



# HCDE 530: Computational Techniques for HCDE

## Data Visualization in Web, Part 2

David McDonald, Sungsoo (Ray) Hong  
University of Washington

## Before we start

- Download HCDE530\_D3\_part2.zip in the course website and unzip it

## What we will cover today

- DOM tree, SVG, selectors, JavaScript
- Basic JavaScript concepts (event driven mechanism, selection, a little bit of jQuery)
- Preparing data for using D3 and visualizing information
- Creating a D3 bar graph step by step
- Adding some interaction to bar graph: hover over
- Creating a D3 line chart step by step
- Adding some interaction to line chart: on and off

## HTML, DOM

- HTML presents various types of **content** (e.g., texts, images, or video) via **elements** (i.e., tags).
- Each element can have **attributes**.
- Each element in HTML is **hierarchically structured** and the structure is often called **DOM tree**.

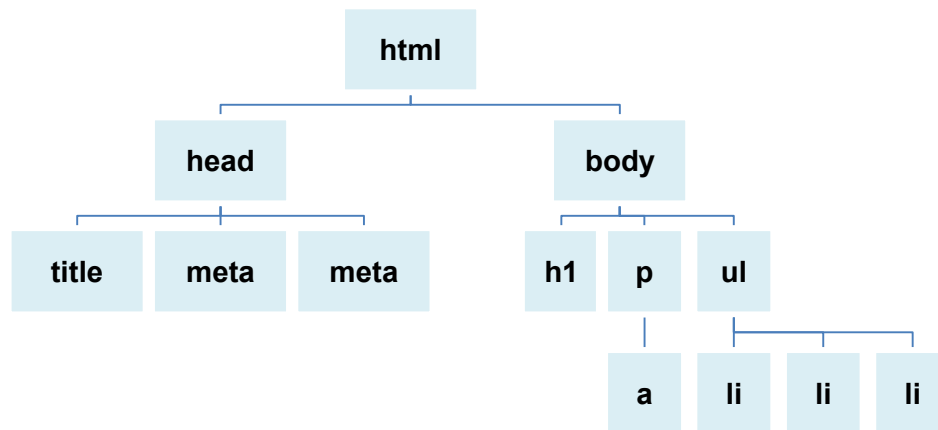
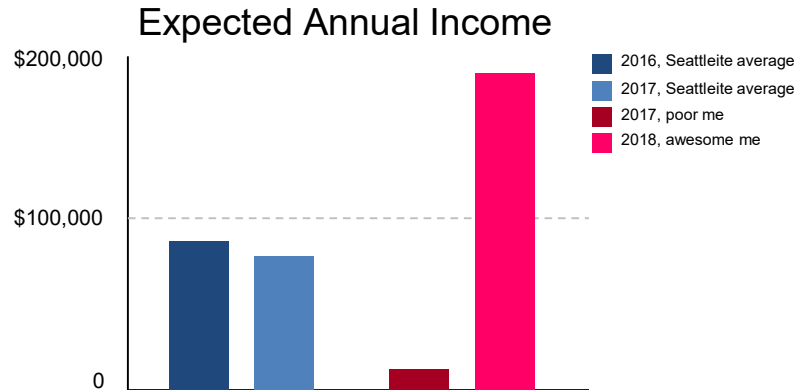


