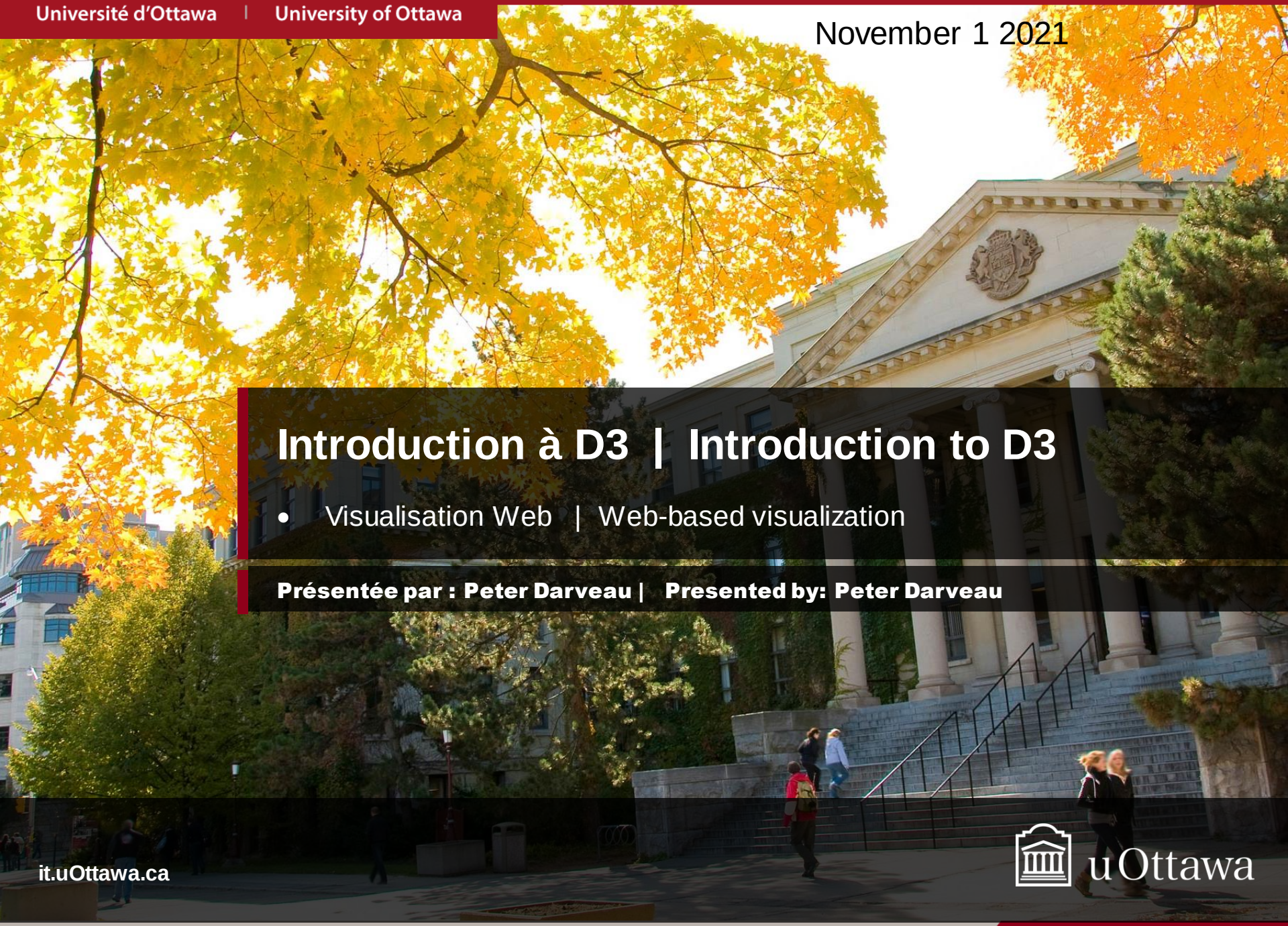
November 1 2021



Introduction à D3 | Introduction to D3

- Visualisation Web | Web-based visualization

Présentée par : Peter Darveau | Presented by: Peter Darveau

D3

- Data-Driven Documents (D3)
- JavaScript library
- Dynamic visualization

Gallery

- (link) https://www.theguardian.com/environment/interactive/2013/may/14/alaska-villages-frontline-global-warming
- (link) https://archive.nytimes.com/www.nytimes.com/interactive/2012/09/06/us/politics/convention-word-counts.html#Economy
- (link) http://www.nytimes.com/newsgraphics/2013/09/07/director-star-chart/index.html

