NEW YORK UNIVERSITY

Data Visualization on the Web with D3

Bowen Yu

April 11, 16

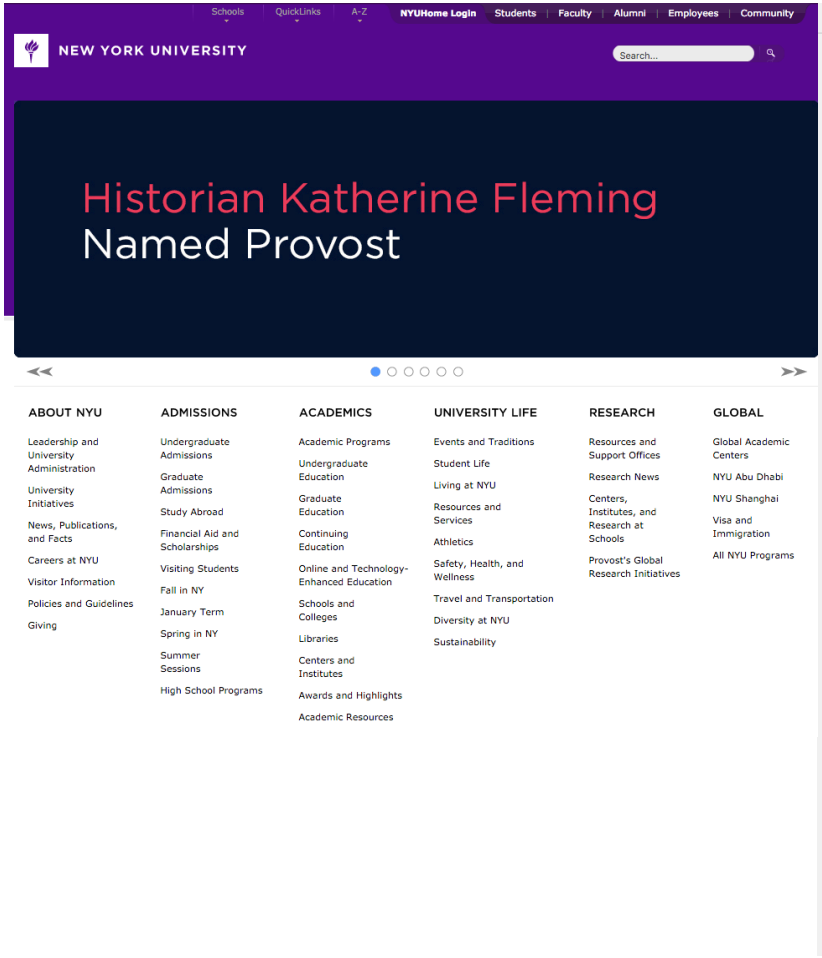
Big Data Analysis

Interactive Analysis

- After data processing with BD techniques, it is necessary to visualize the data so that human analyst can be involved for decision making.

- Today we will go through the necessary tools for creating an interactive web visualization.

HTML

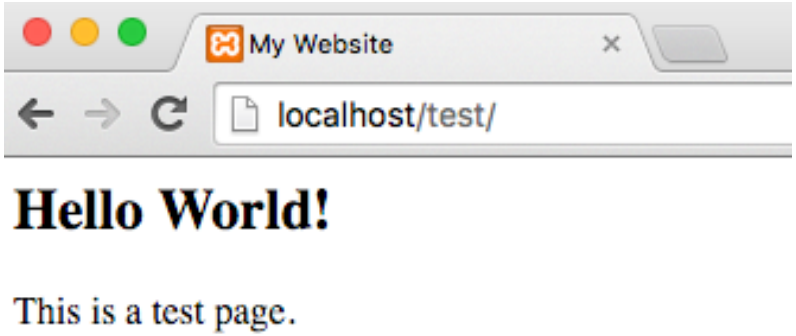



```
html head
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3
3  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4  <head>
5
6
7      <meta name="CMS" content="CQ" />
8      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9      <meta http-equiv="X-UA-Compatible" content="IE=Edge" />
10     <title>New York University</title>
11     <meta http-equiv="keywords" content="">
12
13     <meta name="author" content="NYU Web Communications" />
14     <link rel="icon" href="/common/images/favicon.ico" type="image/x-icon"/>
15
16     <!-- STYLE SHEETS -->
17     <link rel="stylesheet" href="/common/css/base.css" type="text/css" media=
18     <link rel="stylesheet" href="/common/css/homePage.css" type="text/css" me
19     <link rel="stylesheet" href="/common/css/channels.css" type="text/css" me
20     <link rel="stylesheet" href="/common/pe-icon-social/css/pe-icon-social.cs
21     <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/font-awesome/4.2.0
22     <link rel="stylesheet" href="/common/css/print.css" type="text/css" media
23
24     <style type="text/css">@media screen,projection{.cufon-canvas{display:inl
25
26     <!-- PHOTO GALLERY -->
27     <link rel="stylesheet" href="/common/css/jquery.lightbox-0.5.css" type="t
28
29     <!-- SCRIPTS -->
30     <script type="text/javascript" src="/etc/designs/nyu/clientlibs/jquery.js
31
32     <script type="text/javascript" src="/common/js/utils.js"></script>
33
34     <!-- LIGHT BOX -->
35     <script type="text/javascript" src="/common/js/jquery.lightbox-0.5.js"></
36
37     <!-- CUFON -->
```

HTML

- HyperText Markup Language (HTML)
- Tags and content: <{tag}>{content}</{tag}>
  - Layout tags
    - **<p>** paragraph
    - ** ** division
    - **<h2>** header
  - Style tags
    - **<b>** bold text
    - **<i>** italic text
  - Content tags
    - **<img>** images
    - **<iframe>** embedded form/video

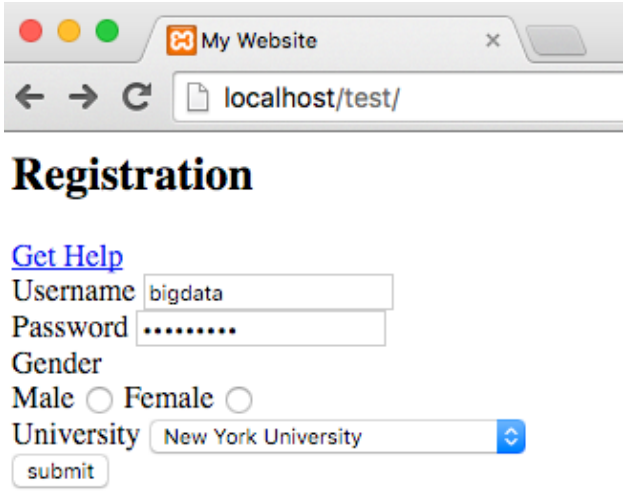
A Simple Page




```
1   <html>
2   <head>
3     <title>My Website</title>
4   </head>
5   <body>
6   <h2>Hello World!</h2>
7   <p>This is a test page.</p>
8   </body>
9   </html>
10
```

index.html

Web Forms

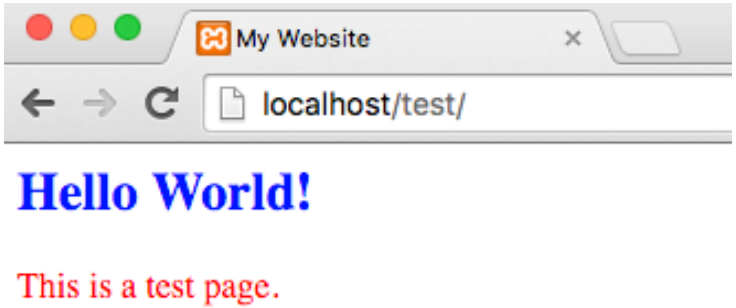



```
7  <h2>Registration</h2>
8  <form action="register.php">
9    <div>
10     <a href="help.html" target="_blank">Get Help</a>
11   </div>
12   <div>
13     <label>Username</label>
14     <input name="username" type="text">
15   </div>
16   <div>
17     <label>Password</label>
18     <input name="password" type="password">
19   </div>
20   <div>
21     <label>Gender</label>
22     <div>
23       <label>Male</label>
24       <input type="radio" name="radio-gender" value="male">
25       <span>Female</span>
26       <input type="radio" name="radio-gender" value="female">
27     </div>
28   </div>
29   <div>
30     <label>University</label>
31     <select>
32       <option value="nyu">New York University</option>
33       <option value="uw">University of Washington</option>
34       <option value="ucb">University of California Berkeley</option>
35     </select>
36   </div>
37   <input type="submit" value="submit">
38 </form>
```

Styling Your HTML

- A plaintext page is boring.
- Use CSS (cascading style sheet) to make a page appear nicer.

