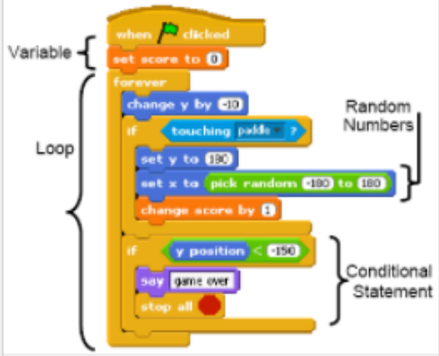
Introducing Tutors

a small tool for generating instructional material

A collection of recent Modules in Modern Computer Science

Programming - Introductory

Programming Fundamentals I (Scratch)



This is an introductory Programming module and assumes no prior knowledge of programming. For the first week or two, we will use Scratch to teach some basic programming concepts. We will work through non-complex problems that will introduce you to the basic constructs of programming languages i.e. Sequence, Selection and Loops.

Dr. Siobhan Drohan & Ms. Mairead Meagher

Programming Fundamentals I (Processing)

This is an introductory Programming module and assumes no prior knowledge of programming. We will use the Java programming language through the Processing Development Environment (PDE). We will work through non-complex problems that will introduce you to the basic constructs of programming languages i.e. sequence, selection and loops. You will also learn to use different data types and manipulate the data.

Dr. Siobhan Drohan & Mairead Meagher

Programming Fundamentals (Java)

The objective of this module is to provide a basic introduction to the Java language. The course assumes no prior programming experience, begins at a slow pace with simple concepts but progressively adds more complex topics. It is essential to gain a fundamental understanding of the language as later course modules such as Web and Android development use Java extensively.

John Fitzgerald

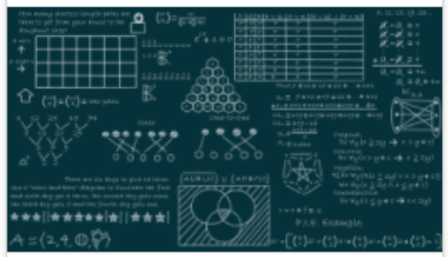
Programming - Intermediate

Data Structures

The objective of this course is to provide a comprehensive introduction to developing modern Java Applications built around Data Structures. Our objective is to provide the student with the skills required to construct efficient and reliable Java applications of moderately complexity.

David Drohan & Peter Carew

Algorithms



An introduction to algorithms in Java

Frank Walsh & Eamonn De Leastar

topics

Web Site Dev

Top Level Topics

Portfolio

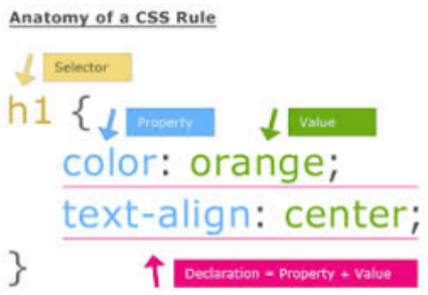
All Slides

All Labs

Introducing HTML

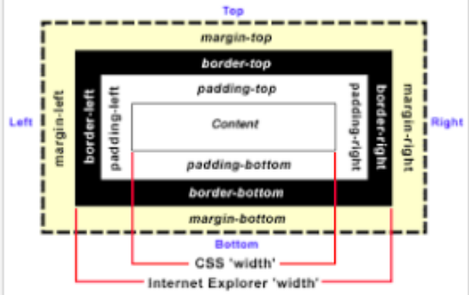
We explore the foundations of web and get to grips with the fundamentals of the HTML language. As you will see, its structure and format is relatively straightforward, and you will be able to understand the basics very quickly. We will be focusing on a small number of 'tags' to get started, and also on the ways in which different html files can be linked together to form a site.