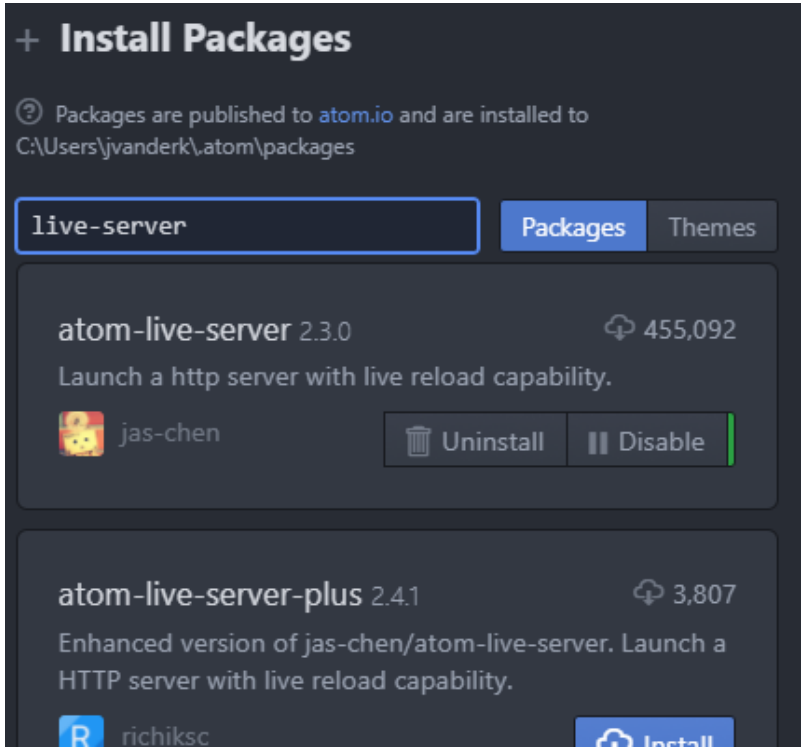
Outline

- Basic website building
- Basic vector graphics
- Low-level D3
- Pre-made D3 libraries