CSS Stylesheet




```css
p {
  color: red;
}

h2 {
  color: blue;
}
```

```html
<html>
<head>
  <title>My Website</title>
  <link rel=stylesheet type="text/css" href="style.css"/>
</head>
<body>
<h2>Hello World!</h2>
<p>This is a test page.</p>
</body>
</html>
```

List of Common CSS Styles

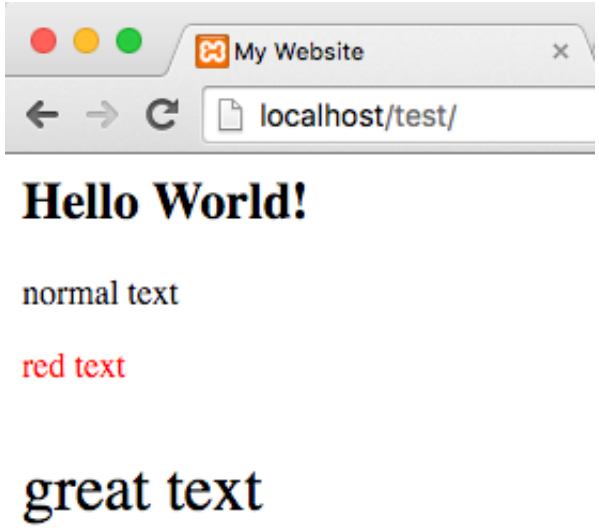
- Text:
  - color, font-size, font-family, font-weight
  - word-wrap
- Rendering Style:
  - border
  - background
- Layout:
  - width, height, line-height
  - left, right, bottom, top
  - margin, padding
  - position, float
  - z-index
- Interaction:
  - Pointer-events

- Search for details! e.g. http://www.w3schools.com/css/default.asp

CSS Syntax

- Selection + Style Settings
- Selection: CSS expression that selects a group of DOM elements
- Selections are typically on tags, classes, ids, DOM order.
  - Tags: body, div, p …
  - Classes: elements of a same category – your design
  - Id: unique identifier for element – your design

CSS Examples




```
.red {
  color: red;
}

#great {
  font-size: 30px;
}
```

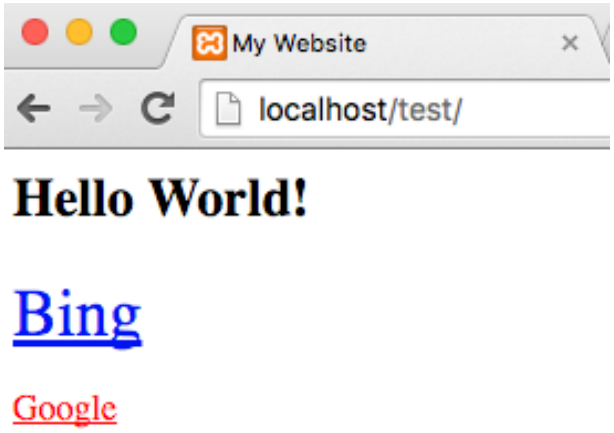
- Class selection begins with a dot.
- Id selection begins with a hash.

```
<h2>Hello World!</h2>
<p>normal text</p>
<p class="red">red text</p>
<p id="great">great text</p>
```

Working with HTML Hierarchy

{parent selection} {child selection}

Add a space between selections to define containing relations

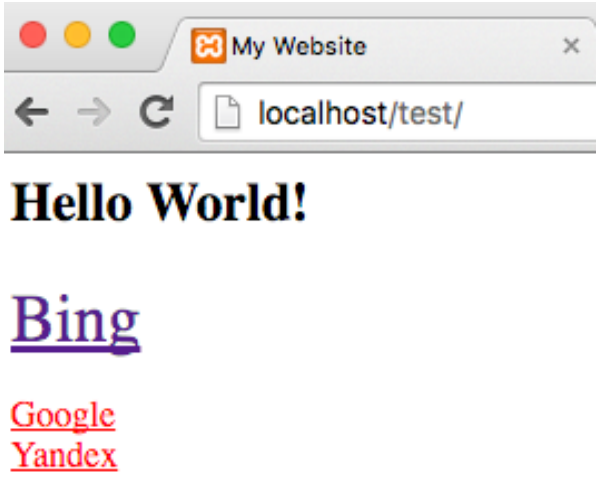



```
1    p a {
2      font-size: 30px;
3    }
4
5    div a {
6      color: red;
7    }
```

Working with HTML Hierarchy

{parent selection} {child selection}

Add a **space** between selections to define containing relations




```css
p a {
  font-size: 30px;
}

div a {
  color: red;
}
```

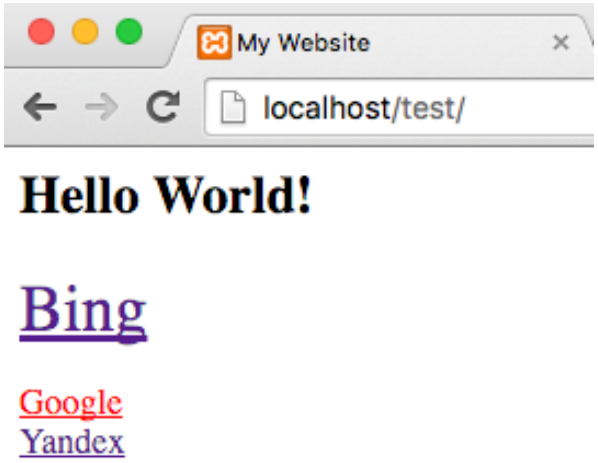
```html
<p>
  <a href="https://bing.com/">Bing</a>
</p>
<div>
  <a href="https://google.com/">Google</a>
  <div>
    <span>
      <a href="https://yandex.com/">Yandex</a>
    </span>
  </div>
</div>
```

Note that the using space the selected child does not need to be *immediate*.

Working with HTML Hierarchy

{parent selection} > {child selection}

Add a **larger sign** between selections to define *immediate* containing relations




```css
p a {
  font-size: 30px;
}

div > a {
  color: red;
}
```

```html
<p>
  <a href="https://bing.com/">Bing</a>
</p>
<div>
  <a href="https://google.com/">Google</a>
  <div>
    <span>
      <a href="https://yandex.com/">Yandex</a>
    </span>
  </div>
</div>
```

Compound Selection

- You may select the DOMs that satisfy multiple criteria.

All paragraphs

The paragraphs with class "highlighted"

The paragraph with id "focused" (typically unique)

Paragraphs and links (OR condition with comma)

```
1   p {
2     font-size: 20px;
3   }
4
5   p.highlighted {
6     color: red;
7   }
8
9   p#focused {
10    font-weight: bold;
11  }
12
13  p, a {
14    font-family: Helvetica, Arial, sans-serif;
15  }
```

Often … CSS becomes nasty when you have complicated hierarchy ☹

```
.visflow .navbar-nav > li > a,
.visflow .navbar-brand {
  font-size: 1em;
  height: 28px;
  padding-bottom: 0;
  padding-top: 4px; }

.visflow .navbar {
  min-height: 28px;
  z-index: 0; }
  .visflow .navbar .dropdown-menu > li > a > i {
    margin-right: 10px; }

.visflow .navbar-fixed-top a:hover {
  cursor: pointer; }

.visflow .bold {
  font-weight: bold; }

.visflow .tooltip {
  z-index: 1001; }

.select2-container .select2-selection--single {
  height: inherit; }
  .select2-container .select2-selection--single .select2-selection__rendered {
    box-shadow: inset 0 1px 1px rgba(0, 0, 0, 0.075);
    line-height: 1.42857143;
    padding-bottom: 6px;
    padding-top: 6px; }

.select2-container.select2-container--default .select2-selection--single .select2-selection__arrow {
  color: #000;
  height: 100%;
  right: 7px; }
```

Augmented CSS: SASS

- http://sass-lang.com/

Now instead of …

```
1  .my-div p {
2    font-size: 1em;
3  }
4
5  .my-div a {
6    color: aqua;
7  }
8
9  .my-div a.highlighted {
10   color: red;
11 }
```

style.css

You can write sassy nested code. ☺

```
1  .my-div {
2    p {
3      font-size: 1em;
4    }
5
6    a {
7      color: aqua;
8
9      &.highlighted {
10       color: red
11     }
12   }
13 }
```

style.scss

Using SASS

- Note that you cannot directly include ".scss" files in the HTML as HTML only recognizes standard CSS.
- You must run "sass" tool to compile SCSS into CSS and then load the generated CSS in your HTML.
- There are other cool features such as functions (mixins) in SASS. More details can be found at the SASS website.
- Alternative: LESS (http://lesscss.org/)

Use Well-Designed Templates

http://getbootstrap.com/

Examples

Build on the basic template above with Bootstrap's many components. We encourage you to customize and adapt Bootstrap to suit your individual project's needs.