Introducing CSS



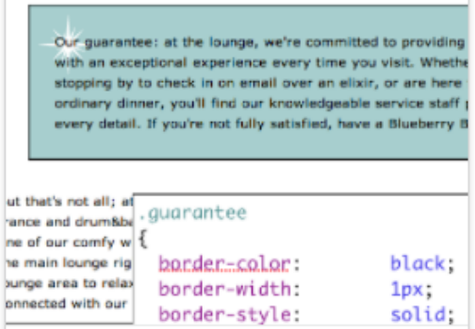
CSS is the language we use the style HTML. This language looks simple at first, but as we will see in the next few weeks, it is considerably more complex than HTML and will require a very careful approach to get right

The Box Model



In order to style the same html elements in different ways we need to use classes. This allows us to target specific occurences of an html element for styling purposes. At the heart of the layout engine in web browsers is a concept called the 'box model'. This defines a general layout structure for all HTML elements, providing a language for specifying important dimensions and relationships to other elements.

Layout

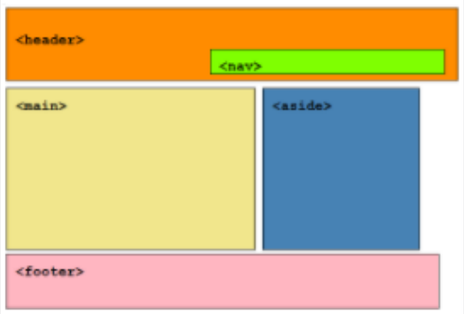


Using an understanding of the fundamental features of the box model we can start to produce more interesting page layouts. Specifically, we can break a page down into sections and use box model properties to dimension and position these sections in a flexible manner. This will allow is to grow multi-column pages that can vary according to the size of the browser windows used to view them.

Navigation

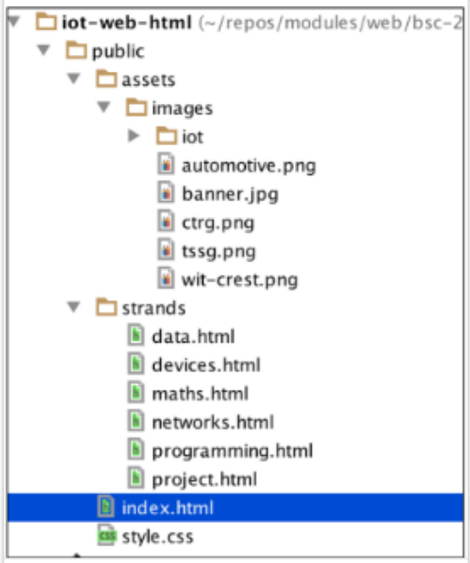
Central to a well design site is a clear and understandably navigation structure. This must easily allow the user to explore the site, provide sufficient context such that the user knows where they are at any stage, and do this in a visually pleasing and efficient manner.

Semantic HTML + More Layout



HTML5, the latest version of the standard, introduced a range of new elements. Among the most interesting are the so-called 'semantic' elements. These attempt to re-examine the proliferation of DIVs in html, and proposed an alternative vocabulary that would better reflect the purpose of many of these DIVs