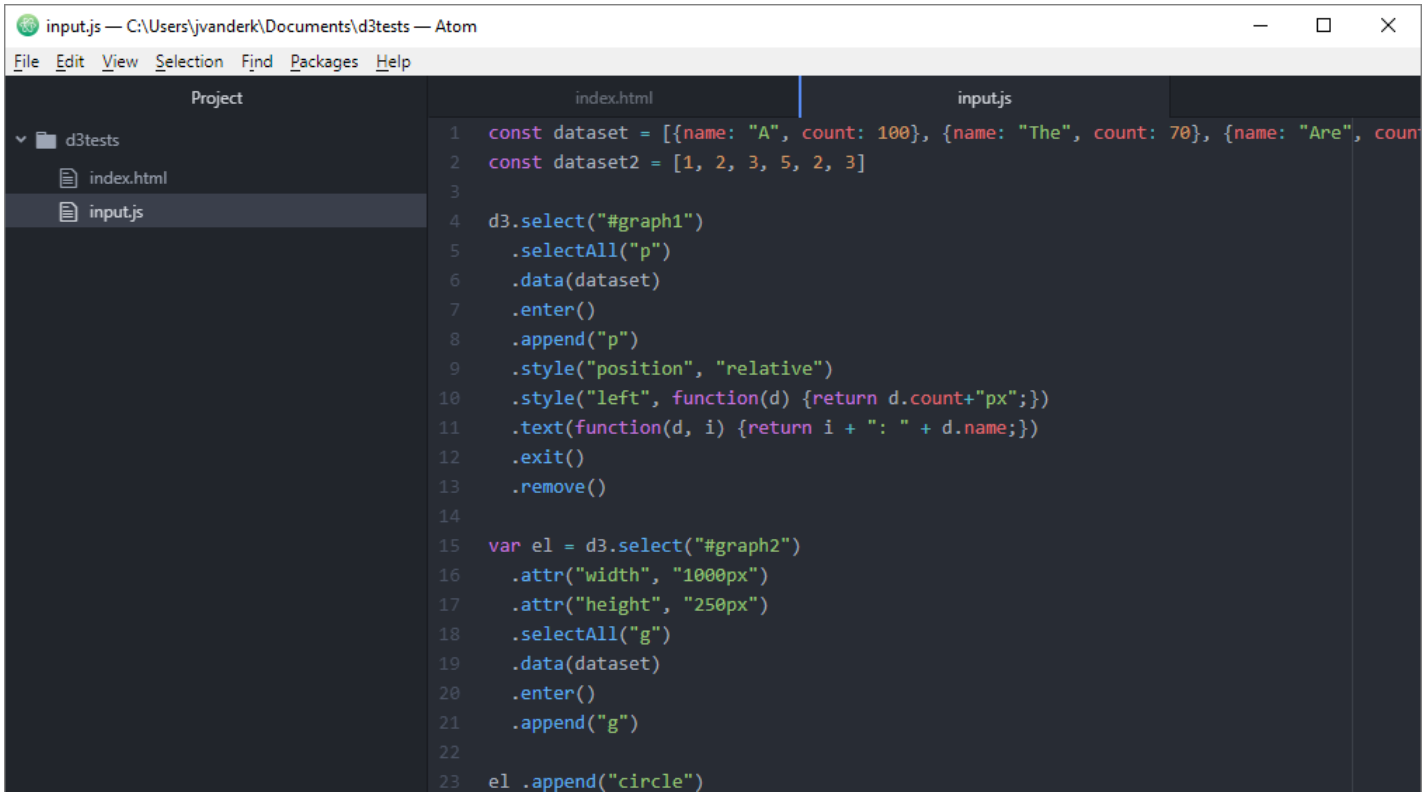
Download these slides: jarno.ca/d3.pdf

Workshop Requirements

- Atom
  - https://atom.io
- Atom-live-server
  - Edit or → Settings → Install
  - Search for live-server
  - Install
- Browser
  - Firefox, Chrome, Safari, Edge, Internet Explorer

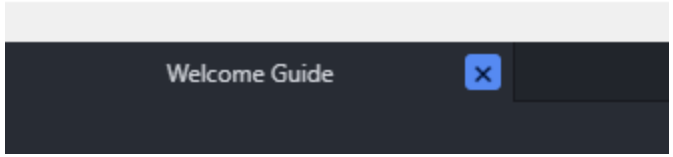


Atom editor


input.js — C:\Users\jvanderk\Documents\d3tests — Atom
File Edit View Selection Find Packages Help
Project
d3tests
index.html
input.js
index.html
input.js
const dataset = [{name: "A", count: 100}, {name: "The", count: 70}, {name: "Are", coun
const dataset2 = [1, 2, 3, 5, 2, 3]
d3.select("#graph1")
.selectAll("p")
.data(dataset)
.enter()
.append("p")
.style("position", "relative")
.style("left", function(d) {return d.count+"px";})
.text(function(d, i) {return i + ": " + d.name;})
.exit()
.remove()
var el = d3.select("#graph2")
.attr("width", "1000px")
.attr("height", "250px")
.selectAll("g")
.data(dataset)
.enter()
.append("g")
el .append("circle")

Starting a new project

- Create a new folder on your computer
  - How to do so depends on Windows, macOS or Linux
- File → Add Project Folder…
- Navigate to newly created folder and press "Select Folder".
- Close the "Welcome Guide"



Basic webpages

- HTML = Hyper Text Mark-up Language
- Uses XML Structure
- Provides the structure and text for websites

```
<!DOCTYPE html>
<html>
<head>
  <title>D3 tests</title>
  <meta charset="utf-8">
  <script src="https://d3js.org/d3.v5.min
</head>
<body>
  <h1>D3 testing ground</h1>
  <div id="graph1"></div>
  <svg id="graph2"></svg>
  <script src="input.js"></script>
</body>
</html>
```

Creating HTML

- Click on File → New File
- Click on File → Save
- Type "index.html" (without quotes)

- Result:
  - Empty page with "index.html" at the top

Simple web page

```
<!DOCTYPE html>
<html>
<head>
  <title>This is my website</title>
</head>
<body>
  <p>Hello world!</p>
</body>
</html>
```

- Click on File → Save
- Click on Packages → atom-live-server → Start server
- This will open your browser with "Hello world!"

Simple web page

Document type

HTML code goes inside here

Head contains stuff that is loaded first

Body contains the body of the text

p = paragraph

Always close every tag

```
<!DOCTYPE html>
<html>
<head>
 <title>This is my website</title>
</head>
<body>
 <p>Hello world!</p>
</body>
</html>
```

- Click on File → Save
- Click on Packages → atom-live-server → Start server

This will open your browser with "Hello world!"

