Reusable modules

VISUALIZATION ON THE WEB

From Javascript code to Modules

- D3.js provides a vast library of examples
- In many projects, an example is modified and adapted for a specific use
- However, the code is difficult to maintain and adapt to different scenarios
- Solution: encapsulate all the code within a module that is bound to data and a container

Javascript and Objects

- We want to organize our visualization into components for
  - Modularity: separate the different parts of a complicated visualization
  - Composability and reusability: reuse smaller pieces in different visualization
  - Simplification: concentrate on smaller part of the main problem first
- To implement this approach we use objects, i.e. entities with properties and functions
- Objects are not fully supported in Javascript (prior to ES2016)
  - We exploit function closuers

An example for Barchart

```
// Creates bar chart component and configures its margins
barChart = chart()
    .margin({top: 5, left: 10});

container = d3.select('.chart-container');

// Calls bar chart with the data-fed selector
container.datum(dataset).call(barChart);
```

General schema for a chart

```
function chart() {
  var width = 720, // default width
      height = 80; // default height

  function my(selection) {
    // generate chart here, using `width` and `height`
  }

  my.width = function(value) {
    if (!arguments.length) return width;
    width = value;
    return my;
  };

  my.height = function(value) {
    if (!arguments.length) return height;
    height = value;
    return my;
  };

  return my;
}
```