Layout + Review

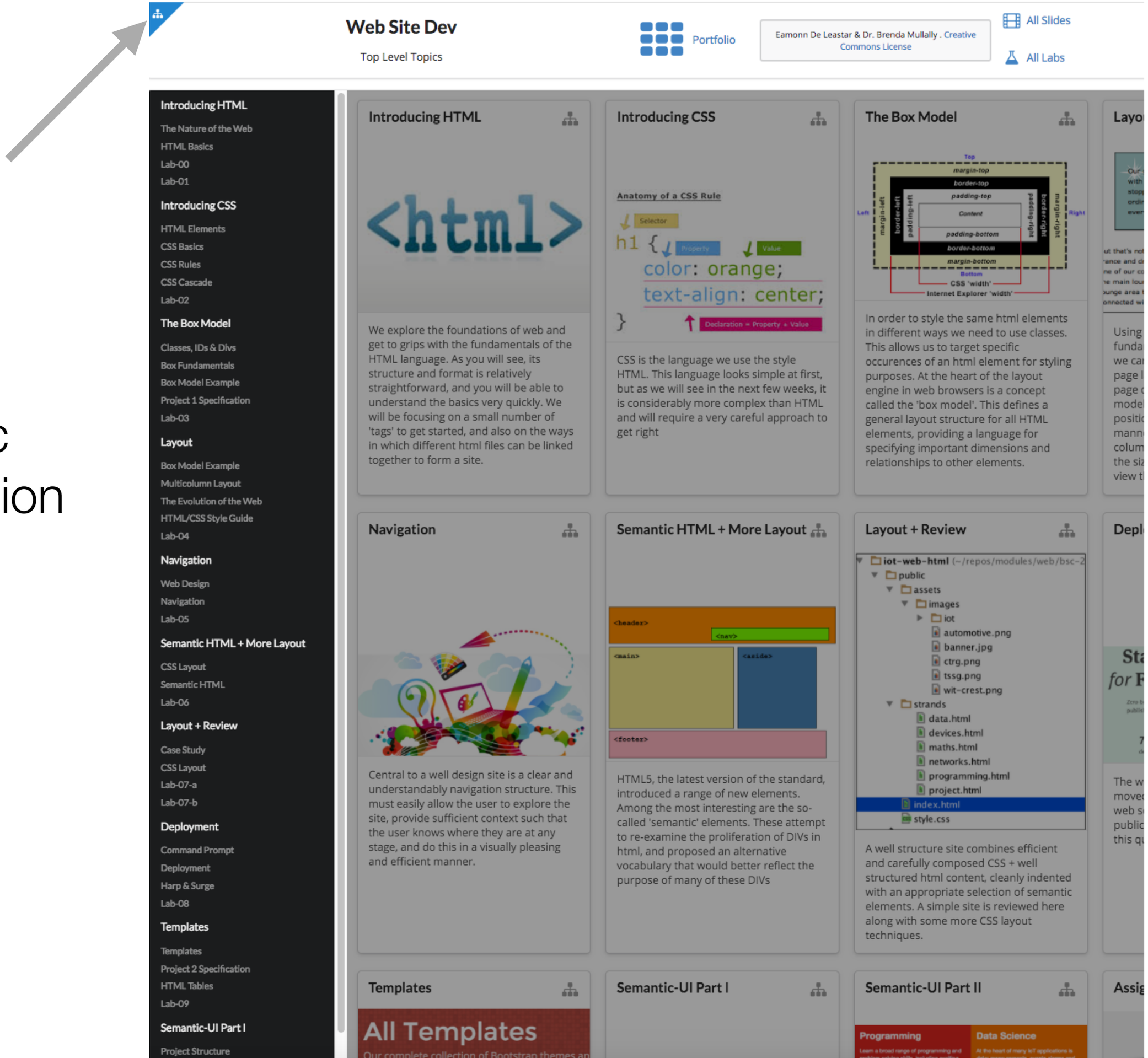


A well structure site combines efficient and carefully composed CSS + well structured html content, cleanly indented with an appropriate selection of semantic elements. A simple site is reviewed here along with some more CSS layout techniques.

Deployment

The web site will ultimately have to be moved from your local folders to a public web server where it can be accessed via a public domain. Modern tools can make this quite seamless and convenient.

topic navigation



12: APIs

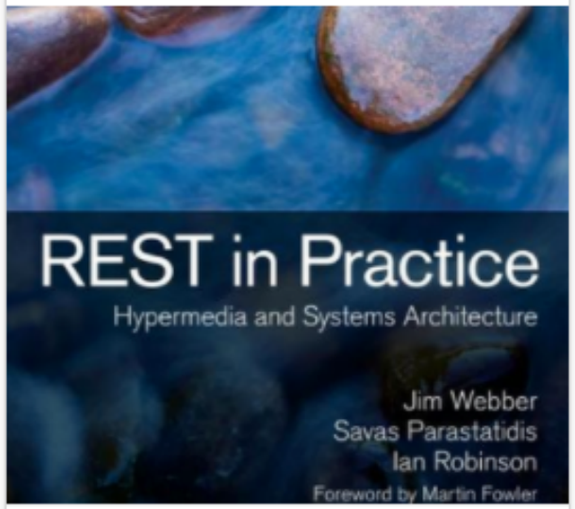
Single Topic

Module

All Slides

All Labs

APIs



An Application Programmer Interface is the published set of http endpoints and messages that a service can support. API design and implementation is a rich field of study - here we take a general overview.