Simple web page

```
<!DOCTYPE html>
<html>
<head>
  <title>This is my website</title>
</head>
<body>
  <p>Hello world!</p>
  <p>Another line of text</p>
</body>
</html>
```

- Click on File → Save
- This will update the website in your browser
- You now have a website!

Simple web page with style!

- CSS = Cascading Style Sheets
- Defines
  - Colours
  - Sizes
  - Location

- File → New File
- File → Save
- Type "style.css" (without quotes)

Simple web page with style!

```
body {
  background-color: lightblue;
}

p {
  color: red;
}
```

- Type text in "style.css"
- File → Save

Simple web page with style!

Corresponds to <body> tags in index.html

```
body {
  background-color: lightblue;
}

p {
  color: red;
}
```

- Type text in "style.css"
- File → Save

Corresponds to **all** <p> tags in index.html

Simple web page with style!

- Go to "index.html"
- Add line
- File → Save

```
<!DOCTYPE html>
<html>
<head>
  <title>This is my website</title>
  <link rel="stylesheet" href="style.css" type="text/css" />
</head>
…
…
```

Basic SVG

- Scalable Vector Graphics (SVG)
- Also uses XML structure
- Mark-up language for graphics, something that HTML doesn't do well
- All elements are basic shapes
- Infinite resolution

Basic SVG

```
<!DOCTYPE html>
<html>
<head>
  <title>This is my website</title>
  <link rel="stylesheet" href="style.css" type="text/css" /></head>
<body>
  <p>Hello world!</p>
  <p>Another line of text</p>
  <svg width="200px" height="200px">
    <circle cx="100px" cy="100px" r="100px" fill="blue"></circle>
  </svg>
</body>
</html>
```

Basic SVG

```
<!DOCTYPE html>
<html>
<head>
  <title>This is my website</title>
  <link rel="stylesheet" href="style.css" type="text/css" /></head>
<body>
  <p>Hello world!</p>
  <p>Another line of text</p>
  <svg width="200px" height="200px">
    <circle cx="100px" cy="100px" r="100px" fill="blue"></circle>
  </svg>
</body>
</html>
```

Start SVG

Add circle

cx = centre x coordinate

cy = centre y coordinate

r = radius of circle

Fill with colour blue

Basic SVG with style!

```
<!DOCTYPE html>
<html>
<head>
 <title>This is my website</title>
 <link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
 <p>Hello world!</p>
 <p>Another line of text</p>
 <svg width="200px" height="200px">
  <circle cx="100px" cy="100px" r="100px" class="somecategory">
  </circle>
 </svg>
</body>
</html>
```

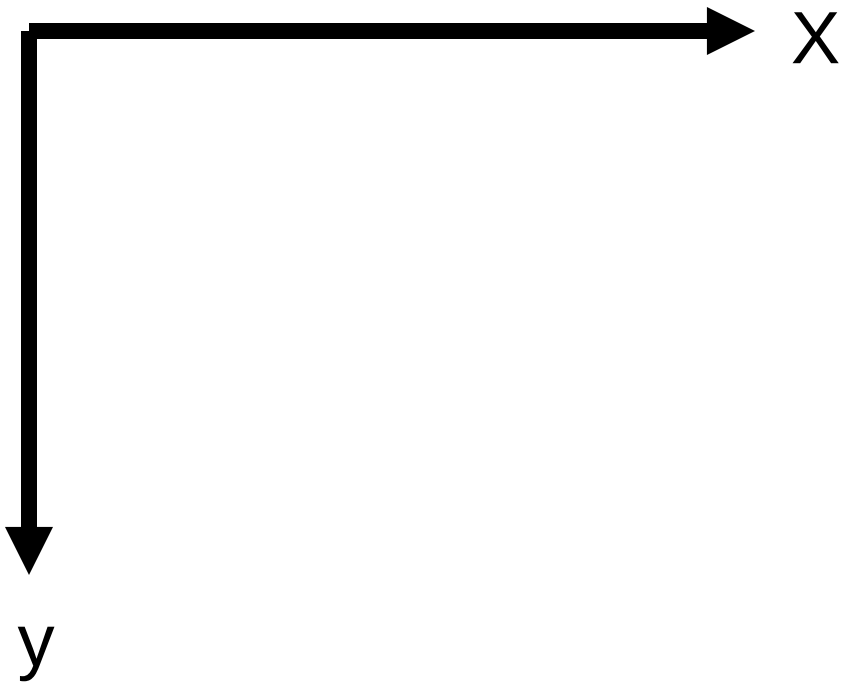
Basic SVG with style!

```
body {
  background-color: lightblue;
}

div {
  color: red;
}

.somecategory {
  fill: blue;
}
```