Figure: Example of DOM tree

## SVG

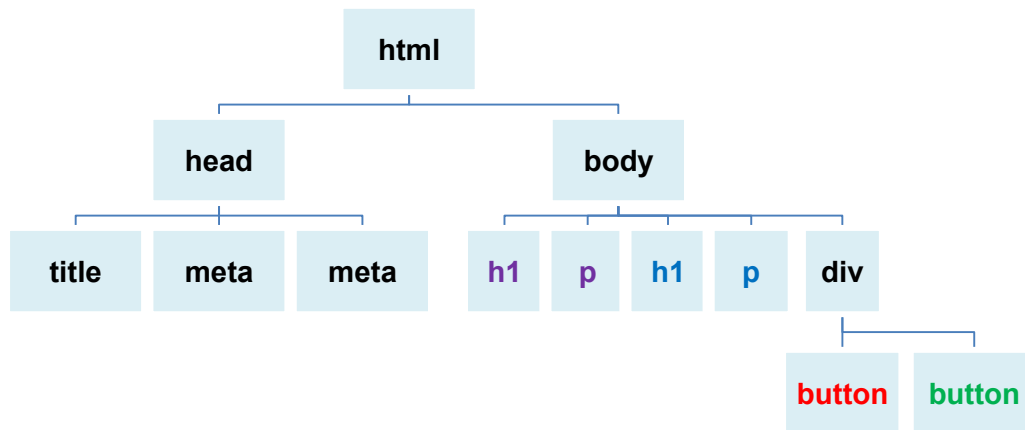
- SVG is an standardized vector image format for web.
- SVG specifications include a variety of two-dimensional graphics with support for animation.
- To use, `<svg>` should be added to DOM tree. Then tags such as `rect`, `circle`, `line`, `text`, `path` can be added



## JavaScript

- **JavaScript** allow you to programmatically **create and modify** some part of **DOM tree** based on a **event** that a user can trigger.
- The following three are all about the JavaScript.
  - Condition:** based on some **event** (e.g., click, mouse over, mouse down) triggered by a user
  - Object:** **select** some part of DOM tree
  - Action:** **modify** the structure or attributes of some selected attributes
- jQuery and D3 are widely used JavaScript library.
  - jQuery** is used to get user **event** and **modify** the DOM tree.
  - D3** is used to convert quantitative information to SVG graphics.
- Will cover Object and Action part, then Condition later.

## Selector (&amp; event)



- **Condition.** When a user *click* **button**, **Object.** Select **h1** and **p**. **Action.** Change the contents inside the tags
- **Condition.** When a user *mouse over* **button**, **Object.** Select **h1** and **h1**. **Action.** Change font-family of the tags
- Selector: id selector, class selector, and tag selector

## Revisit: selector

### CSS with class selector

```
.class_name{  
    attribute1: value1;  
    attribute2: value3;  
    attribute2: value3;  
    ...  
}
```

### CSS with id selector

```
#id_name{  
    attribute1: value1;  
    attribute2: value2;  
    attribute3: value3;  
    ...  
}
```

### CSS with tag selector

```
tag_name{  
    attribute1: value1;  
    attribute2: value2;  
    attribute3: value3;  
    ...  
}
```



# jQuery

- Widely used JavaScript library
- Selection + Action mechanism
- Selector:

Class selector `$(".class")`, ID selector `$("#id")`,

Element selector `$( "element" )`, Descendant Selector `$( "selector1 selector2" )`,

<https://api.jquery.com/category/selectors/>

- Action (for modifying DOM tree):

`.attr()`, `.remove()`, `.empty()`, `.append()` (and more)

## Demonstration

- **Object** . Select which element(s) to modify. **Action**. specify which action(s) to do
- Visit [www.rayhong.net](http://www.rayhong.net) with Firefox
- Click right mouse button and select “Inspect Element”
- Let’s see how the DOM tree looks like first.

Select this tab

Then type here

The screenshot shows a browser window with a page titled "Hiking North West" dated "Sep 26 2016". The page content includes a paragraph about hiking in the Pacific North West and another paragraph about the benefits of hiking in Washington State. On the left, a DOM tree is visible with categories like "Maul", "Tree", "Hiking", "Austin", "Beer", "P.L.C. MyTime", "Astoria", "VDMO", "CH2015", "San Diego", "Hair", "Wingley", "Traffigram", "Research", "Life", "Design", and "Travel". On the right, the browser's developer tools are open, showing the "css" tab selected (indicated by a red box). The console shows a message: "Unknown property 'moz-box-shadow'. Declaration dropped." and "rayhong.css:63:18". Below the console, there is an input field (indicated by a red box) for typing CSS rules.

