Name: …… Class: ……

Getting Started with JS-Eden

This activity will guide you through an introduction to JS-Eden.

Starting JS-Eden

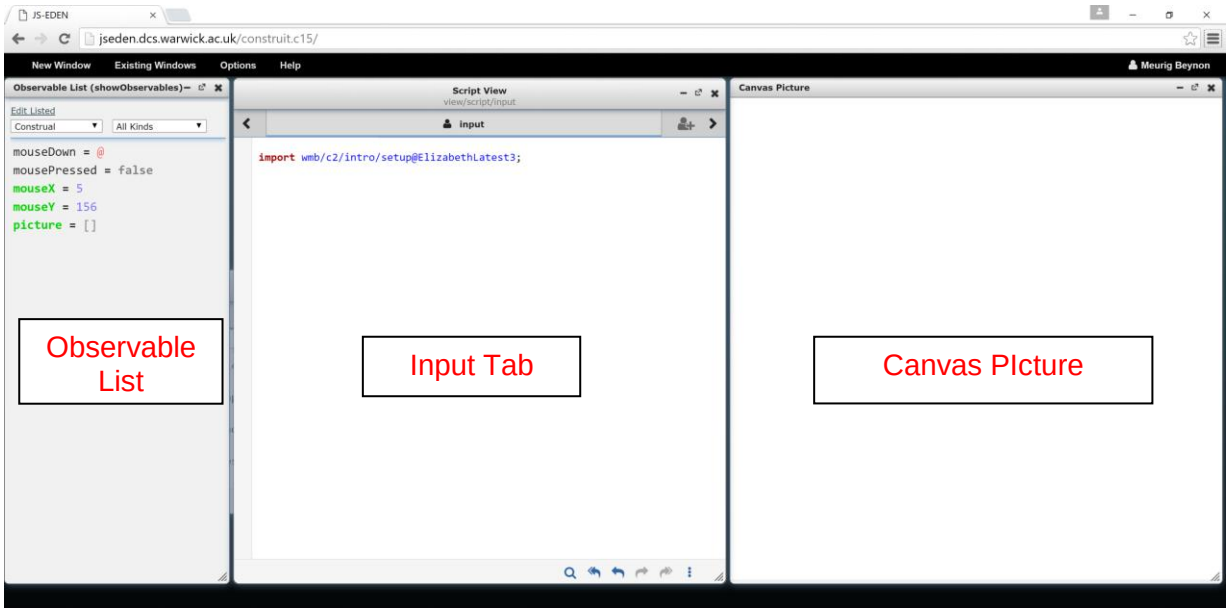
You can open JS-Eden by pointing your web browser to:

http://jseden.dcs.warwick.ac.uk/construit.c15/

When you first start JS-Eden, you will see two windows: Canvas Picture on the left and Script View on the right. The Script View has a ‘tab’ labelled ‘input’ within it. Click your mouse in the top left hand corner of this tab. An input text prompt will appear. Type the command:

import c15/worksheet1/setup;

If you place the mouse over the grey region (“the gutter”) at the left of this command, an arrow symbol will appear. Left click on this arrow to set up the windows for the Getting Started with JS-Eden exercise. From left to right on the screen there are three windows labelled ‘Observable List’, ‘Script View’ and ‘Canvas Picture’.



In the JS-Eden environment, we can construct models using three concepts: observables, dependencies and agency. Let's get started by showing how to create observables and dependencies through the Input Tab.

Entering text input

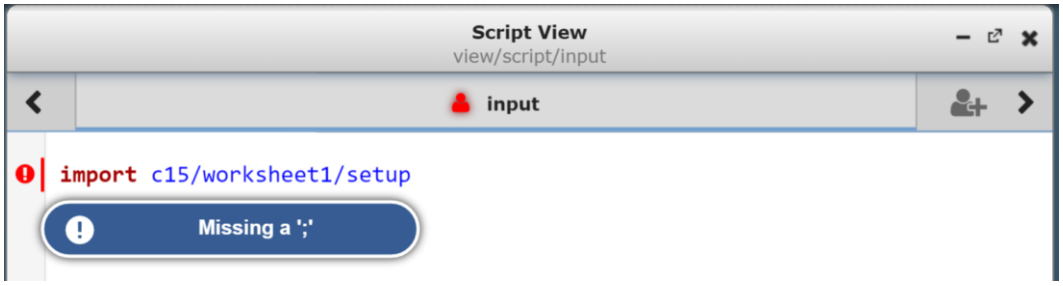
For this activity, you will need to enter text into the Input Tab. You can enter a line of input by following the steps you used to input the command highlighted in red above:

1. Click your mouse in the top left hand corner of the Input Tab.
2. Type your input at the input prompt.
3. “Execute the input”: Place the mouse in the gutter to the left of each input line and left-click on the dark grey arrow symbol that appears.

Step 3 is called ‘executing’ the input. You will only be able to do step 3 when your input is valid. For instance, if you delete the semi-colon from the command you entered before, so that the text to be input is

Import c15/worksheet1/setup

you will see that the gutter is blocked by an exclamation mark in a red circle. Click on the red circle and the error message “Missing a ‘;’” appears:



You can enter several lines of text into the Input Tab, each with a semi-colon at the end. You can execute each line separately, as in Step 3 above, or select several lines by holding and running the mouse down the gutter and executing all of them together using the one dark grey arrow.