- IMPORTANT: There is a period in front of somecategory

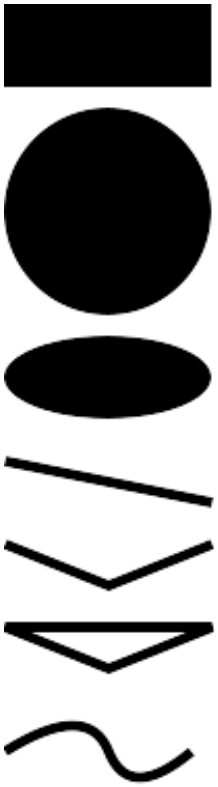
Coordinate system

- Note: On computer screens, y is "upside-down"



Other SVG shapes

- rect
- circle
- ellipse
- line
- polyline
- polygon
- path
- g (group of shapes)



https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Basic_Shapes

What we have so far

- A simple website
  - With style
- A simple SVG embedded in the website
  - With style

- Everything is static

Onward to D3.js

- Outline
  - Loading the library
  - Using the library
  - Loading data

D3.js

- D3 is a JavaScript library
- D3 ignores errors so it's hard to spot them

Tips for troubleshooting

- Use Developer Tools
- Switch to "Console"
  - Safari
    - ⌘ ⌥ I
  - Every other browser
    - F12

Loading D3.js

- Open index.html
- Add <script… line
- File → Save

```
<!DOCTYPE html>
<html>
<head>
  <title>This is my website</title>
  <link rel="stylesheet" href="style.css" type="text/css" />
  <script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<body>
...
...
</body>
</html>
```

Using D3.js

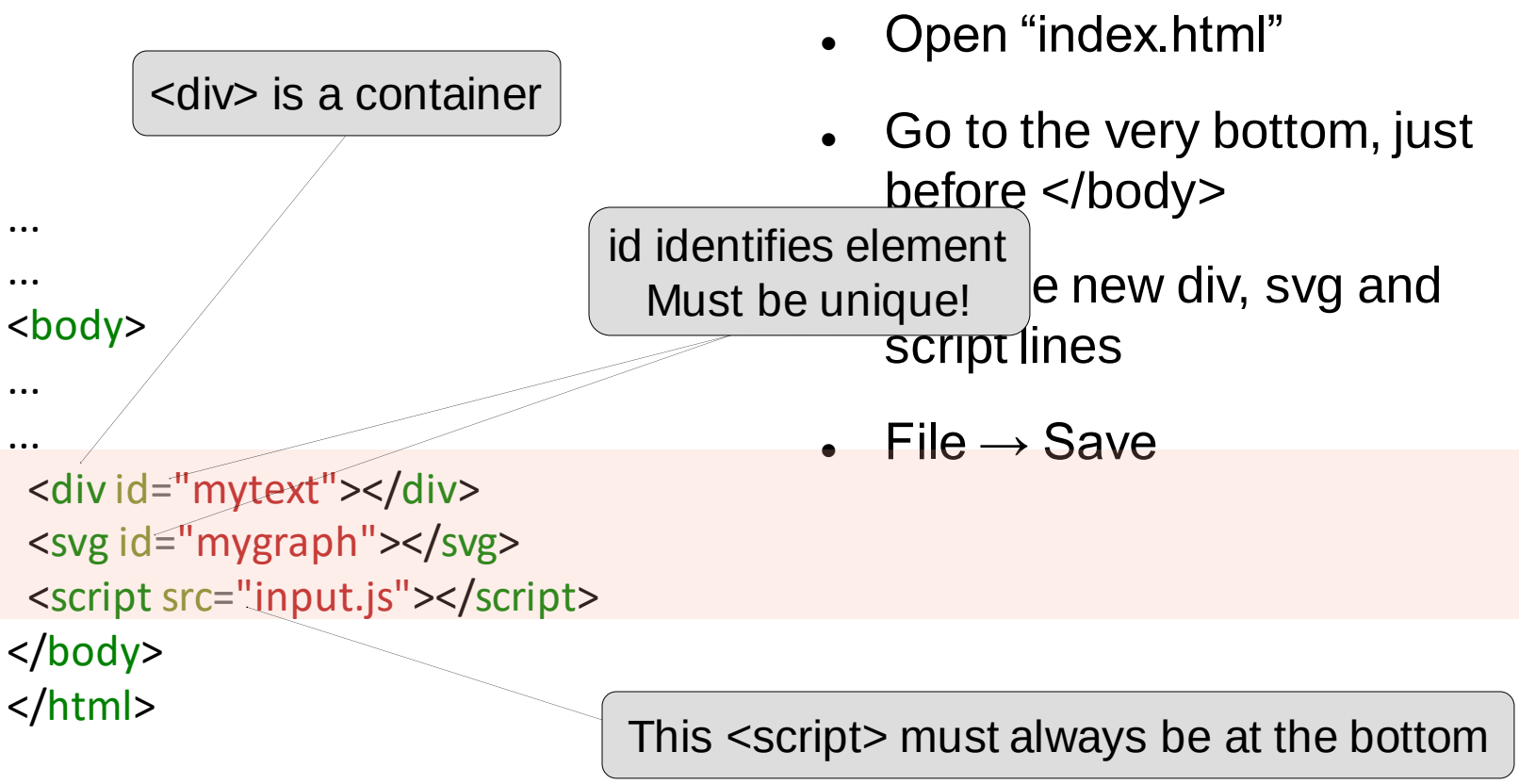
- We need to create a new file with our script
- In Atom:
  - File → New
  - File → Save
  - Call it "input.js"

