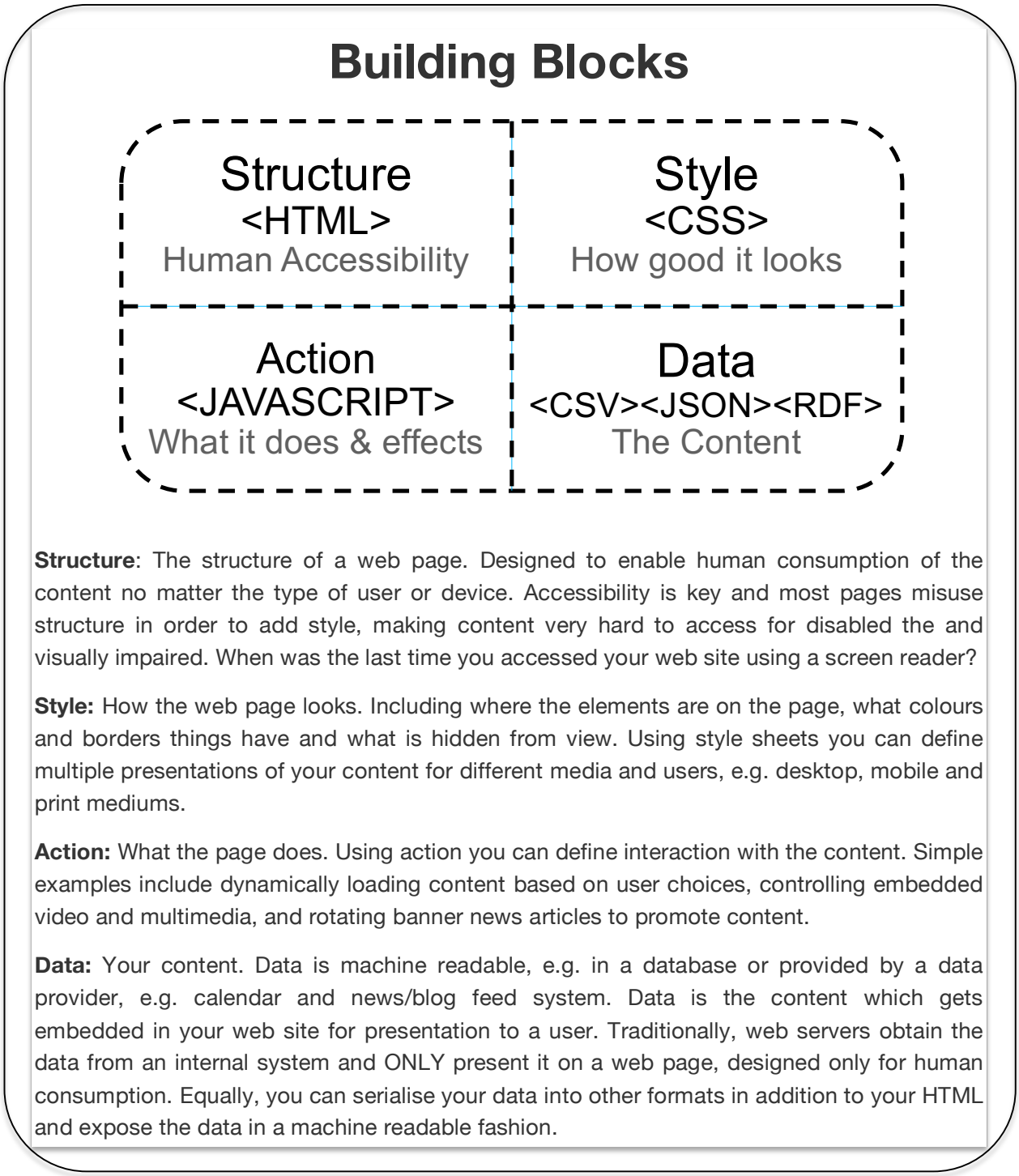
Exploring complex datasets

In this exercise we shall be using codepen.io to build an interface to explore complex data formats.

codepen.io provides a safe environment to experiment with building interactive web pages using modern HTML5 components outlined below. This exercise focusses on the **Action** block to load data and generate an advanced crossfilter visualisation.



Structure: The structure of a web page. Designed to enable human consumption of the content no matter the type of user or device. Accessibility is key and most pages misuse structure in order to add style, making content very hard to access for disabled the and visually impaired. When was the last time you accessed your web site using a screen reader?

Style: How the web page looks. Including where the elements are on the page, what colours and borders things have and what is hidden from view. Using style sheets you can define multiple presentations of your content for different media and users, e.g. desktop, mobile and print mediums.

Action: What the page does. Using action you can define interaction with the content. Simple examples include dynamically loading content based on user choices, controlling embedded video and multimedia, and rotating banner news articles to promote content.

Data: Your content. Data is machine readable, e.g. in a database or provided by a data provider, e.g. calendar and news/blog feed system. Data is the content which gets embedded in your web site for presentation to a user. Traditionally, web servers obtain the data from an internal system and ONLY present it on a web page, designed only for human consumption. Equally, you can serialise your data into other formats in addition to your HTML and expose the data in a machine readable fashion.

Exercise

The idea of this exercise is to extend beyond simple HTML5 pages and look at how Javascript can be used to generate interactive graphics from raw data

In order to complete this exercise we are going to use the codepen.io tool.

codepen.io

codepen (shown below) is an online sandbox which allows you freely to experiment with the building blocks of the web. It has three main panels that allow you to define structure, style and action related to your web page. The fourth panel (at the bottom) shows the result of your editing; this panel automatically updates as you enter your code.