Endpoints

Expose access to the Candidates model as a REST endpoint. This involved defining new routes and handlers, which respond simple JSON representations.

Testing Endpoints

Tools like Postman and Insomnia usefully exercise endpoints. However, we can also exercise them problematically, which offers some significant advantages.

Lab-12 Apis

Start the development of an API for the donation service, focusing initially on providing access to the Candidates model. Implement the API using simple REST principles.

topic contents - talks & labs

Using Asserts

- You could use this assert to check all sorts of things, including whether numbers are equal to each other.

```
int a = 2;
//...
assertTrue (a == 2);
```

- To check that two integers are equal, a method that takes two integer parameters might be more useful.

```
public void assertEquals (int a, int b)
{
  assertTrue(a == b);
}
```

- We can now write the first test a little more expressively:

```
int a = 2;

assertEquals (2, a);
```

labs

steps

formatted text

images

syntax-highlighted source code

Get Candidate Endpoint

The first endpoint we have just implemented retrieves all candidates. We can also introduce a route to retrieve a single candidate:

routesapi.js

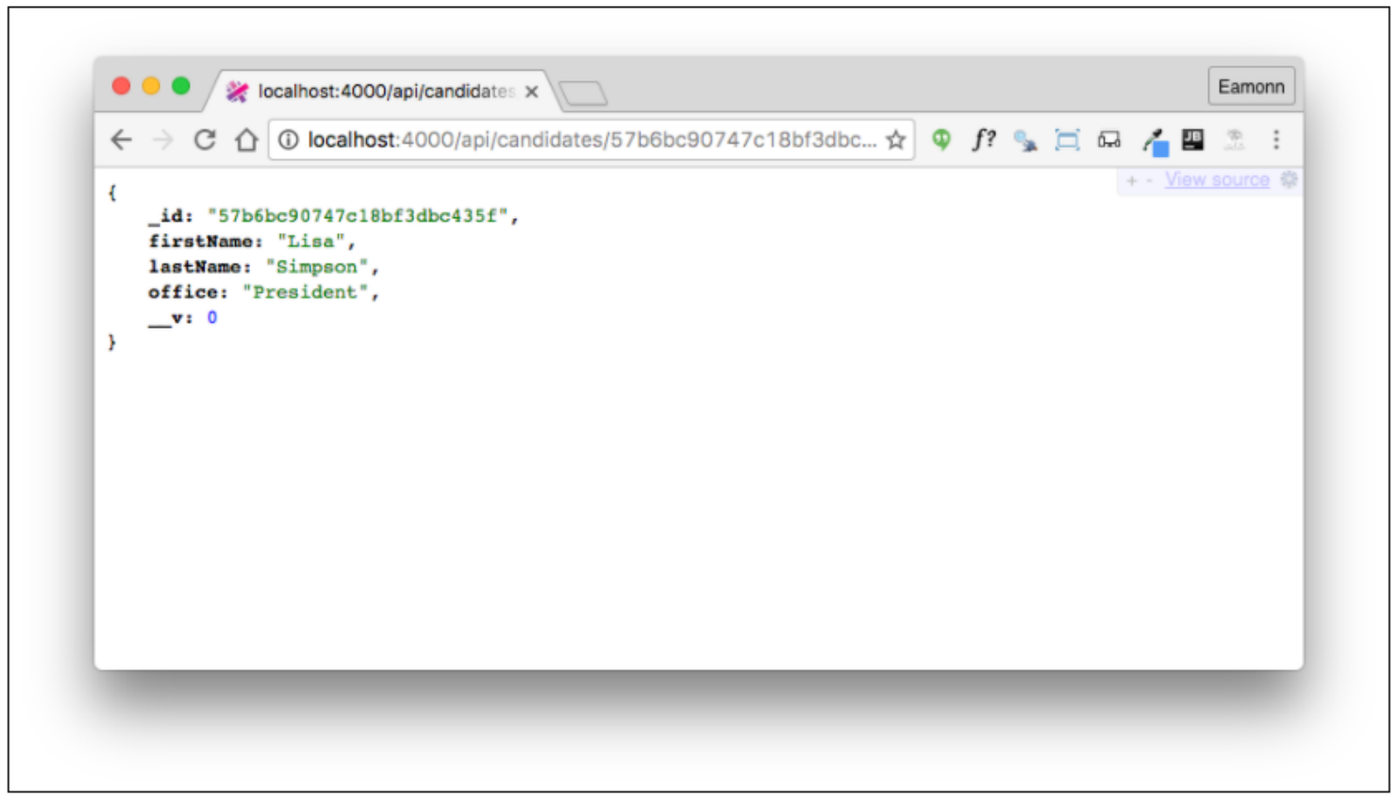
```
  { method: 'GET', path: '/api/candidates/{id}', config: CandidatesApi.findOne },
```