Using D3.js

- Open "index.html"
- Go to the very bottom, just before </body>
- Add the new div, svg and script lines
- File → Save

```
...
...
<body>
...
...
  <div id="mytext"></div>
  <svg id="mygraph"></svg>
  <script src="input.js"></script>
</body>
</html>
```

Using D3.js



- Open "index.html"
- Go to the very bottom, just before </body>
- e new div, svg and script lines
- File → Save

Summary so far

- We have three files:
  - index.html
  - style.css
  - input.js
- Functions:
  - Text and structure
  - Colour and position
  - Executable code

Dynamic text

- Open “input.js”
- Type the text
- File → Save

```
d3.select("#mytext")
  .append("p")
  .text("Hello dynamic world")
```

Dynamic text

- Open "input.js"
- Type the text
- File → Save

Use d3 to access the library

Select the element with id "mytext" in index.html

```
d3.select("#mytext")
  .append("p")
  .text("Hello dynamic world")
```

Append a paragraph element

Insert text into the added element p

Dynamic shapes

- Open "input.js"
- Type the text
- File → Save

```
d3.select("#mygraph")
  .attr("width", "200px")
  .attr("height", "200px")
  .append("circle")
    .attr("cx", "100")
    .attr("cy", "100")
    .attr("r", "100")
    .attr("fill", "pink")
```

Dynamic shapes

With D3 we have:

```
d3.select("#mygraph")
  .attr("width","200px")
  .attr("height","200px")
  .append("circle")
    .attr("cx", "100")
    .attr("cy", "100")
    .attr("r", "100")
    .attr("fill","pink")
```

Before, in HTML, we had:

```
<svg width="200px" height="200px">
 <circle cx="100px" cy="100px" r="100px" fill="blue"></circle>
</svg>
```

Dynamic shapes

With D3 we have:

Change/set attributes

Append circle

```
d3.select("#mygraph")
  .attr("width","200px")
  .attr("height","200px")
  .append("circle")
    .attr("cx", "100")
    .attr("cy", "100")
    .attr("r", "100")
    .attr("fill","pink")
```

Before, in HTML, we had:

```
<svg width="200px" height="200px">
 <circle cx="100px" cy="100px" r="100px" fill="blue"></circle>
</svg>
```

The "Data-Driven" part

- The power of D3 is the representation of data
- Website changes dynamically depending on input

Data input

- Open "input.js"
- Type the text at the very top
- File → Save

```
var mynumberdata=[100,400,303,80,40]
var mytextdata=["begin", "middle", "end"]
```

Data input

- Open "input.js"
- Type the text at the very top
- File → Save

"var" means variable

```
var mynumberdata=[100,400,303,80,40]
var mytextdata=["begin", "middle", "end"]
```

Square brackets for arrays

Using data for text

- Open "input.js"
- Replace #mygraph or add d3.select("#mytext")....
- File → Save

```
d3.select("#mytext")
  .selectAll("p")
  .data(mytextdata)
  .enter()
  .append("p")
  .text("Hello dynamic world")
```

Using data for text

- Open "input.js"
- Replace d3.select("#mytext")....
- File → Save

Empty set of all paragraph tags p

Use this dataset

Enter the dataset

```
d3.select("#mytext")
  .selectAll("p")
  .data(mytextdata)
  .enter()
  .append("p")
  .text("Hello dynamic world")
```

