = 0.8;
    if (y+r+yspeed > height) {
      yspeed = yspeed*-coef;
    }

    if (y-r+yspeed < 0) {
      yspeed = (yspeed+gravity) *-coef;
    }

    y += yspeed;
  }

  this.show = function() {
    ellipse(x, height-y, 2*r, 2*r);
  }
}
```



```
function BallObject() {
  var r = 15;
  var x = width/2;
  var y = height/3;

  var yspeed = 5;

  var gravity = 0.25;

  this.move = function() {

    yspeed -= gravity;

    var coef = 0.8;
    if (y+r+yspeed > height) {
      yspeed = yspeed*-coef;
    }

    if (y-r+yspeed < 0) {
      yspeed = (yspeed+gravity) *-coef;
    }

    y += yspeed;
  }

  this.show = function() {
    ellipse(x, height-y, 2*r, 2*r);
  }
}
```

# Getting started

## Downloading Atom

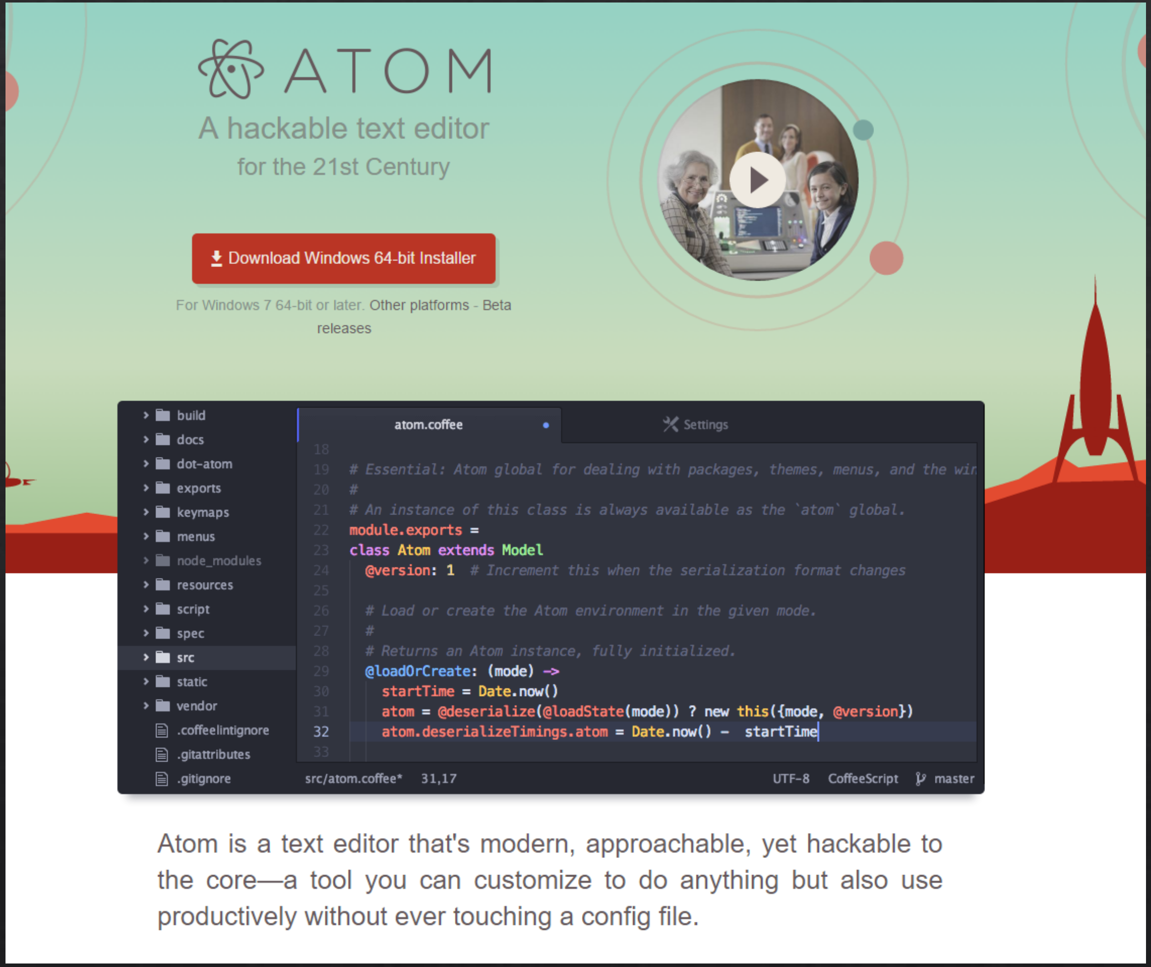
To download Atom visit this website:

https://atom.io/

Then go to Download page and download p5.js complete package. This is what should appear on your screen:

Download Atom and install it.

Now let's see how to configure eveything we've downloaded.



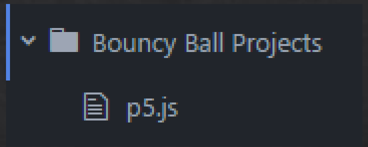
```
# Essential: Atom global for dealing with packages, themes, menus, and the wir
#
# An instance of this class is always available as the `atom` global.
module.exports =
class Atom extends Model
  @version: 1  # Increment this when the serialization format changes