app/api/candidatesapi.js

```
exports.findOne = {

  auth: false,

  handler: function (request, reply) {
    Candidate.findOne({ _id: request.params.id }).then(candidate => {
      reply(candidate);
    }).catch(err => {
      reply(Boom.notFound('id not found'));
    });
  },

}
```

In order to retrieve the candidate, we will need the ID for the candidate of interest:

- http://localhost:4000/api/candidates/57b6bbd3a11377b03d31da0a



The Id changes every time we launch the application, as our database seeder clears all collections each time.

If we specify an unknown id, Boom will generate the appropriate error:

Programming Fundamentals I (Processing)

All Slides in the Course

Module

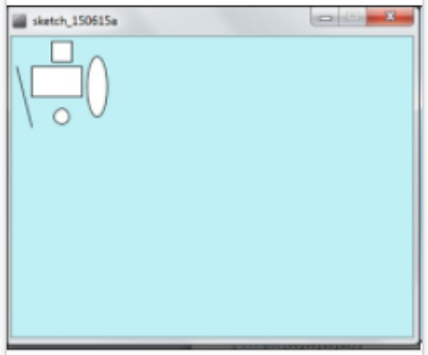
All Slides

All Labs

Introduction to the PDE

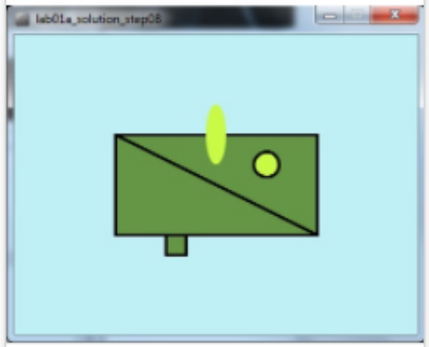
We start by exploring Processing and then looking into the Processing Development Environment (PDE).

Static Drawings



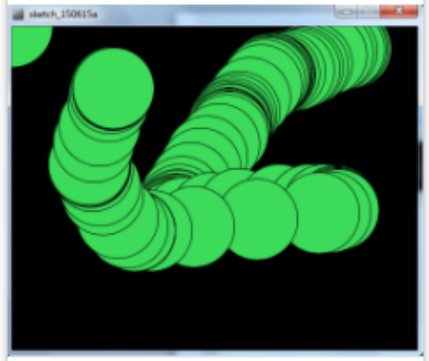
Here we will start to code. In particular, we will step through the creation of static drawings using basic shapes. You will also cover Grayscale and RGB colour schemes.

Formatting Shapes



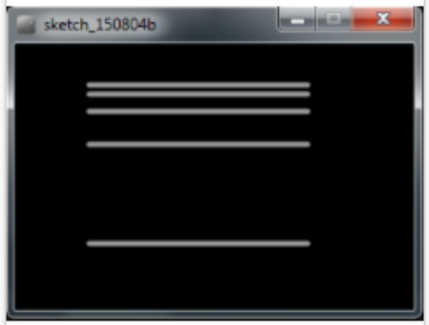
Here we will format basic shapes with colour and outline. We will also look at commenting your code.

Basic Animation



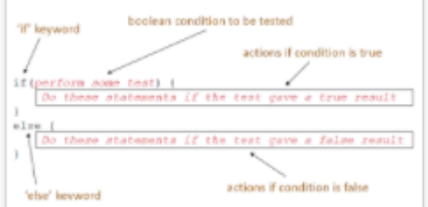
We start by exploring the setup() and draw() functions that animate our drawings. We will also look at system variables that come with Processing.