There are some other buttons at the bottom of the Input Tab that you will find useful:

You can press the Rewind button at any stage to clear the Input Tab. Do this now so that you start with a blank Input Tab. The other buttons will be introduced later.

The Input Tab

The Input Tab is the primary place to create the observables and dependencies that make up a model.

Creating observables

Type in the Input Tab:

```
a = 5;
b = 6;
```

Then left click in the gutter to the left of each line to 'execute' it. You can inspect the current values of a and b in the Observable List. Here are some other types of observable:

```
myReal = 1.234;
myString = "Hello World";
myList = [myReal, myString];
```

There are these 4 basic types of observables in JS-Eden: integer, real, string, list.

JS-Eden has operators and functions like those in traditional programming languages. Here are some mathematical operators:

```
a = 20 + 15;
b = a * 2;
c = a % 5;
```

And here are some mathematical functions to try:

```
x = max(2,8);
y = sqrt(16);
z = floor(5.9);
```

Task 1: Write the answer and the mathematical meaning of the three functions.

10 % 3	**result ……**	**meaning …….**
sqrt(81)	**result ……**	**meaning …….**
floor(2+3.6)	**result ……**	**meaning …….**

Another function we will use is the random function:

```
r = random();
s = floor(random() * 10);
```

The random() function generates a random real between 0 and 1.

Creating dependencies

As shown above, you create a new observable called c:

```
c = a + b;
```

The observable c should now have the value of a+b. However, there is another way of recording the value a+b, using dependency:

```
d is a + b;
```

Now c and d should have the same value. Try changing the value of a:

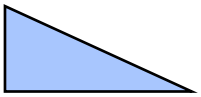
```
a = 1;
```

Are c and d still the same? They are not, because d has changed. It has updated to maintain the 'dependency' of d is a+b. In JS-Eden, 'is' is used to create dependencies between observables, much like the dependencies between cells in a spreadsheet.

*In JS-Eden, we say that "c is a **Base Observable**", and "d is a **Dependent Observable**".*

Task 2: Can you model a right angled triangle with sides a, b and c? The value of c should always be the hypotenuse (longest side) and should be calculated as a dependency on a and b. Hint: use the `sqrt()` function.

If a = 3 and b = 4, then c = ...



What is your definition of c? c is .....

You can create dependencies on any type of observable! (e.g. integer, real, string, list)

Try executing the following script:

```
name is "Ant";
welcome is "Hello " // name;
```

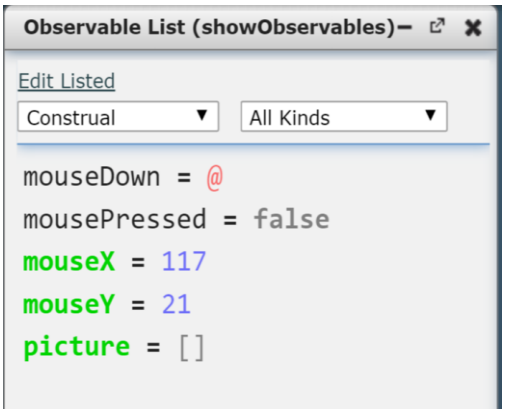
Note: the // is the operator to join strings.

Now try to change the value of name:

```
name is "James";
```

The Observable List

The Observable List is the window on the left hand side. When you create a new base observable or dependent observable you will see its name and current valuein the Observable List.



You saw the observables a, b and c being created via the Input Tab. Note that the Observable List shows the current value of c (not the dependency definition). Dependent observables are shown in `green` to differentiate them from base observables.

Redefining observables and dependencies

To change an observable or a dependency, you can edit its definition in the Input Tab and execute it by left clicking in the gutter. You can also 'live edit' the definition of an observable by holding the left mouse button down for two seconds in the gutter until a star appears. If the value of an observable is a number, you can then adjust its value by hovering the mouse over it and sliding the mouse to the right and to the left.

You can change observables and dependencies at any time!

The Canvas Picture

The Canvas Picture is currently a large empty white space on the right hand side. Using your basic knowledge of observables/dependencies, you can now add some simple graphics.

Creating a line

```
lineA is Line(0, 0, 50, 100);
picture is [lineA];
```

The first definition creates a line from point (0,0) to point (50,100). To draw a line from (x1,y1) to (x2,y2) then you can create a definition for Line(x1, y1, x2, y2).

The second definition places the line inside the `picture`. (Without this, you would not see the line in the Canvas Picture.)

Note that (0,0) is the top left corner of the Canvas Picture.

You can modify the line:

```
lineA is Line(100, 50, 100+10*a, 50);
```

Now the line is dependent on the value of the observable ‘a’.

Task 3: What happens when you change 'a'?

Answer .....

Create a new line called lineB and add it to the picture:

```
lineB is Line(100, 50, 100, 50+10*b);
picture is [lineA, lineB];
```

Task 4: Create lineC to make a right-angled triangle with lineA and lineB. When you change the values of a and b then the triangle should change.