Internal properties of the object: width and height

Constructor and preparation for the chart attached to the selection

Getter and setter for width

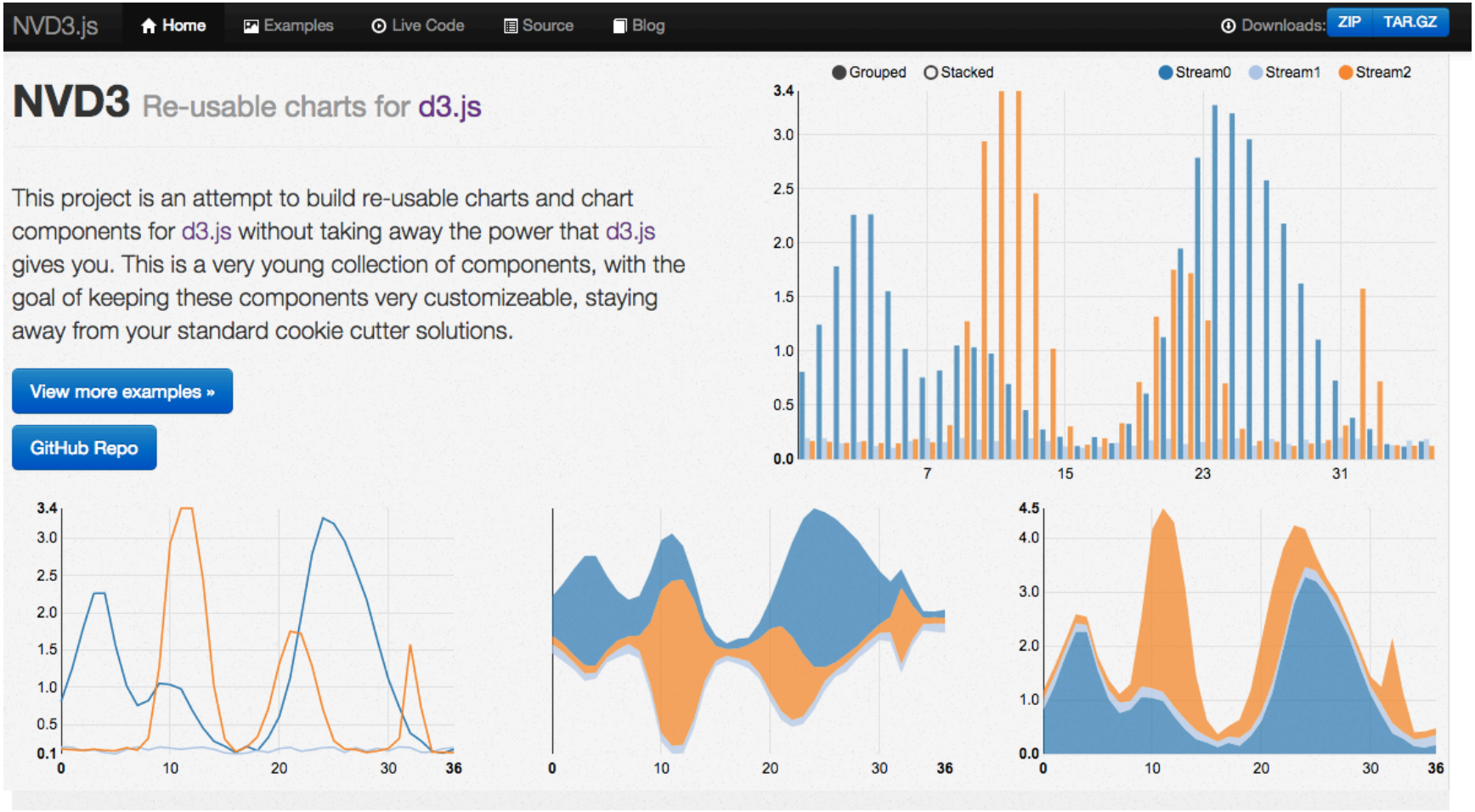
Getter and setter for height

Export the internal function outside this scope

Line chart to a reusable component

- Specification
  - Input: the component takes in input an array of numbers
  - Visualization: each number is rendered as a line proportional to its value; an axis provide reference for the values

NVD3.js


NVD3.js
Home
Examples
Live Code
Source
Blog
Downloads:
ZIP
TAR.GZ
NVD3 Re-usable charts for d3.js
This project is an attempt to build re-usable charts and chart components for d3.js without taking away the power that d3.js gives you. This is a very young collection of components, with the goal of keeping these components very customizeable, staying away from your standard cookie cutter solutions.
View more examples »
GitHub Repo
Grouped
Stacked
Stream0
Stream1
Stream2

NVD3

- An high level library built on top of D3js
- Provides reusable charts
  - A large library of components
  - Manage annotations and interaction support
  - Extensible with new plugins

NVD3 – Getting Started

- Install NVD3 via NPM
  - npm install nvd3 –save
- Include the library files within HTML page:
  - CSS and JS
  - Important: d3js should be imported before nvd3

NVD3 – Skeletal HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>My First Chart</title>
        <link href="nv.d3.css" rel="stylesheet" type="text/css">
        <script src="d3.v3.js"></script>
        <script src="nv.d3.js"></script>
    </head>
    <body>
        <svg style='height:600px'/>

        <script type="text/javascript">
          // your code here
         </script>
    </body>
</html>
```

NVD3 – Chart Initialization code

```
nv.addGraph(function() {                //This adds the chart to a global rendering queue.
    var chart = nv.models.lineChart();  //Create instance of nvd3 lineChart

    chart.xAxis
        .axisLabel("X-axis Label");     //Set X-axis attributes

    chart.yAxis
        .axisLabel("Y-axis Label")      //Set Y-Axis attributes.
        .tickFormat(d3.format("d"))     //Set Y-Axis label formatting.
        ;

    d3.select("svg")                    //Select the document's <svg> element
        .datum(myData())                //Attach data to the <svg> element.
        .call(chart);                   //Passthe d3.selection to our lineChart.

    nv.utils.windowResize(              //Updates the window resize event callback.
            function() {
                chart.update();         //Renders the chart when window is resized.
            }
        );

    return chart;   //Must return the enclosed chart variable so the global rendering queue can store
it.
});
```

NVD3 – Data format

```
[
    {
        key: "<Series name>",
        color: "<CSS color>",
        values: [
            {x: 0, y: 10},
            {x: 1, y: 20},
            {x: 2, y: 30}
            ....
        ]
    },
    {
        key: "<Series name>"
        ...
    }
]
```

Exercise – Colors of Art