Get the source code for every example below by downloading the Bootstrap repository. Examples can be found in the `docs/examples/` directory.

Using the framework

Starter template

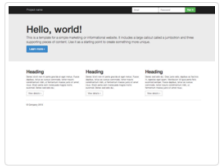
Nothing but the basics: compiled CSS and JavaScript along with a container.

Bootstrap theme

Load the optional Bootstrap theme for a visually enhanced experience.

Grids

Multiple examples of grid layouts with all four tiers, nesting, and more.



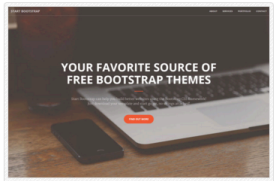
Jumbotron

Build around the jumbotron with a navbar and some basic grid columns.

Narrow jumbotron

Build a more custom page by narrowing the default container and jumbotron.

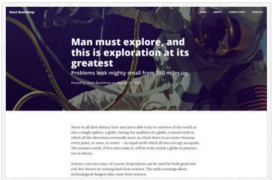
http://startbootstrap.com/



Creative

A one page creative theme.

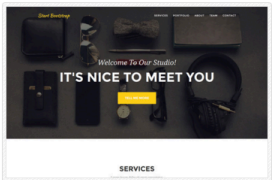
Preview & Download



Clean Blog

A clean blog theme.

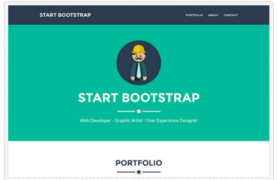
Preview & Download



Agency

A one page agency theme.

Preview & Download



Freelancer

A one page freelancer theme.

Preview & Download

Grayscale

A multipurpose one page theme.

Preview & Download

SB Admin 2

A free Bootstrap admin theme.

Preview & Download

Dynamic Pages

- HTML + CSS are good for creating static nice pages.
- However, static pages most of the time cannot meet the needs of web usage today.
- Consider the scenario that you want to access your personal bank account summary:
  - The bank must determine who you are and then serve the page properly.
- Obviously, some scripting is needed to achieve dynamic pages.

PHP Solution to Dynamic Pages

- **PHP:** a general-purpose scripting language running at the server side.
  - The PHP script is inaccessible to the client. The client may only access the PHP output.
  - When the client requests a PHP page, the PHP script determines the identity of the client (by cookies, login credentials, etc.) and then return a dynamically generated page.
  - Loose typing. Line by line execution.
  - PHP provide good security. However, it is hard to provide page interactions at the client side.

JavaScript Solution to Dynamic Pages

- **JavaScript (JS):** Programming language that runs in a client's browser.
  - Manipulate DOM elements directly in the browser.
  - Perform requests to the server and update the page without a refresh.
  - Loose typing. Line by line execution.
  - Not much related to Java ☺
- Today we will be focusing on using JS to render visualizations in the web.

Run Your JS

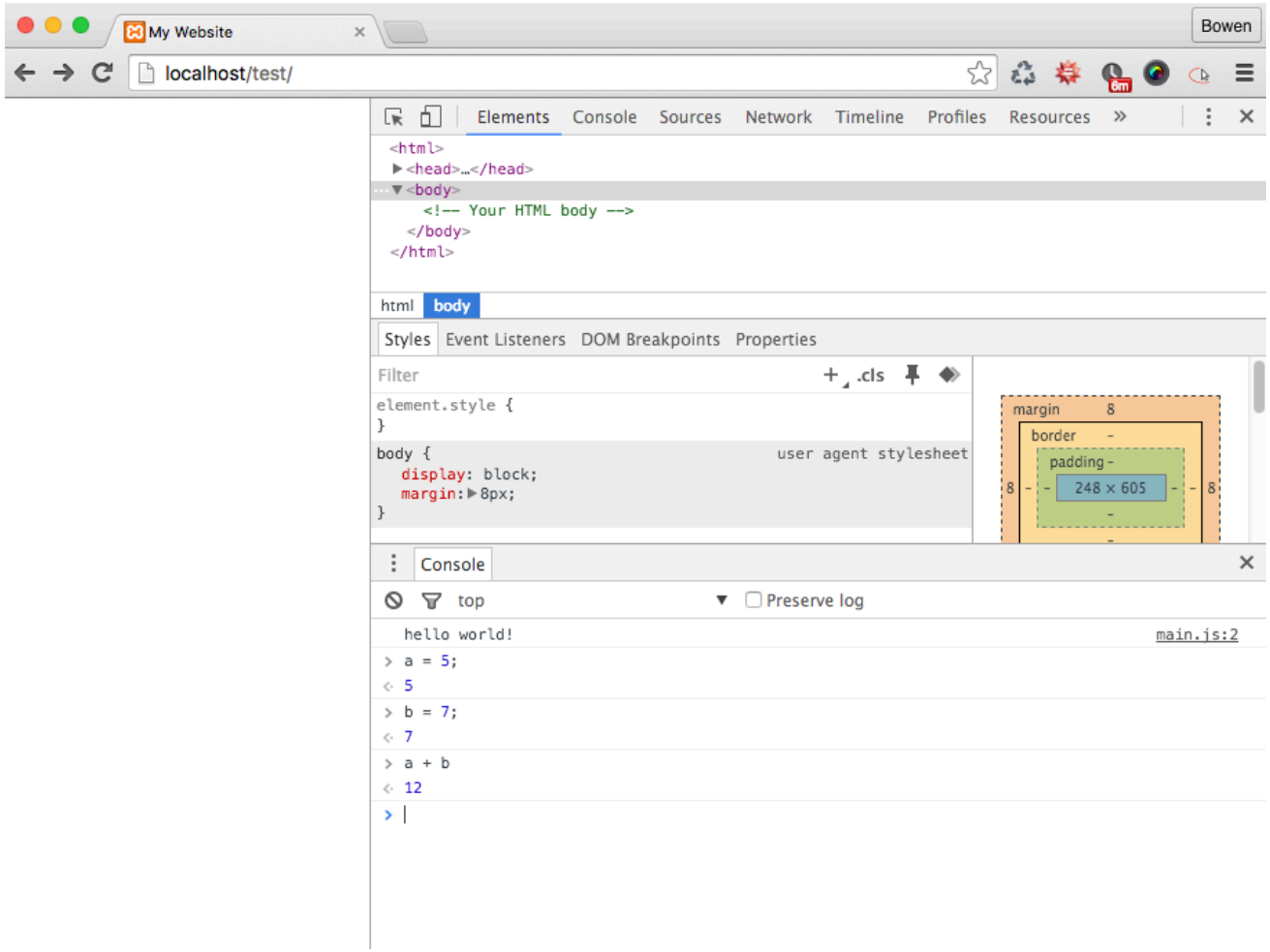
- Include the script file in your HTML (head or body)

```html
<html>
<head>
  <title>My Website</title>
  <script type="text/javascript" src="main.js"></script>
  <link rel=stylesheet type="text/css" href="style.css"/>
</head>
<body>
  <!-- Your HTML body -->
</body>
</html>
```

```
console.log('hello world!');
```