Data Types



We will investigate Java's primitive data types and learn about some arithmetic operators that we can use with them.

Conditional Statements

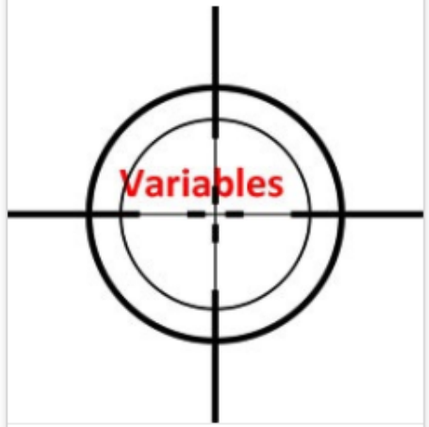


We will learn how to write conditional statements (if statements) and boolean expressions in Java. We will also learn about logical operators.

Mouse Events

We will learn how to handle mouse events. We will also do a recap on Arithmetic Operators but this time, we will look at the order of evalation of these operators.

Scope of Variables



We will look at the principles behind where a where a variable is availble for use. Also we look at some nice new assignment statements.

Slide Wall

Programming Fundamentals I (Processing)

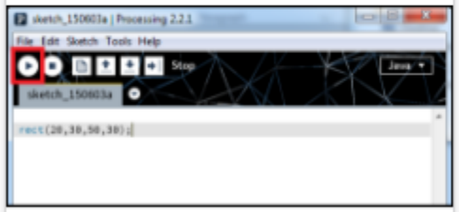
All Labs in the Course

Module

All Slides

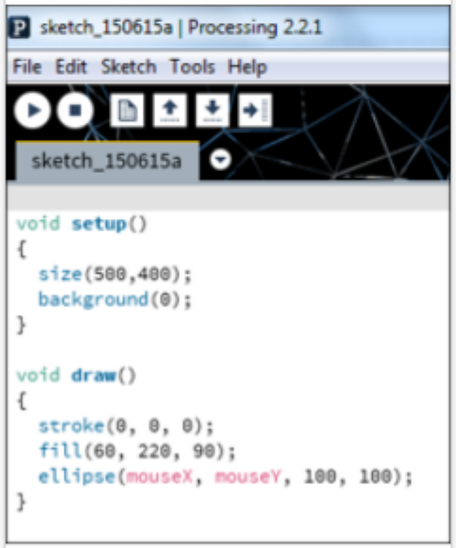
All Labs

Lab-01



On completion of this lab you should:

Lab-02

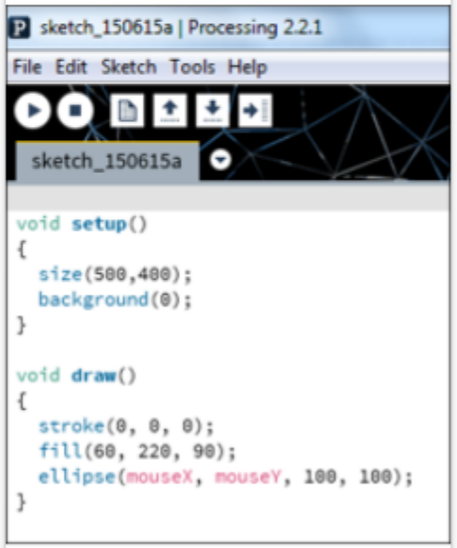


On completion of this lab you should:

Code to Download

On completion of this lab you should:

Lab-03



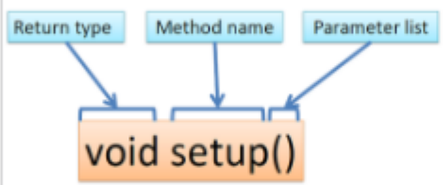
On completion of this lab you should be able to code animated drawings using the following constructs:

Lab-04

```
int yCoordinate = 60;

size(600, 300);
background(102);
fill(255);
noStroke();

for(int i = 0; i < 4; i++)
{
    rect(50, yCoordinate, 500, 10);
    yCoordinate = yCoordinate + 20;
}
```

On completion of this lab you should understand variable scope and be able to code static drawings using for and while loops.