- Let's see how the DOM tree looks like first.
- Click the second tab.
- Let's type some codes. e.g.,

```
$(".news rect").attr("height", 50)
```

```
$(".news rect").attr("width", 30)
```

```
$("#g_5 rect").attr("fill", "rgb(255,0,0)")
```

Object

Action

## jQuery event

- Condition that specific action can be triggered.

<http://api.jquery.com/category/events/>

Some fancy events here: click, dbclick, mousedown, **mouseenter**, mouseleave, mousemove, mouseout, mouseover, mouseup, ready

- `$(selector).on(which event, function(){do which action});`

```
$("#button1").on("click", function(){  
    console.log("You clicked the button1!");  
    $("h1").attr("font-family", "Source Sans Pro");  
});
```

- Let's see the event **ready**

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

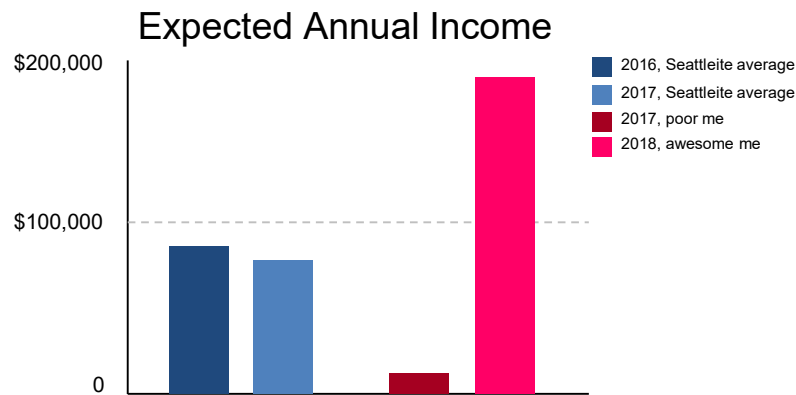
## D3

- Help designers to easily convert some quantitative / categorical array of data into visual
- **Condition:** a user requested some visualization
- **Object:** find a place to create a visualization
- **Action:** Get the data (e.g., from server) and convert that data to the right type of SVG element

Input	Output	
Quantitative	Size	Position
Categorical	Color	Length

## D3

- Help designers to easily convert some quantitative / categorical array of data into visual
- **Condition:** a user requested some visualization
- **Object:** find a place to create a visualization
- **Action:** Get the data (e.g., from server) and convert that data to the right type of SVG element



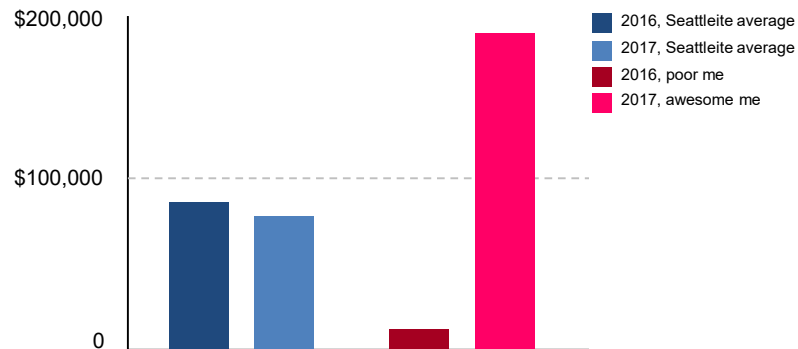
- **Action:** Get the data (e.g., from server) and convert that data to the right type of SVG element

Input

Category, Amount  
2016\_Seattleite, 80000  
2017\_Seattleite, 70000  
2016\_me, 20000  
2017\_me, 190000



