D3 Tutorial

Introduction of Basic Components: HTML, CSS, SVG, and JavaScript

D3.js Setup
HTML - Hyper Text Markup Language

• HTML is the standard markup language for creating Web pages
  • HTML describes the structure of Web pages using markup

• HTML elements
  • HTML elements are the building blocks of HTML pages
  • represented by tags

• Tags
  • HTML tags label pieces of content such as
    • <head> tag for “heading”
    • <p> for “paragraph”
    • <table> for “table” and so on
  • Browsers do not display the HTML tags, but use them to render the content of the page
• If we display the information only by plain text

HTML Basics

HTML is designed for marking up text by adding tags such as <p> to create HTML elements.

Example image:
HTML - Codes and the Result

```html
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Tutorial</title>
  </head>
  <body>
    <h1>HTML Basics</h1>
    <p><strong>HTML</strong> is designed for marking up text by adding tags such as <em>em</em> by adding tags such as <code>&lt;p&gt;&lt;p&gt;</code> to create HTML elements.</p>
    <p><strong>Example image:</strong></p>
    <img src="https://www.osu.edu/assets/web/logo-web/TheOhioStateUniversity-Stacked.jpg" style="width: 200px" />
  </body>
</html>
```
HTML - DOM

• When a web page is loaded, the browser creates a Document Object Model of the page
• The HTML DOM model is constructed as a tree of Objects
"HTML" is designed for marking up text by adding tags such as `<p>` to create HTML elements.

Example image:
HTML - DOM

• With the object model, **JavaScript** can create **dynamic** HTML by manipulating the objects:
  • JavaScript can change all the HTML elements in the page
  • Change all the HTML attributes in the page
  • Change all the CSS styles
  • Remove existing HTML elements and attributes
  • Add new HTML elements and attributes
  • React to all existing HTML events in the page
  • Create new HTML events in the page
CSS - Cascading Style Sheets

- CSS describes how HTML elements are to be displayed on screen
- CSS saves a lot of work
  - It can control the appearance of multiple elements and web pages all at once
CSS

```html
<style>
  body {
    background-color: black;
  }

  h1 {
    color: white;
    text-align: center;
  }

  p {
    color: white;
    font-family: verdana;
    font-size: 20px;
  }

  img {
    width: 200px;
    border-radius: 50%;
  }
</style>
```

HTML Basics

HTML is designed for marking up text by adding tags such as `<p>` to create HTML elements.

Example image:

[The Ohio State University logo]
CSS - Box Model

- Margin
- Border
- Padding
- Content
CSS - Box Model

- Margin
- 20px 40px 60px

Box Model

Content.

Box Model

Content.

Box Model

Content.
CSS - Box Model

- Border
- 10px
- 15px
- 20px
CSS - Box Model

- Border style
  - solid
  - dotted
  - dashed

- Other styles
  - double, groove, ridge, insert, outset, none, hidden
CSS - Box Model

- Padding
- 20px
- 15px
- 20px

**Box Model**

- Content.

**Box Model**

- Content.

**Box Model**

- Content.
SVG - Scalable Vector Graphics

• SVG defines vector-based graphics for the Web
• svg HTML tag
  • `<svg width="500" height="50"> </svg>`
    • Create a SVG canvas with 500px width and 50px height
• svg coordinates system

![SVG Coordinates Diagram](image)
SVG - Shapes

```xml
<svg>
  <line x1="5" y1="100" x2="5" y2="30" stroke="black"/>
  <line x1="100" x2="150" y1="220" y2="40" stroke="black"/>

  <rect x="150" y="150" width="15" height="15"/>
  <rect x="220" y="90" width="15" height="15"/>
  <rect x="110" y="90" width="15" height="15"/>
  <rect x="220" y="90" width="15" height="15"/>
  <rect x="50" y="130" width="15" height="15"/>
  <rect x="250" y="130" width="15" height="15"/>

  <circle cx="250" cy="25" r="7"/>
  <circle cx="150" cy="75" r="7"/>
  <circle cx="80" cy="85" r="7"/>
  <circle cx="110" cy="35" r="7"/>
  <circle cx="50" cy="75" r="7"/>
  <ellipse cx="180" cy="30" rx="15" ry="25"/>

  <text x="160" y="120">Hello!</text>
</svg>
```
SVG - Shapes + CSS

```html
<style>
  rect{
    fill: green;
  }

  circle{
    stroke: orange;
    stroke-width: 3;
  }
</style>
```
SVG - Polygon and Polyline