Lab-05

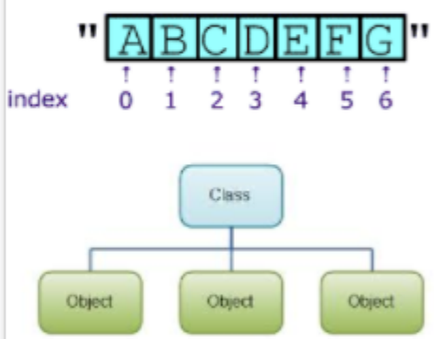


- On completion of this lab you should be able to use methods to handle mouse events and also be able to write your own methods.

Lab-06

- On completion of this lab you should be able to write more sophisticated methods, particularly using parameters and returning data. You will also learn how to use the String methods.

Lab-07



On completion of this lab you should:

Lab Wall

lab navigation

Programming Fundamentals I (Processing)

All Labs in the Course

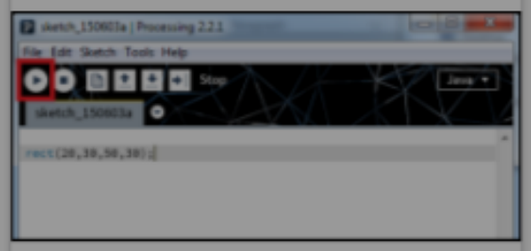
Module

All S

All L

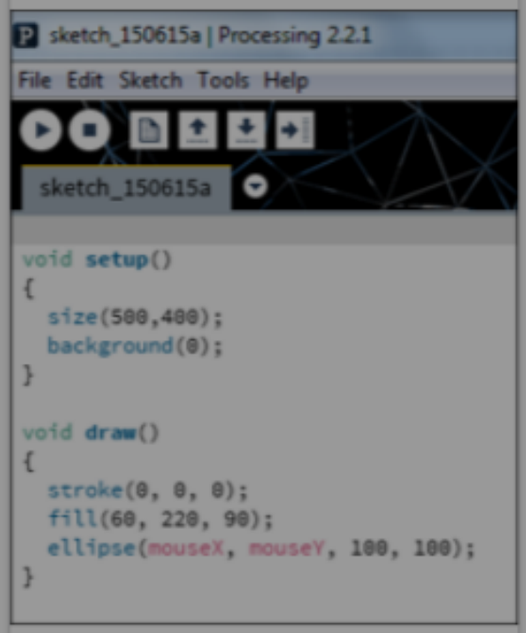
- Lab-01
- Lab-02
- Code to Download
- Lab-03
- Lab-04
- Lab-05
- Lab-06
- Lab-07
- Lab-08
- Lab-9a
- Lab-9b
- Lab-9c
- Lab-10a
- Lab-10b
- Lab-10c
- Formative Assessment
- Assignment-1
- Assignment-2

Lab-01



On completion of this lab you should:

Lab-02



On completion of this lab you should:

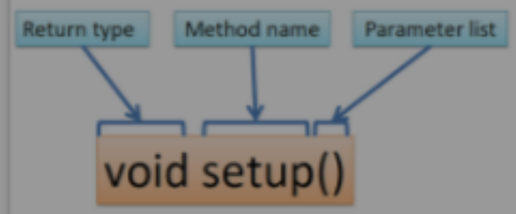
Code to Download

On completion of this lab you should:

Lab-04

```
int yCoordinate = 60;

size(600, 300);
background(102);
fill(255);
noStroke();

for(int i = 0; i < 4; i++)
{
    rect(50, yCoordinate, 500, 10);
    yCoordinate = yCoordinate + 20;
}
```

Lab-05



- On completion of this lab you should be able to use

Lab-06

- On completion of this lab you should be able to

slide navigation

Programming Fundamentals I (Processing)

All Slides in the Course

Module

Dr. Siobhan Drohan & Mairead Meagher . Creative Commons License

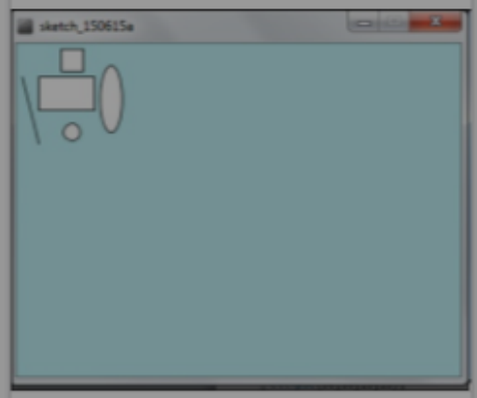
All Slides

All Labs

Introduction to the PDE

We start by exploring Processing and then looking into the Processing Development Environment (PDE).