Output



- Action: **Get the data** (e.g., from server) and convert that data to the right type of SVG element
- Prepare: open bar.tsv, line\_data.tsv


1	letter	frequency
2	km	79793
3	#nikeplus	59362
4	#fitness	57383
5	run	55250
6	mi	53827
7	Just	50756
8	#RunKeep	40992
9	complete	35982

date	#RunKeep	complete	Check	#fitness	#Fitness	en
2013-01-01-00	380	340	233	461	110	61
2013-01-01-02	289	253	202	438	91	34
2013-01-01-04	204	181	141	322	119	53
2013-01-01-06	363	332	233	342	86	181
2013-01-01-08	734	676	424	372	96	320
2013-01-01-10	655	606	354	414	91	283
2013-01-01-12	742	654	437	460	102	300
2013-01-01-14	920	827	524	704	130	346
2013-01-01-16	771	677	448	717	159	364
2013-01-01-18	535	476	354	733	190	282
2013-01-01-20	425	371	290	622	125	160
2013-01-01-22	415	384	259	694	122	80
2013-01-02-00	389	338	240	584	206	99
2013-01-02-02	282	249	196	491	91	50
2013-01-02-04	247	213	165	399	88	55
2013-01-02-06	273	249	181	341	64	86
2013-01-02-08	486	422	286	415	122	138
2013-01-02-10	586	509	375	548	158	195
2013-01-02-12	563	478	377	811	167	248
2013-01-02-14	509	442	322	897	165	212
2013-01-02-16	664	599	429	923	263	373
2013-01-02-18	652	564	369	967	204	546
2013-01-02-20	484	410	309	1000	214	187
2013-01-02-22	632	558	1694	798	1437	197
2013-01-03-00	554	472	1252	888	1052	192
2013-01-03-02	310	260	207	618	112	142



- Action: **Get the data** (e.g., from server) and convert that data to the right type of SVG element
- Prepare: bar.tsv, line\_data.tsv
- Open: vis\_bargraph.js

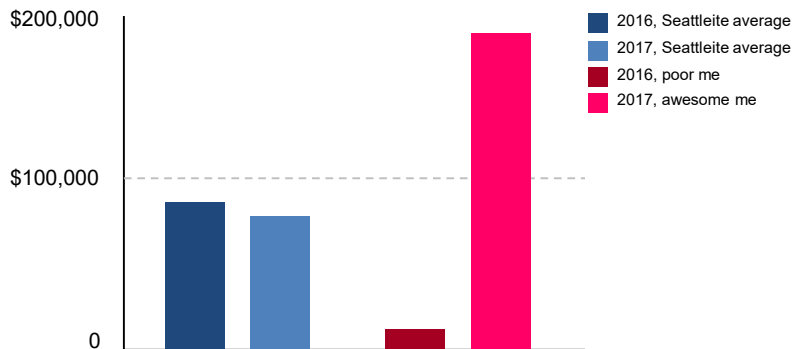
```
$(document).ready(function(){  
  d3.tsv("data/bar.tsv", function(error, data){  
    if (error) throw error;  
    console.log(data);  
  });  
});
```



```
▼ Array[24]  
▼ 0: Object  
  frequency: "79793"  
  letter: "km"  
  ▶ __proto__: Object  
▼ 1: Object  
  frequency: "59362"  
  letter: "#nikeplus"  
  ▶ __proto__: Object  
▼ 2: Object  
  frequency: "57383"  
  letter: "#fitness"  
  ▶ __proto__: Object  
▶ 3: Object  
▶ 4: Object  
▶ 5: Object  
▶ 6: Object  
▶ 7: Object  
▶ 8: Object
```

- Action: Get the data (e.g., from server) and **convert that data to the right type of SVG element**
- **Scale, that is used to convert the input to fit into the screen**

Category, Amount  
2016\_Seattleite, 80000  
2017\_Seattleite, 70000  
2016\_me, 20000  
2017\_me, 190000