Test and Debug Your JS

- Use your browser's console



JS 101 – Number and Strings

```
1   var x = 5, y = 7;
2
3   console.log(x + y); // 12
4
5   var s1 = 'abc', s2 = '123';
6
7   console.log(s1 + s2); // 'abc123'
8
9   console.log(s2 + x); // '1235'
10
11  console.log(s2 - x); // 118
```

JS 101 - Arrays and Objects

```
1
2   var myList = [];
3
4   myList.push(1);
5   myList.push(2);
6   myList.push('abc');
7
8   console.log(myList); // [1, 2, 'abc']
9
10  var myObj = {};
11
12  myObj[1] = 'one';
13  myObj['2'] = 'two';
14  myObj['abc'] = myList;
15
16  console.log(myObj); // Object {1: "one", 2: "two", abc: Array[3]}
```

JS 101 - Loops

```
1
2   for (var i = 0; i < 10; i++) {
3     console.log(i); // 0, 1, 2, ...
4   }
5
6   var counter = 0;
7   while (counter < 10) {
8     console.log(counter); // 0, 1, 2, ...
9     counter++;
10  }
11
12  var list = [1, 2, 3];
13  for (var index in list) { // NOT RECOMMENDED!
14    console.log(list[index]); // 1, 2, 3
15  }
16  for (var index = 0; index < list.length; index++) {} // RECOMMENDED
17
18  var obj = {'a': 1, 'b': 2};
19  for (var key in obj) {
20    console.log(obj[key]); // 1, 2
21  }
```

JS 101 - Functions

```
1  function add(a, b) {
2    return a + b;
3  }
4
5  var multiply = function(a, b) {
6    return a * b;
7  };
8
9  console.log(add(1, 2)); // 3
10 console.log(multiply(3, 4)); // 12
11
12 var compute = function(a, b, operator) {
13   return operator(a, b);
14 };
15
16 console.log(compute(1, 2, add)); // 3
17 console.log(compute(3, 4, multiply)); // 12
18
19 var list = ['a', 'b', 'c'];
20 list.forEach(function(value, index) {
21   console.log(value, index); // 'a' 0, 'b' 1, 'c' 2
22 });
```

JS 101 – Nested Types

```
1   var student = {
2     name: 'Brian',
3
4     id: 'N12345678',
5
6     address: {
7       street: undefined,
8       city: 'Brooklyn',
9       apt: 'A2',
10      zip: '10000'
11    },
12
13    scores: [97, 95, 88],
14
15    averageScore: function() {
16      var sum = 0;
17      for (var i = 0; i < this.scores.length; i++) {
18        sum += this.scores[i];
19      }
20      return sum / this.scores.length;
21    }
22  };
```

JS 101 - Class

```
1  var Student = function(name, id, address, scores) {
2    this.name = name;
3    this.id = id;
4    this.address = address; // object reference!
5    this.scores = scores; // array reference!
6  };
7
8  Student.prototype.averageScore = function() {
9    var sum = 0;
10   for (var i = 0; i < this.scores.length; i++) {
11     sum += this.scores[i];
12   }
13   return sum / this.scores.length;
14 };
15
16 var brian = Student('Brain', 'N12345678', {
17   street: undefined,
18   city: 'Brooklyn',
19   apt: 'A2',
20   zip: '10000'
21 }, [97, 95, 88]);
```

Go Deeper in JS

- Search for JS documentation.
- Play in your browser's console (seeing is believing).

- **Challenge:** How to get numerical value **123** in JS in one command with no more than 50 characters, if you can only use 4 types of characters: `[ ] ! +`
- **Hint:** `+ ! ! [ ]` gives you the number 1

Manipulate DOMs with JS

```
<div class="highlighted">a</div>
<div id="focused">b</div>
```

- Old-school methods

```
1   var highlighted = document.getElementsByClassName('highlighted');
2   // [ <div class="highlighted">a</div> ]
3
4   var focused = document.getElementById('focused');
5   // <div id="focused">b</div>
```

- Returned values are the selected DOM *Element(s).* (not HTML strings)

Manipulate DOMs with JS

- Modern practice: use the jQuery library (http://jquery.com/)

```
1   <html>
2   <head>
3     <title>My Website</title>
4     <script src="https://code.jquery.com/jquery-1.12.0.min.js"></script>
5     <script type="text/javascript" src="main.js"></script>
6     <link rel=stylesheet type="text/css" href="style.css"/>
7   </head>
8   <body>
9     <div class="highlighted">a</div>
10    <div id="focused"></div>
11  </body>
12  </html>
```

- Include jQuery BEFORE your code.

jQuery Selection

- Use the jQuery namespace $

```
1    var highlighted = $('.highlighted');
2    // <div class="highlighted">a</div>
3
4    var focused = $('#focused');
5    // <div id="focused">b</div>
```

- Note that the returned values are jQuery selections (compound objects, not strings, not DOM element).
- You can select anything with CSS selection rules.

Manipulate the jQuery Selection

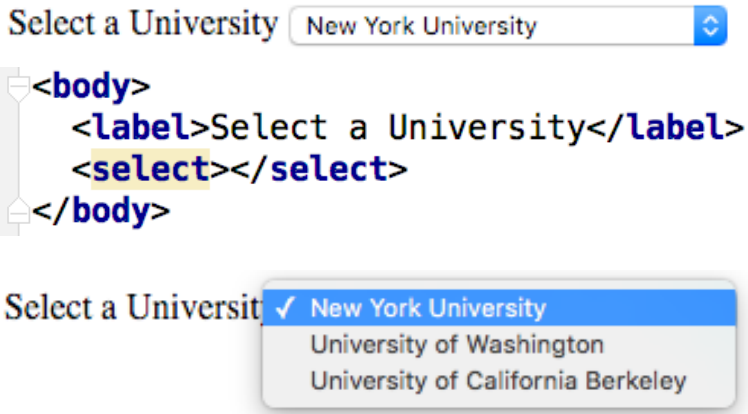
```
1
2   // Make all paragraph text appear red
3   $('p').css('color', 'red');
4
5   // Change the value (entered text) of an input box
6   $('input').val('please enter the value');
7
8   // Hide the selected div
9   $('div.canceled').hide();
10
11  // Clone the div (create a copy) and append to the cloned list.
12  $('div.to-be-cloned').clone().appendTo('.cloned-list');
```

Listen for Interaction


```
<input id="value-a" value=""> +
<input id="value-b" value=""> =
<span id="answer">?</span>
<button id="add">Add!</button>
```


```
1  function addListeners() {
2    $('#add').on('click', function() {
3      var a = +$('#value-a').val();
4      var b = +$('#value-b').val();
5      $('#answer').text(a + b);
6    });
7  }
8
9  $(document).ready(function() {
10   // Must manipulate DOMs AFTER all DOMs are loaded!
11   addListeners();
12 });
```

Generate Selection List from Data




```
var univs = [
  {
    id: 'NYU',
    name: 'New York University'
  },
  {
    id: 'UW',
    name: 'University of Washington'
  },
  {
    id: 'UCB',
    name: 'University of California Berkeley'
  }
];

$(document).ready(function() {
  var select = $('select');
  for (var i = 0; i < univs.length; i++) {
    var univ = univs[i];
    $('<option></option>')
      .val(univ.id)
      .text(univ.name)
      .appendTo(select);
  }
});
```

AJAX Request

- jQuery provides handy AJAX request wrapper so that you can perform queries without refreshing the page.

```
$.get('get-data.php', {
  dataId: 'car',
  queryParams: {
    mpg: [15, 20]
  }
}).done(function(data) {
  // success handler
}).fail(function(res) {
  // server error response, connection error, etc.
}).always(function() {
  // always perform the always() operations
});
```

Visualization with jQuery

- Theoretically you can use jQuery to manipulate any element on the web page. Therefore you may for every data object create a corresponding visualized data point.

```
var data = [1, 7, 5, 2, 3];

for (var i = 0; i < data; i++) {
  $('<div></div>')
    .addClass('bar')
    .css({
      left: i * 10,
      height: data[i] * 10,
      width: 10
    })
    .appendTo('#canvas-div');
}
```

- However it is non-trivial to perform updates, since elements can be added, removed, changed…