  # Load or create the Atom environment in the given mode.
  #
  # Returns an Atom instance, fully initialized.
  @loadOrCreate: (mode) ->
    startTime = Date.now()
    atom = @deserialize(@loadState(mode)) ? new this({mode, @version})
    atom.deserializeTimings.atom = Date.now() -  startTime
```

Atom is a text editor that's modern, approachable, yet hackable to the core—a tool you can customize to do anything but also use productively without ever touching a config file.

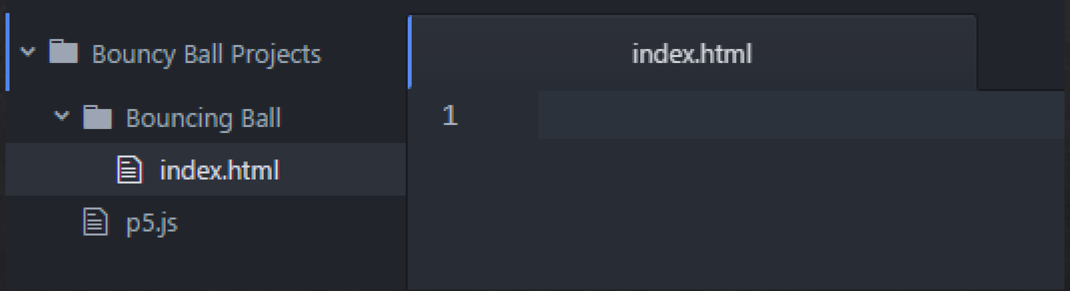
# Creating your first project

Open Atom. You will see a nice welcome message and a settings tab, where you can set up Atom. There are many of ways to customize Atom, it's all matter of preference. First thing I like adjust in every new editor is to set the theming to be dark (dark background exhausts my eyes least, that's why I like it). My favourite in Atom is called 'One Dark'. You can find it in ‚Themes' settings tab.

Let's configure our project. In top left corner of the screen click File tab and choose Open Folder. Now navigate to the folder where you have put p5.js file in. Enter it and click Select Folder. In left top corner, now you should see your project folder opened in Atom:



Now, click on the folder with right mouse button and click New Folder for our first project. You may call it Bouncing Ball, which is going to be our first program. Now, right-click on the newly created folder and create a new file. Call it ‚index.html'. For now, you should see this:

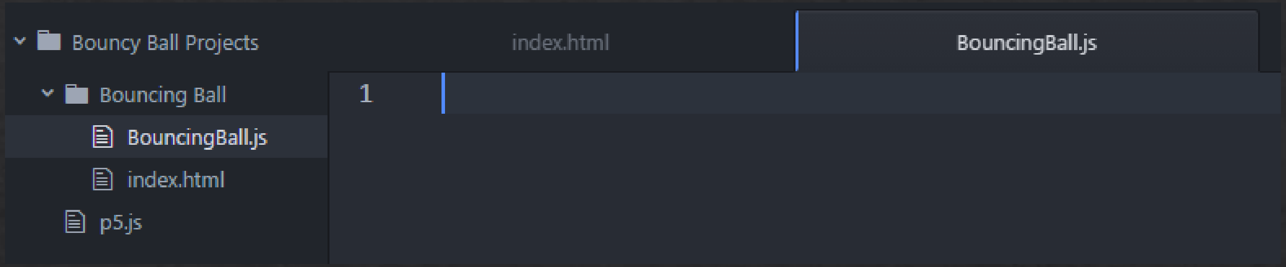


The html file is the one which you will launch in your browser to run  JavaScript programs you will write. You may even open the file now, but you will see nothing but mysterious… whiteness! We will do something about it in a minute.

Oh, one more thing. In programming, you can name files whatever you like, just without using spaces (usually, at least). I called it ‚index' because that is how sample html files are usually called and to make it easily distinguishable. Of course, you can name it differently, whatever suits you most ;-)

# Configuring HTML file – final step before getting to programming

Now, let's create the JavaScript file. Again, right-click on the Bouncing Ball project and create a new file. Call it ' BouncingBall.js '. Now, make sure you can see everything like this:



Great! Let's configure HTML file. This is not a course about HTML, so I'll go through it quickly. Copy this piece of code into your index file:

```
<html>
  <head>
  </head>

  <body>
  </body>
</html>
```

This is an HTML file template. Everything happens between opening <html> and closing </html> elements (/ denotes end of an element). In <head></head> we put commands which describe data used in the file. We will insert references to our JavaScript files here. <body></body> tags describe content of a page. However, as we are not going to display anything in our webpage apart from our JavaScript file's content, we are leaving this blank.

# Configuring HTML file – final step before getting to programming

In head, write this line:

```
<head>
  <title>Bouncing Ball</title>
</head>
```

The content that you will put between title tags is the title of the webpage that will be displayed on the top of your browser window. You can write whatever you want between the title tags. Now let's include p5.js and our code into the webpage. Write this inside head:

```
<script language="javascript" type="text/javascript" src="../p5.js"></script>
<script language="javascript" type="text/javascript" src="BouncingBall.js"></script>
```

`src="../p5.js"` and `src="BouncingBall.js"` describe paths to our p5.js and Bouncing Ball files. As p5.js file is in another folder, we have to tell html that is it so. ../ denotes that we exit the folder which we are in ad browse the path from there. As p5.js file is in one folder above we write "../p5.js" to access it. BouncingBall.js is in the same folder the index.html file is in, so we just write `"BouncingBall.js".` Make sure that the file names you write here are exactly the same as the ones you are going to write your code in.

Finally, add this line to remove margins of your webpage. This will allow us to display our javascript drawings in the upper left corner.

```
<style> body {padding: 0; margin: 0;} </style>
```

## Configuring HTML file – final step before getting to programming

In the end, your HTML file should look like this:

```
<html>

  <head>
    <title>Bouncing Ball</title>
    <script language="javascript" type="text/javascript" src="../p5.js"></script>
    <script language="javascript" type="text/javascript" src="BouncingBall.js"></script>
    <style> body {padding: 0; margin: 0;} </style>
  </head>

  <body>
  </body>

</html>
```

Now we are ready to start programming a Bouncing Ball!