Category (e.g., 2016\_Seattleite) to bin (i.e., the first bin in x axis)

Quantity (e.g., 80000) to position (i.e., the height of the dark-blue bar)

- Action: Get the data (e.g., from server) and **convert that data to the right type of SVG element**
- **Scale, that is used to convert the input to fit into the screen**

Category (e.g., 2016\_Seattleite) to bin (i.e., the first bin in x axis)

Quantity (e.g., 80000) to position (i.e., the height of the dark-blue bar)

Domain

Data

Input

Range

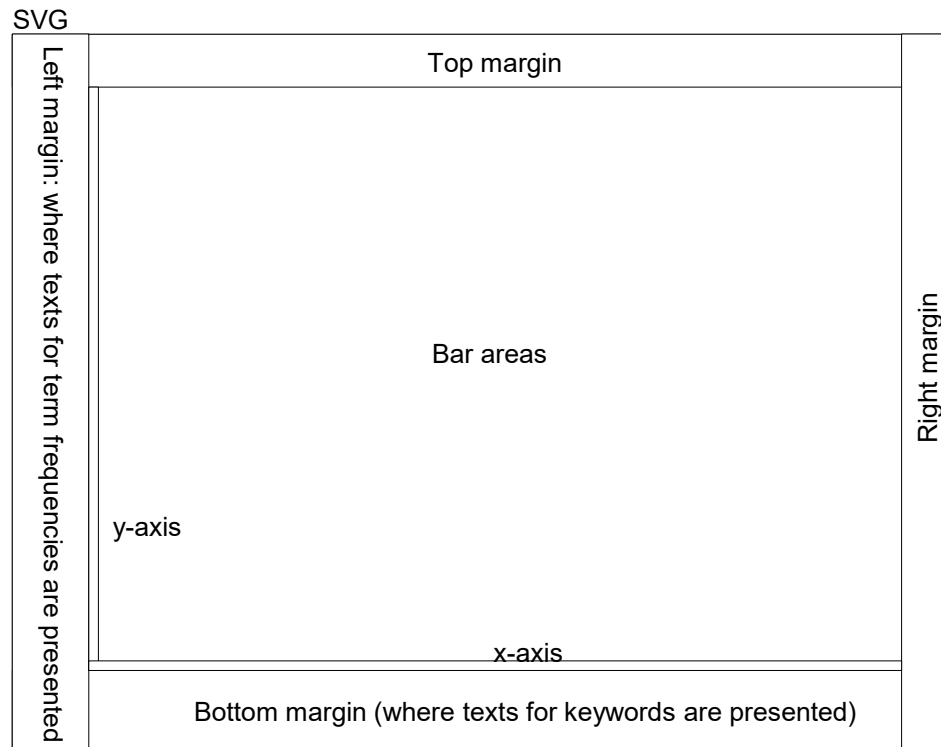
SVG attribute (e.g., x, y, height, width, cx, cy)

Output

# D3, convert from data to SVG



Domain for y axis = "frequency"  
Range for y axis = the height of  
rect for each "frequency"



Domain for x axis = "letter"  
Range for x axis = the x coordinate of rect for each "letter"

- Let's see the D3 js code:  
Open vis\_bargraph.js

- Let's write some code (presenting a tooltip):
- Open index.html and add `<div id="popup"></div>` in `class="visualization">`
- Open mycss.css and remove `/* ... */` in #popup
- Open vis\_bargraph.js

1. Add event for `mouseover`, `mousemove`, and `mouseout`

2. When mouse is over

get the id: `$(this).attr("id")`

get the mouse coordinate: `d3.mouse(this)[0]` for x coordinate,

`d3.mouse(this)[1]` for y coordinate

When mouse is moving

When mouse is out

- Let's see the D3 js code:  
Open `vis_linechart.js`
- For each keyword,
- X axis: time
- Y axis: frequency

Thank you!