D3.js

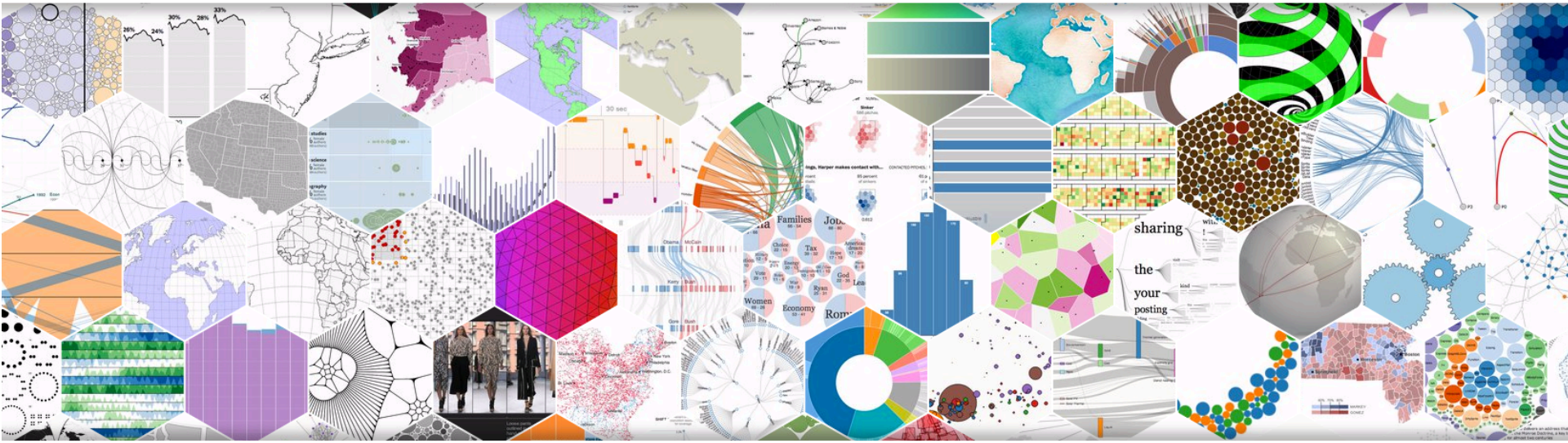
- Data-Driven Documents https://d3js.org/
- **Core concept:** Mapping data to its visual representations.
- D3 provides other handy utilities as well, e.g.
  - Visualization layout: treemap, force-directed graph, histogram
  - Axis with ticks
  - Scales, color mapping
  - Transition

D3 Site

Overview Examples Documentation Source

Fork me on GitHub

Data-Driven Documents



D3.js is a JavaScript library for manipulating documents based on data. **D3** helps you bring data to life using HTML, SVG, and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

See more examples.

Download the latest version (3.5.16) here:

- d3.zip

Or, to link directly to the latest release, copy this snippet:

```
<script src="https://d3js.org/d3.v3.min.js" charset="utf-8"></script>
```

D3 Usage

- D3 works with SVG (scalable vector graphics), which does not suffer from lower resolution when resized.

```
<html>
<head>
  <title>My Website</title>
  <script src="https://code.jquery.com/jquery-1.12.0.min.js"></script>
  <script src="https://d3js.org/d3.v3.min.js" charset="utf-8"></script>
  <script type="text/javascript" src="main.js"></script>
  <link rel=stylesheet type="text/css" href="style.css"/>
</head>
<body>
  <!-- D3 renders SVG -->
  <svg></svg>
</body>
</html>
```

D3 Selection

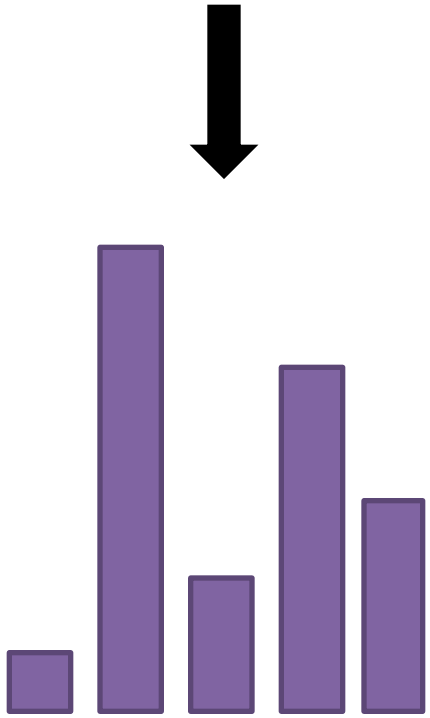
- Pretty much the same as jQuery

```
1   // Select a single rect (first if many exist)
2   var rect = d3.select('rect');
3
4   // Select all rects
5   var rects = d3.selectAll('rect');
6
7   // Set rect(s) width to 10
8   rect.attr('width', 10);
9   rects.attr('width', 10);
10
11  // set rect(s) stroke (border) color to red
12  rect.style('stroke', 'red');
13  rects.style('stroke', 'red');
```

- Note that *rect* and *rects* are D3 selections, not jQuery selections, not DOM elements.

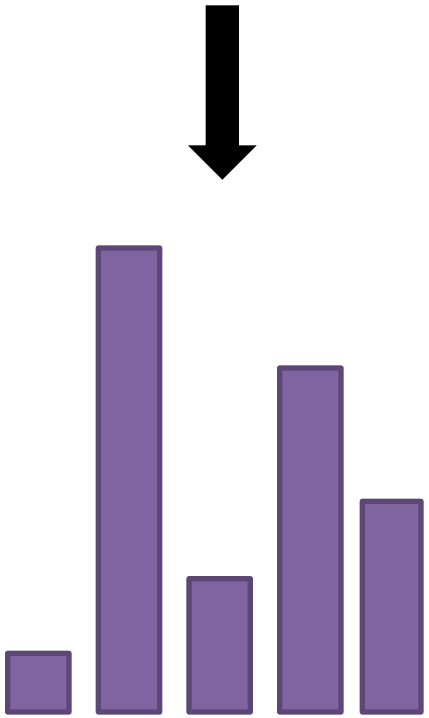
Binding Data

```
var data = [1, 7, 2, 5, 3];
```

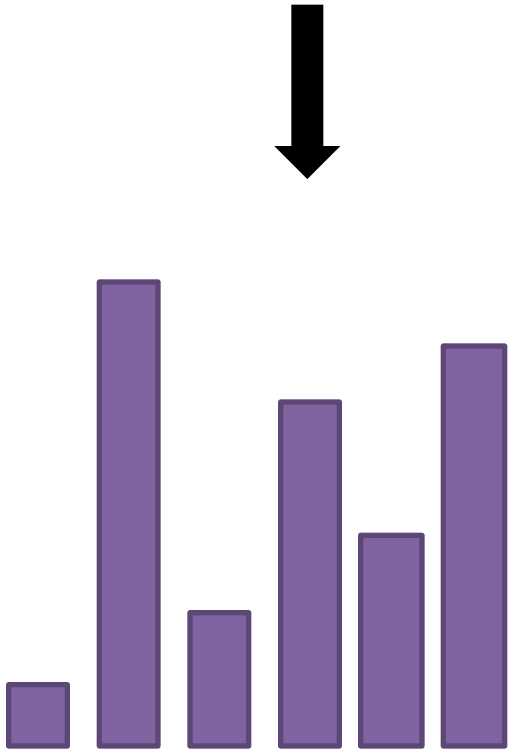


Binding Data (Add)

```
var data = [1, 7, 2, 5, 3];
```

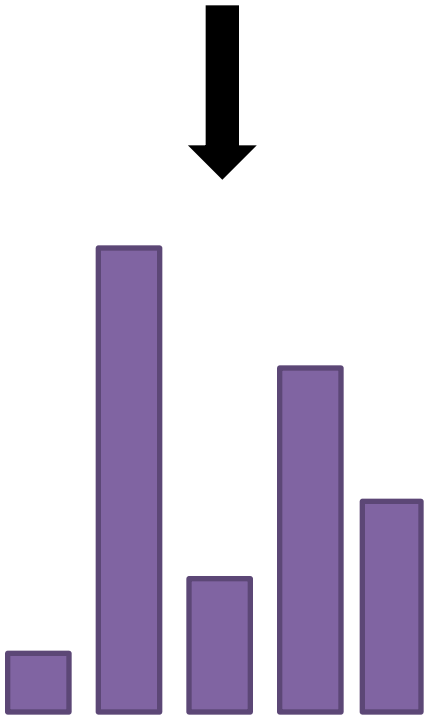


```
var data = [1, 7, 2, 5, 3, 6];
```