In this exercise we shall be loading data dynamically and the aim is to produce a live, automatically updating, dashboard view of data on the web.

Starting point

In order to speed up this exercise a basic template has been created.

http://codepen.io/davetaz/pen/Kdberx

This template has already defined the basic HTML elements as well as core JavaScript functions that will draw a basic graph of some data.

Load this template and fork it into you own account for editing.

Note that the **D3**, **bootstrap**, **crossfilter** and **dc** libraries have already been configured and loaded. These libraries make our job of making an interactive visualising much easier.

1. The framework

Before we begin, lets look at the content of the HTML and JavaScript blocks.

```
<div id="status_pie"></div>
<div id="timeline"></div>
<table id="dc-data-table"></table>
<div id=”supplier_row”></div>
```

The HTML block defines 4 elements that are going to contain one chart (and a table) each. There is additional mark-up that helps define the layout of the page. You will also notice that the table should already have data in it.

```
// Create variables and define the charts to use from the crossfilter library
var dataTable = dc.dataTable('#dc-data-table');
var statusPie = dc.pieChart('#status_pie');
var timeline = dc.barChart('#timeline');
var supplierRow = dc.rowChart('#supplier_row');

// Load the data and create the charts
d3.csv('http://training.theodi.org/DataWorkshop/course/en/exercises/Spend_april_10_e
nriched.csv', function (data) {
    …
    …
});
```

The JavaScript block (shown above) contains a number of lines that create our chart templates, but we have yet to define what to do with any of them except the table. There are also many other lines that are comments (they start with `“//”`). Comment lines are not executed and can contain anything you like.

All other code added in this exercise goes in the function block, marked “`ALL OF YOUR CODE GOES HERE`”.

2. Setting up the cross filter

The cross filter libraries are pre loaded so we just need to load the data into them ready to manipulate and visualise.

Just below where the data is loaded you will see the following.

```
var ndx = crossfilter(data);
var all = ndx.groupAll();
```

These two variables create a cross filter called `ndx` and a cross reference variable call `all`. We will need these to draw graphs. If we draw the graphs with `data`, the cross filter would not work.

At the end of this function you will see the code below and this must remain at the end of the function at all times.

```
dc.renderAll();
```

This function call tells our charts to render on the page. So far only the table is being rendered. This may look complex but as we built the other graphics is should look less daunting.

3. Add a company status pie chart

Adding graphics involves two main steps:

1) Define the data.
2) Define the chart and load the data into it.

Add the following to your code block ("ALL YOUR CODE GOES HERE") to define the data required for the pie chart.

```
var supplierStatus = ndx.dimension(function(d) {
          return d["Company Status"];
    });

var supplierStatusGroup = supplierStatus.group();
```

Here an `ndx dimension` is a row in the table. This row is loaded into a variable called `d`. We select the column called "Company Status" from this row. The last line adds this chart to the cross filter grouping.

Add the following below the data definition to create a pie chart for the data.

```
statusPie
    .width(180)
    .height(180)
    .radius(90)
    .dimension(supplierStatus)
    .group(supplierStatusGroup)
    .ordinalColors(['blue','red','orange','gray'])
    .label(function(d) {
        return(d.key);
    });
```

Feel free to change the height, width, radius and colours of the pie chart to suit your own preferences.

Once done you should have rendered a pie chart on your page which shows how many companies are Active or otherwise.

This is a simple visualisation, but not a great cross filter as we only have one chart. So let's plot another.

4. Adding a second chart

In this section we are going to add a second chart to our cross filter. In particular we are going to add a timeline showing how many transactions there are per day.

Add the following below your pie chart code to load the timeline data.

```
var date = ndx.dimension(function(d) {
          return d.dd;
    });
var dateGroup = date.group();
```

This is almost exactly the same as before when define a dimension and group. The only difference being that the data we are using is a date object that has been created in the code template already. Feel free to find the function that handles the dates. There are two more variables it uses to define the chart, see if you can spot them as we proceed.

Add the following below your data load to add the chart to your cross filter.

```
timeline
     .width(500)
     .height(220)
     .dimension(date)
     .group(dateGroup)
     .centerBar(false)
     .elasticX(false)
     .gap(0)
     .xUnits(function(){return 50;})
     .x(d3.time.scale().domain([mindate,maxdate]))
     .renderHorizontalGridLines(true);
```

With this chart rendered you now have your first cross filter. To use it select an option from any chart to filter all other charts.

With the line chart you can also change axis properties. Try changing the elasticX property to **true** and see what the difference is when you select a company status in the pie chart.

5. Adding a third chart (less guidance)

The last chart to add will look at either the “Expense Area” or “Supplier Name” columns and you can choose.

First create the data dimension following the same method as in (3). Make sure you select and call your variable by the right name.

With the data created now add the new chart.

```
flexibleRow
    .width(480)
    .height(800)
    .dimension(flexible)
    .group(flexibleGroup)
    .label(function(d) {
      return d.key;
    })
    .title(function(d) {
      return d.value;
    })
    .elasticX(true)
    .ordering(function(d) {
      return -d.value;
    })
    .xAxis().ticks(6);
```

The completed example is available from the following URL:

http://codepen.io/davetaz/pen/xvyLrx

6. Self exploration

Try adding a new type of chart? The documentation for the crossfilter library can be found at the following location:

https://github.com/dc-js/dc.js/blob/master/web/docs/api-1.6.0.md