Using data for text

- Open "input.js"
- Replace .text
- File → Save

```
d3.select("#mytext")
  .selectAll("p")
  .data(mytextdata)
  .enter()
  .append("p")
  .text(function(d,i){return d})
```

Using data for text

- Open "input.js"
- Replace .text
- File → Save

```
d3.select("#mytext")
  .selectAll("p")
  .data(mytextdata)
  .enter()
  .append("p")
  .text(function(d,i){return d})
```

d = data item

i = index (starts at 0)

Using data for SVG

- Open "input.js"
- Replace d3.select("#mygraph")....
- File → Save

```
d3.select("#mygraph")
    .attr("width","100%")
    .attr("height","200px")
  .selectAll("circle")
  .data(mynumberdata)
    .enter()
    .append("circle")
      .attr("cx", function(d, i){return 100 + i*100})
      .attr("cy", "100px")
      .attr("r", function(d, i){return d/8})
      .attr("fill","pink")
```

Using data for SVG with style

- Open "input.js"
- Replace d3.select("#mygraph")....
- File → Save

```
d3.select("#mygraph")
    .attr("width","100%")
    .attr("height","200px")
  .selectAll("circle")
  .data(mynumberdata)
    .enter()
    .append("circle")
      .attr("cx", function(d, i){return 100 + i*100})
      .attr("cy", "100px")
      .attr("r", function(d, i){return d/8})
      .attr("class", "somecategory")
```

Data-driven

- Open "input.js"
- Go to the top
- Change the data
- File → Save

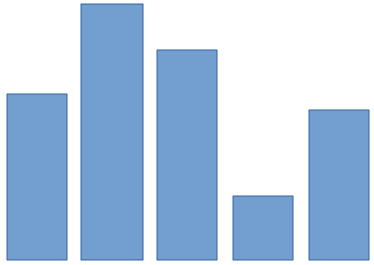
```
var mynumberdata=[100,40]
var mytextdata=["begin","middle","end","more", "words"]
```

Summary so far

- We can now dynamically create a website
- We can add text and SVG
- We can use data and generate content

Creating a bar chart

- We will use rectangles now
  - <rect x="0" y="0" width="35" height="100"></rect>
- We have the tools!
- Steps to take:
  - Add a new <svg> element to the html page with id "barchart"
  - In input.js, add d3 commands to format the rectangles (similar to the circles):



A solution

- index.html

```
<svg id="barchart"></svg>
```

- input.js

```
d3.select("#barchart")
  .attr("width", "100%")
  .attr("height", "450px")
.selectAll("rect")
.data(mynumberdata)
  .enter()
  .append("rect")
    .attr("x", function(d, i) {return i*40})
    .attr("y", function(d, i) {return 400 - d})
    .attr("width", 35)
    .attr("height", function(d, i) {return d})
```

Loading external data

- We can now create graphs
- But the data still comes from input.js
- D3 supports loading
  - CSV
  - TSV
  - JSON

Example data

- Canadian Women's Labour-Force Participation
- https://vincentarelbundock.github.io/Rdatasets/datasets.html

Loading the data

- Using d3.csv, we load, then format, and then display

```
d3.csv("https://vincentarelbundock.github.io/Rdatasets/csv/carData/Bfox.csv",
 function(d,i){
  // function to format or fix data (optional)
  return {}
 }).then(function(data){
  // Do something with data
  // This is where we generate our bar chart
 }).catch(function(error){
  // Show error
 })
```

When reading files, the **.then** method is necessary to make sure all the data is loaded

Loading the data

- Open "input.js"
- Go to just above "d3.select("#barchart")
- Add the text
- File → Save

```
d3.csv("https://vincentarelbundock.github.io/Rdatasets/csv/carData/Bfox.csv",
  function(d,i) {
    // function to format or fix data (optional)
    return {
      year: Object.values(d)[0],
      menwage: +d.menwage,
      womwage: +d.womwage,
      avgwage: (+(d.menwage)+(+d.womwage)) / 2
    }
  }
)
```

Loading the data

- Open "input.js"
- Go to just above "d3.select("#barchart")
- Add the text
- Save

d3.csv loads the data

The second argument is a function to format the data

returns the new columns in curly brackets

First column has no name

'+' sign converts to number

You can create new columns too

```
d3.csv("https://vincentarelbundock.github.io/Rdatasets/csv/carData/Bfox.csv",
 function(d,i) {
  // function to format or fix data (optional)
  return {
   year: Object.values(d)[0],
   menwage: +d.menwage,
   womwage: +d.womwage,
   avgwage: (+(d.menwage)+(+d.womwage)) / 2
  }
}
```