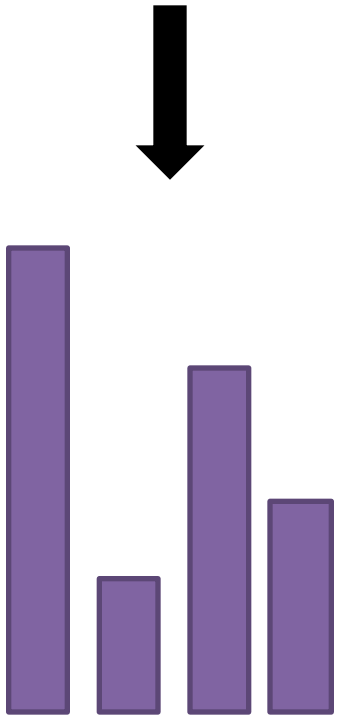


Binding Data (Remove)

```
var data = [1, 7, 2, 5, 3];
```

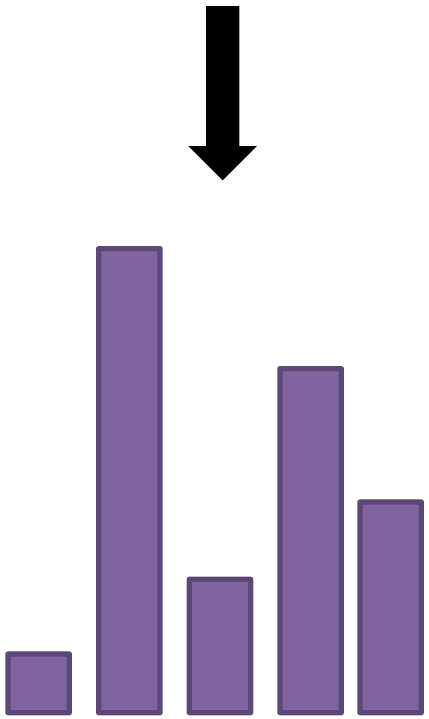


```
var data = [7, 2, 5, 3];
```

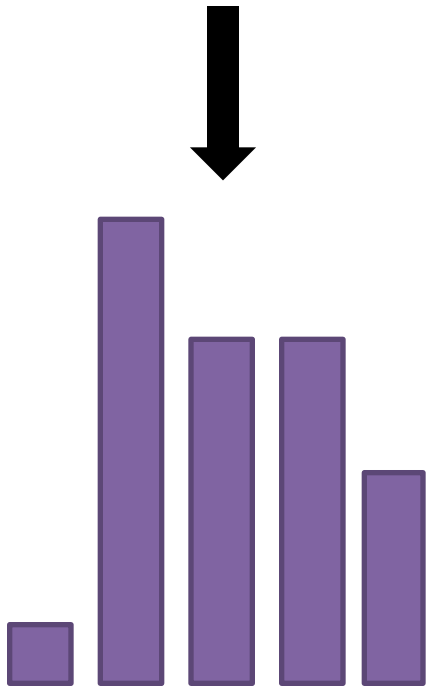


Binding Data (Modify)

```
var data = [1, 7, 2, 5, 3];
```



```
var data = [1, 7, 5, 5, 3];
```



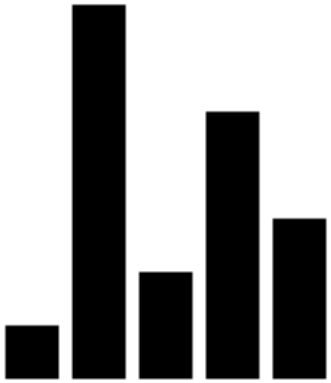
D3 Data Mapping

```
1   var svg = d3.select('svg');
2   var values = [1, 7, 2, 5, 3];
3
4   var rects = svg.selectAll('rect').data(values);
5
6   // Rectangles that will newly appear
7   rects.enter();
8
9   // Rectangles that will disappear
10  rects.exit();
11
12  // Rectangles that exist.
13  rects;
```

Suppose on the canvas we have elements {1, 2, 3, 4, 5}
Now the new data is {1, 2, 3, 6, 7}.
Then we have:

Entering: {6, 7}
Exiting: {4, 5}
Updating: {1, 2, 3}

Mapping Rendering Properties

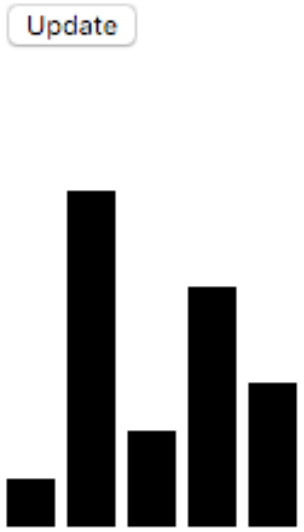


```
var draw = function() {
  // ensure that svg size is large enough to contain the graphics
  // otherwise some rendered elements will be truncated!
  var svg = d3.select('svg')
    .attr('width', 400)
    .attr('height', 300);
  var values = [1, 7, 2, 5, 3];

  var rects = svg.selectAll('rect').data(values);

  rects.enter()
    .append('rect')
    .attr('x', function(value, index) {
      return index * 25;
    })
    .attr('y', function(value, index) {
      // Note that y axis goes from top to bottom
      return 200 - value * 20;
    })
    .attr('height', function(value, index) {
      return value * 20;
    })
    .attr('width', 20);
};

$(document).ready(function() {
  draw();
});
```

Working with Updates

- Let's add some data updates.
- Once the update button is pressed, we try to update the bar cha with a new array of values.

```
var draw = function(values) {
  // ensure that svg size is large enough to contain the graphics
  // otherwise some rendered elements will be truncated!
  var svg = d3.select('svg')
    .attr('width', 400)
    .attr('height', 300);
  var rects = svg.selectAll('rect').data(values);

  rects.enter()
    .append('rect')
    .attr('x', function(value, index) {
      return index * 25;
    })
    .attr('y', function(value, index) {
      // Note that y axis goes from top to bottom
      return 200 - value * 20;
    })
    .attr('height', function(value, index) {
      return value * 20;
    })
    .attr('width', 20);
};

$(document).ready(function() {
  draw([1, 7, 2, 5, 3]);

  $('button').click(function() {
    draw([1, 2, 3, 4, 5]);
  });
});
```




```
<body>
  <button>Update</button>
  <div>
    <svg></svg>
  </div>
</body>
```

But …

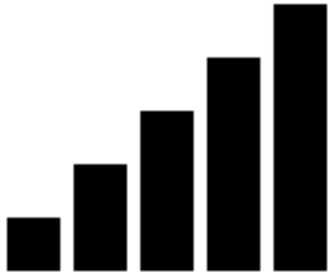
- But when we pressed the button, nothing happened!

- Why? Because we only set the rendering properties for the enter(), which contain only those newly entering data points.