Static Drawings

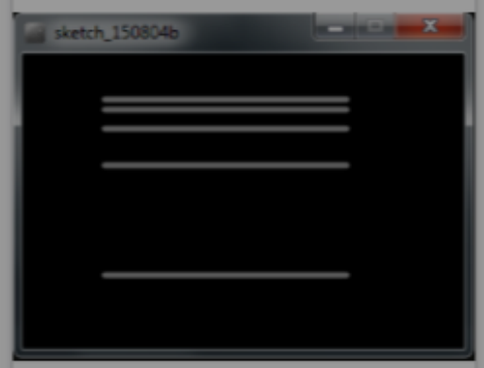


Here we will start to code. In particular, we will step through the creation of static drawings using basic shapes. You will also cover Grayscale and RGB colour schemes.

Formatting S

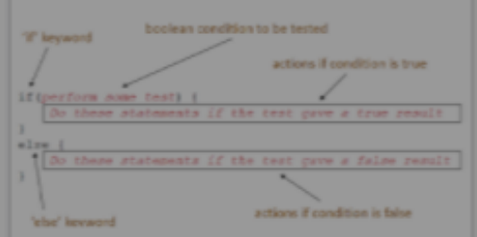
Here we will for shapes with col We will also look commenting yo

Data Types



We will investigate Java's primitive data types and learn

Conditional Statements



We will learn how to write conditional statements (if statements) and boolean

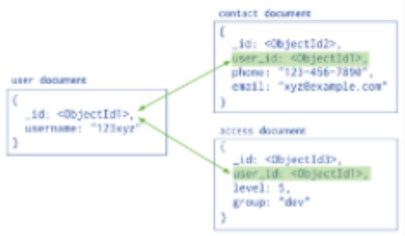
Mouse Event

We will learn ho mouse events. recap on Arithm

Enterprise Web Development

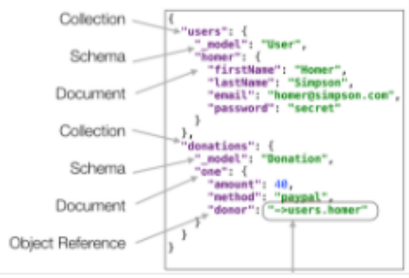
Database Seeding

Relationships between Mongo Documents



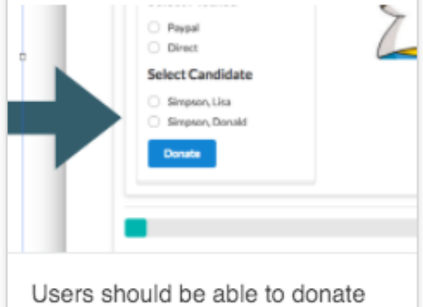
Creating and maintaining relatopnships between mongo documents enable powerful models to be constructed and queried.

Mongoose Seeding



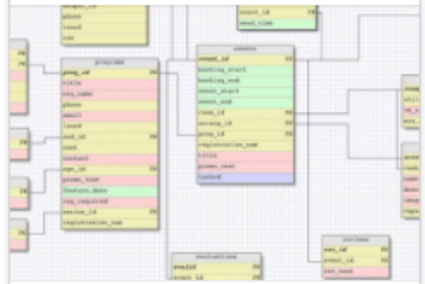
Seeding the database can simplify exploratory development, prepopulating the database with simple test data during development.

Candidate Model



Users should be able to donate to different candidates. We extend the model to include Candidates, incorporating candidate references into donations.

Lab-11 Seeding



Include a mongoose seeder component in the application. Use this to validate a new Candidate model, preloading it with a json specified object graph.

topics ‘landed’ into moodle