Answer. `lineC is`…

Another optional parameter of Line is the colour:

```
colour is "red";
lineA is Line(100, 50, 100+10*a, 50, colour);
```

Now you can change the colour observable:

```
colour is "green";
```

Or you could make the colour observable a conditional dependency:

```
colour is "red" if a > 10 else "green";
```

Task 5: What happens when you change the value of 'a' to be more than 10?

Answer.

Conditional dependency

```
answer is true_value if condition else false_value;
```

The meaning is if "condition" is true then the answer is "true_value" else it is "false_value".

Text on the Picture

The Picture is still quite empty with only lines, so let's create some text, and then put it on the picture:

```
textA is Text("Hello folks", 50, 50, 16, "blue");
picture is [lineA, lineB, lineC, textA];
```

The 5 parameters for Text are the text, x position, y position, font size and text colour.

We can make the text dependent on another observable:

```
textA is Text(stringA, 50, 50, 16, "blue");
stringA = "I am a label";
```

Task 6: Put textA next to lineA and make stringA a dependency on the observable 'a'. If a has the value of 3 then textA should show "a = 3".

Answer. **`textA is`**…

Mouse observables

Take a look at the Observable List again and you might notice some observables that you did not create. Try clicking on the Picture and see what changes on the Observable List.

For example, "mousePressed", "mouseX" and "mouseY" are automatically provided by JS-Eden and they correspond to the mouse status and mouse pointer position in the picture. Try this:

```
myLine is Line(0, 0, mouseX, mouseY);
```

There is also an observable ("mouseDown") that records the last position of the mouse click which represents where the user presses down on the picture.

Task 7: Can you make the line turn red when the mouse button is being pressed? (Hint: Use a "conditional dependency".)

Answer. **`myLine is`**…

Shapes

There are several different shape type observables you can create.

Rectangles

To make a rectangle:

```
myRect is Rectangle(20, 20, 220, 100);
```

You will also need to add the rectangle to the picture (same as a Line). The minimum parameters for a rectangle are x, y, width and height. Rectangle has 2 optional parameters which are similar to the Circle properties: fill colour and outline colour. Example:

```
myRect is Rectangle(20, 20, 220, 100, "orange", "black");
```

Circles

To make a circle:

```
myCircle is Circle(100, 100, 50);
```

The minimum parameters for a Circle consist of x, y, radius. There are 2 optional parameters for Circle: fill colour and outline colour.

Task 8: You are given this script:

```
x is 275;
myLine is Line(x, 10, x, 360);
colour is "orange";
myCircle is Circle(mouseX, mouseY, 20, colour, "black");
picture is [myLine, myCircle];
```

Change the dependency of "colour" so that if the user moves the circle to the left side of the line then the circle is yellow and to the right then the circle is green.

Answer. **`colour is`**…

Removing things

If you want to remove a line, circle or rectangle from the picture then you can change your picture dependency. For example, to remove all:

```
picture is [];
```

If you want to start a new model then you can refresh your browser. You should probably do that before the next task!

Animation

In JS-Eden, animation can be thought of as "position dependent on time".

Create a clock

A clock consists of an observable to update (e.g. `tick`) and an update period in milliseconds (e.g. 1000):

```
tick = 0;
setedenclock(&tick, 1000);
```

After submitting, check that a new observable called `tick` is in your Observable List and it will update every 1000 milliseconds (every 1 second).

Task 9: You are given this script:

```
earthX is 100;
earth is Circle(earthX, 100, 10, "green");
picture is [earth];
tick = 0;
```

Use setedenclock() and the tick observable to animate the circle moving from left to right, by editing "earthX" only.

Answer. **`earthX is`**…

If Earth moves off the screen then bring it back with:

```
tick = 0;
```

Building a model of the solar system

Let's make Earth move in a circular path around the sun!

Continuing from Task 9, we can make a circle bounce backwards and forwards along the x-axis.

```
originX = 200;
earthDistance = 100;
earthX is originX + sin(tick) * earthDistance;
```

Next, speed up your animation by modifying the clock.

```
setedenclock(&tick, 100);
```

Task 10: Add a yellow circle called sun at (originX,originY). Change the earth dependency by introducing a dependency for earthY which causes earth to move in a circular path around the sun.

Hint: use **`cos(tick)`** ***in your dependency for earthY.***

Some facts about the solar system:

a) Earth is a distance of 1 AU from the sun.
b) Mercury, Venus and Mars are 0.4 AU, 0.7 AU and 1.5 AU from the sun respectively.
c) Earth moves once around the sun every 365 days
d) Mercury moves once around the sun every 88 days (it is ~4.15 times faster than Earth). Orbits for Venus and Mars are 225 days and 694 days respectively.

Task 11: Add Mercury, Venus and Mars with correct relative distances and speeds to Earth.

Hint: you can make a planet move 2x faster by using **`sin(tick * 2)`**

Congratulations! If you got to task 10 or 11 then you are well on your way becoming a JS-Eden modeller. Find out more about this tool by exploring the Project List from the menu.