Loading data

- Your old bar chart generating function goes inside "then"

```
.then(function(data){
  d3.select("#barchart")
    .attr("width","100%")
    .attr("height","450px")
  .selectAll("rect")
  .data(data)
    .enter()
    .append("rect")
      .attr("x", function(d, i) {return i*40})
      .attr("y", function(d, i) {return 400-10*d.womwage})
      .attr("width",35)
      .attr("height", function(d, i) {return 10*d.womwage})
})
```

Handling load errors

- Checking for error is extremely useful
- Errors are "caught"

```
.catch(function(error) {
  d3.select("#barchart")
  .attr("width", "100%")
  .attr("height", "450px")
  .append("text")
    .attr("x", "50%")
    .attr("y", "225px")
    .text(error)
})
```

- Make a spelling error in the URL, see what happens

Multiple columns

- Until now we only have used one column
- Our data file has multiple columns
- First, use variables to split the code

```
var svg = d3.select("#barchart")
  .attr("width", "100%")
  .attr("height", "450px")
svg.selectAll("rect")
.data(data)
  .enter()
  ...
```

Multiple columns

- Until now we only have used one column
- Our data file has multiple columns
- Use variables to split the code

"barchart" element stored in "svg"

```
var svg = d3.select("#barchart")
  .attr("width","100%")
  .attr("height","450px")
svg.selectAll("rect")
.data(data)
  .enter()
  ...
```

Applies to all rect in "svg" group

Adding more charts

- Copy-paste all of the svg.selectAll("rect") so that you have two copies in "input.js"
- Change the first from womwage to menwage and add colour

```
svg.selectAll("rect")
.data(data)
  .enter()
  .append("rect")
    .attr("x", function(d, i) {return i*40})
    .attr("y", function(d, i) {return 400-10*d.menwage})
    .attr("width", 30)
    .attr("height", function(d, i) {return 10*d.menwage})
    .attr("fill", "red")
  .exit()
```

Adding more charts

- Why did that not work?

```
svg.selectAll("rect")
.data(data)
  .enter()
  .append("rect")
    .attr("x", function(d, i) {return i*40})
    .attr("y", function(d, i) {return 400-10*d.menwage})
    .attr("width", 30)
    .attr("height", function(d, i) {return 10*d.menwage})
    .attr("fill", "red")
  .exit()
```

Adding more charts

- Why did that not work?

No longer selecting an empty set to add to

Data gets replaced

```
svg.selectAll("rect")
.data(data)
  .enter()
  .append("rect")
    .attr("x", function(d, i) {return i*40})
    .attr("y", function(d, i) {return 400-10*d.menwage})
    .attr("width", 30)
    .attr("height", function(d, i) {return 10*d.menwage})
    .attr("fill", "red")
  .exit()
```

Solution

- Add classes to distinguish

```
svg.selectAll("rect.men")
.data(data)
  .enter()
  .append("rect")
    .attr("x", function(d, i) {return i*40})
    .attr("y", function(d, i) {return 400-10*d.menwage})
    .attr("width", 30)
    .attr("height", function(d, i) {return 10*d.menwage})
    .attr("fill", "red")
    .attr("class", "men")
  .exit()
```

Signals the end of the newly created Class

- Same for women

Play to see what happens

- Try changing numbers
- Try also plotting "avgwage"

- Bonus points:
  - Use "style.css" to change bar colours (remember classes start with a period in CSS)
  - Add the years with append("text")
  - Add x and y axis

A final note about good coding practices …

- Treat source code primarily as a means for communicating with peers, supervisor or IT support staff
- Indent code consistently
- Don't skimp on spacing and squash the code up – make it easy to read
- Apply your chosen style consistently
- Keep out unused code
- Don't skip writing comments:
  - Avoid parroting the code
  - Ensure they add to the understanding of your code
- Use descriptive variable and function names

Following these simple suggestions go a long way in helping others help you !

Further resources

- linkedin.com/learning/search?keywords=d3
  - Access to a large library of video workshop. Free for Ontario schools.
- D3js.org
  - Examples and documentation
- Bl.ocks.org
  - Examples

Further assistance

- For help with
  - Compute Canada
  - Programming
  - Servers
  - Any research-related computing needs
- Contact Jarno van der Kolk or Peter Darveau through Service Desk
  - https://topdesk.uottawa.ca
- Or email
  - jvanderk@uottawa.ca
  - pdarveau@uottawa.ca