Use the Updating Selection

```
1  var draw = function(values) {
2    var svg = d3.select('svg')
3      .attr('width', 400)
4      .attr('height', 300);
5    var rects = svg.selectAll('rect').data(values);
6
7    rects.enter()
8      .append('rect');
9
10   rects // UPDATE!
11     .attr('x', function(value, index) {
12       return index * 25;
13     })
14     .attr('y', function(value, index) {
15       return 200 - value * 20;
16     })
17     .attr('height', function(value, index) {
18       return value * 20;
19     })
20     .attr('width', 20);
21 };
```

Update



Use the Exiting Selection

```
1  var draw = function(values) {
2    var svg = d3.select('svg')
3      .attr('width', 400)
4      .attr('height', 300);
5    var rects = svg.selectAll('rect').data(values);
6
7    rects.enter()
8      .append('rect');
9
10   rects
11     .attr('x', function(value, index) {
12       return index * 25;
13     })
14     .attr('y', function(value, index) {
15       return 200 - value * 20;
16     })
17     .attr('height', function(value, index) {
18       return value * 20;
19     })
20     .attr('width', 20);
21
22   rects.exit().remove(); // EXIT!
23 };
24
25 $(document).ready(function() {
26   draw([1, 7, 2, 5, 3]);
27
28   $('button').click(function() {
29     draw([1, 2, 3]);
30   });
31 });
```

Custom Data Mapping

- Default data binding uses array index. But it is possible to bind data items by other properties, usually ids.
- This is particularly useful for rendering data in a form of compound objects.

```
draw([
  {id: 'a', value: 1},
  {id: 'b', value: 7},
  {id: 'c', value: 2},
  {id: 'd', value: 5},
  {id: 'e', value: 3}
]);
```

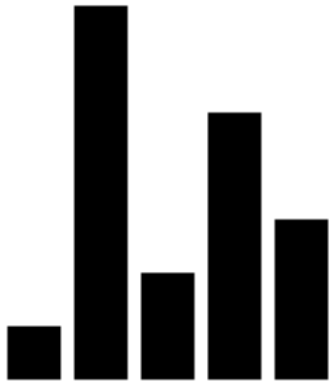
- We may add a functional mapping as:

```
var rects = svg.selectAll('rect').data(values, function(obj) {
  return obj.id;
});
```

Custom Data Mapping

- Each data item passed to the rendering is now the object, instead of the original numerical values.

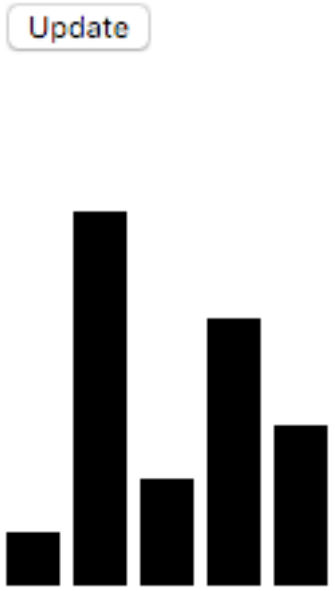
```
rects.enter()
  .append('rect');

rects
  .attr('x', function(obj, index) {
    return index * 25;
  })
  .attr('y', function(obj, index) {
    return 200 - obj.value * 20;
  })
  .attr('height', function(obj, index) {
    return obj.value * 20;
  })
  .attr('width', 20);

rects.exit().remove();
```



D3 Scale

Linear Scale

```
var yScale = d3.scale.linear()
  .domain([0, 10])
  .range([0, 200]);

rects.enter()
  .append('rect');

rects
  .attr('x', function(obj, index) {
    return index * 25;
  })
  .attr('y', function(obj, index) {
    return yScale(10) - yScale(obj.value);
  })
  .attr('height', function(obj, index) {
    return yScale(obj.value);
  })
  .attr('width', 20);

rects.exit().remove();
```

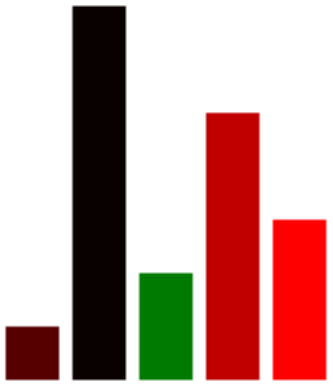


D3 Scale

Color Scale

```
var yScale = d3.scale.linear()
  .domain([0, 10])
  .range([0, 200]);
var colorScale = d3.scale.linear()
  .domain([0, .5, 1])
  .range(['red', 'black', 'green']);
rects.enter()
  .append('rect');
rects
  .attr('fill', function(obj) {
    return colorScale(obj.value2);
  })
  .attr('x', function(obj, index) {
    return index * 25;
  })
  .attr('y', function(obj, index) {
    return yScale(10) - yScale(obj.value);
  })
  .attr('height', function(obj, index) {
    return yScale(obj.value);
  })
  .attr('width', 20);
rects.exit().remove();
```

```
draw([
  {id: 'a', value: 1, value2: 0.333},
  {id: 'b', value: 7, value2: 0.482},
  {id: 'c', value: 2, value2: 0.977},
  {id: 'd', value: 5, value2: 0.125},
  {id: 'e', value: 3, value2: 0.000}
]);
```



D3 Axis

```
var drawAxis = function() {
  var svg = d3.select('svg');
  var xScale = d3.scale.linear()
    .domain([0, 100])
    .range([100, 800]);
  var xAxis = d3.svg.axis()
    .scale(xScale)
    .ticks(5)
    .orient('bottom');
  svg.append('g')
    .classed('axis', true)
    .attr('transform', 'translate(0, 50)')
    .call(xAxis);
};
$(document).ready(function() {
  d3.select('svg')
    .attr({width: 1000, height: 300});
  drawAxis();
});
```

CSS

```
.axis path, .axis line {
  fill: none;
  stroke: #000;
  shape-rendering: crispEdges;
}
```

Loading Data with D3

- D3 has built-in utilities for loading data, e.g. CSV and JSON.

```
1  d3.json('data.json', function(error, data) {
2    // data is now a variable parsed from JSON
3    // note that this callback is asynchronous!
4    console.log(data);
5  });
6
7  d3.csv('data.csv')
8    .row(function(d) {
9      // define the parsed value format for each row
10     // the attribute names under d are from the CSV header line
11     return {key: d.key, value: +d.value};
12   })
13   .get(function(error, rows) {
14     console.log(rows);
15   });
16
17 // if you already have a csv string as str
18 d3.csv.parse(str);
```

D3 Layout

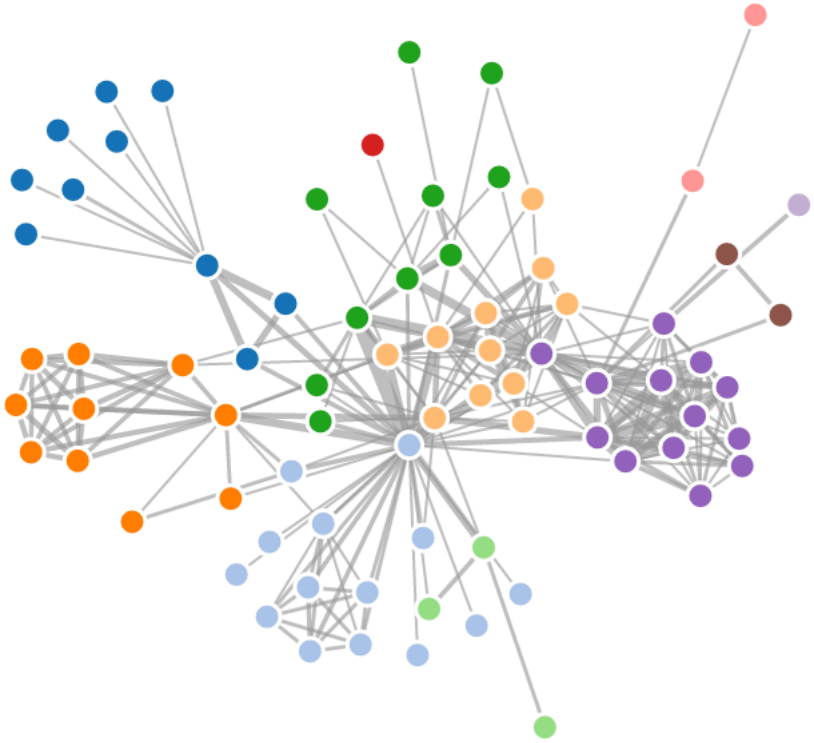
- A fastest way to get started with D3 layouts is to click on an example in the gallery and then look at the source code.

Data-Driven Documents

- Here we use an example of d3.layout.force, from http://bl.ocks.org/mbostock/4062045

Force-Directed Layout

- Edges pull nodes closer
- Nodes repulse each other



- Create a force

```
var width = 960, height = 500;
var force = d3.layout.force()
  .charge(-120) // force parameters
  .linkDistance(30)
  .size([width, height]); // define the forcing region
```

```
7   d3.json('miserables.json', function(error, graph) {
8     if (error) throw error;
9
10    force
11      .nodes(graph.nodes)
12      .links(graph.links)
13      .start();
14
15    var link = svg.selectAll('.link')
16      .data(graph.links)
17      .enter().append('line')
18      .attr('class', 'link')
19      .style('stroke-width', function(d) { return Math.sqrt(d.value); });
20
21    var node = svg.selectAll('.node')
22      .data(graph.nodes)
23      .enter().append('circle')
24      .attr('class', 'node')
25      .attr('r', 5)
26      .style('fill', function(d) { return color(d.group); })
27      .call(force.drag);
28
29    node.append('title')
30      .text(function(d) { return d.name; });
31
32    force.on('tick', function() {
33      link.attr('x1', function(d) { return d.source.x; })
34        .attr('y1', function(d) { return d.source.y; })
35        .attr('x2', function(d) { return d.target.x; })
36        .attr('y2', function(d) { return d.target.y; });
37      node.attr('cx', function(d) { return d.x; })
38        .attr('cy', function(d) { return d.y; });
39    });
40  });
```

Using the Force-Directed Layout

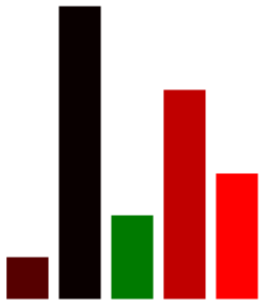
D3 Interactions

```
rects.enter()
  .append('rect')
  .on('mouseover', function(d) {
    d3.select('label').text(d.id + ': ' + d.value + ', ' + d.value2);
  });
```

```
$(document).ready(function() {
  d3.select('svg')
    .attr({width: 1000, height: 300});
  draw([
    {id: 'a', value: 1, value2: 0.333},
    {id: 'b', value: 7, value2: 0.482},
    {id: 'c', value: 2, value2: 0.977},
    {id: 'd', value: 5, value2: 0.125},
    {id: 'e', value: 3, value2: 0.000}
  ]);
});
```

```
<div>
  <label></label>
  <svg></svg>
</div>
```

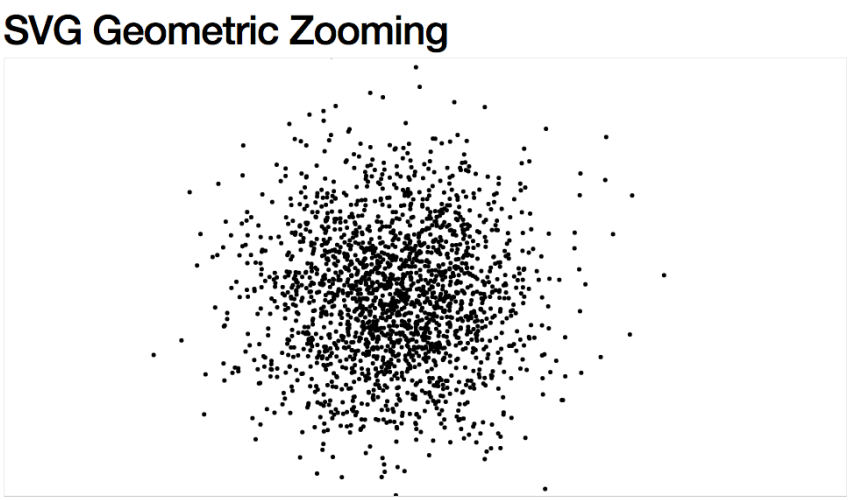
c: 2, 0.977



D3 Zoom

- There are multiple ways to achieve zooming. We take an example at http://bl.ocks.org/mbostock/3680999

```
var zoomObj = d3.behavior.zoom()
  .scaleExtent([1, 8])
  .on('zoom', zoomHandler);
var svg = d3.select('svg')
  .call(zoomObj)
  .append('g');
svg.append('rect')
  .attr('class', 'overlay')
  .attr('width', width)
  .attr('height', height);

svg.selectAll('circle')
  .data(data)
  .enter().append('circle')
  .attr('r', 2.5)
  .attr('transform', function(d) { return 'translate(' + d + ')'; });

function zoomHandler() {
  svg.attr('transform', 'translate(' + d3.event.translate + ')scale(' + d3.event.scale + ')');
}
```



Package Management

- Including a lot of library codes here and there will make your code messy and unstable (e.g. a CDN may go down).
- It would be safer to have the library files located on your server.
  - However, you don't want to include them as your source code.
  - You want the user of your code to be able to easily get started with your code.
- Therefore, we use package management systems for front-end development.

Bower

- bower is a front-end package management system
- You may create a package specification by running `bower init`
- This creates a package spec called `bower.json`

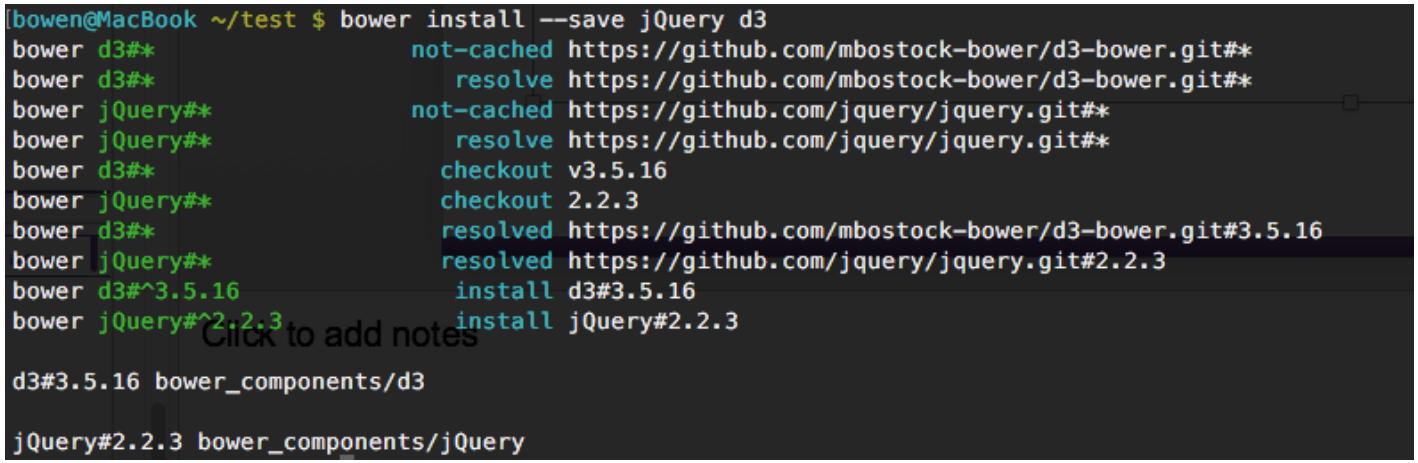
```
bowen@MacBook ~/test $ bower init
[? name test
[? description
[? main file index.html
[? keywords
[? authors Bowen Yu <yubowenok@gmail.com>
[? license MIT
[? homepage
[? set currently installed components as dependencies? Yes
[? add commonly ignored files to ignore list? Yes
[? would you like to mark this package as private which prevents it from being accidentally published to the registry? Yes

{
  name: 'test',
  authors: [
    'Bowen Yu <yubowenok@gmail.com>'
  ],
  description: '',
  main: 'index.html',
  license: 'MIT',
  homepage: '',
  private: true,
  ignore: [
    '**/.*',
    'node_modules',
    'bower_components',
    'test',
    'tests'
  ]
}

[? Looks good? Yes
bowen@MacBook ~/test $
```

Click to add notes

Install Your Packages

- First look for package names at http://bower.io/.
- For jQuery and D3:

```
bower install --save jQuery d3
```



- Save means that you want to update your `bower.json` to include those libraries as your dependencies, so that another user (after cloning your code) can just run `bower install` to get all required dependencies.

Use Bower Components

- Now include the downloaded bower components to replace the CDN sources.

```
<head>
  <title>My Website</title>
  <script src="bower_components/jQuery/dist/jquery.min.js"></script>
  <script src="bower_components/d3/d3.min.js"></script>
  <script type="text/javascript" src="main.js"></script>
  <link rel=stylesheet type="text/css" href="style.css"/>
</head>
```

- DO NOT commit bower components folder to Git! (only commit bower.json)