• Use coordinates to specify path

```xml
<svg>
  <polygon style="fill: purple; stroke: black;"
    points="48,16 16,96 96,48 0,48 80,96" />

  <polyline fill="none" stroke="blue" stroke-width="2"
    points="05,130 15,130 15,120 25,120 25,110 35,110" />
</svg>
```
SVG - PATH

• M x y – Move to (x,y)
  • m dx dy – Move by (dx,dy)

• L x y – Line to (x,y)
  • l dx dy

• H x, V y – draw horizontal and vertical lines
  • h dx, v dy

• Z, z close path

• Curve commands (Beziers Curves and Arcs)
<svg width="1000" height="1000">
  <path d="M 100 500
  H 400
  l 100 -200
  Z
  m 500 100
  l 0 200
  L 800 800
  l 0 -200
  Z"
  fill="orange" stroke="black"/>
</svg>
SVG - Transform

- translate($dx, dy$)
  - move a shape by ($dx, dy$)

```xml
<text x="20" y="20">Hello</text>
<text x="60" y="20">World!</text>
```

Hello World!

```xml
<text x="60" y="20" transform="translate(10, 10)">World!</text>
```

Hello World!
SVG - Transform

• \( \text{rotate}(a, x, y) \)
  • rotate a shape by \( a \) degrees about a given point \((x, y)\)
SVG - Transform

- scale($x$, $y$)
  - scales both the shape’s size and its coordinates

Hello

World!
(60, 20)

Hello

World!
(60*2=120, 20*3=60)
SVG - Transform

• Multiple functions

Transform in the reverse order, i.e. the order of rotate, translate, and scale
SVG - Group + Transform

• Group multiple shapes
  • `<g>` tag

Hello World!

Hello World!
SVG - No Layer

• Any pixel-paint applied later obscure any earlier paint and therefore appear to be "in front"

```xml
<svg>
  <rect x="0" y="0" width="30" height="30" fill="purple"/>
  <rect x="20" y="5" width="30" height="30" fill="blue"/>
  <rect x="40" y="10" width="30" height="30" fill="green"/>
  <rect x="60" y="15" width="30" height="30" fill="yellow"/>
  <rect x="80" y="20" width="30" height="30" fill="red"/>
</svg>
```
JavaScript

• JavaScript works with HTML and CSS
  • HTML to define the content of web pages
  • CSS to specify the appearance of web pages
  • JavaScript to program the behavior of web pages

• JavaScript is the programming language with C/C++ style syntax
  • *for, while, continue, break, if/else, switch* are similar to C/C++
  • *operators (+,-,\*,/,%)* are also similar (except ==,!=,///)
JavaScript - Hello, Console

- Easy and quick way to test JavaScript code and debug
  - The result of “console.log()” will appear here
- You can type JavaScript code directly into your browser in a web page
  - The console accepts one line of code at a time
- Open Console
  - Chrome
    - Select View -> Developer -> JavaScript Console
  - Firefox
    - Tools -> Web Developer -> Web Console
  - Safari
    - Safari -> Preferences -> Advanced -> Show Develop menu in menu bar
    - Develop -> Show JavaScript Console
JavaScript - Hello, Console

• An example using Chrome Console
  • Line 1: var x = 3;
    • Assign value 3 to the variable x
    • The value of the statement “var x = 3;” is undefined
  • Line 2: x + 1
    • The Console evaluates the value of “x + 1”, which is 4
  • Line 3: console.log(x+1)
    • Print the value of “x+1”, which is 4
    • The value of the statement “console.log(x+1)” is undefined
  • Line 4: (function xplusone() { return x+1; })() 
    • Define a function to compute x+1 and then, execute the function
JavaScript - Data Types

• Numbers
  • 42, 3.1415926

• Logical
  • true, false

• Strings
  • "Hello", 'Hello'

• null

• undefined*
  • Yes. undefined is not null!
  • Usually to indicate a variable is not defined
JavaScript - Data Types

• functions
  • function(x) { return x+1; }
  • Can be assigned to variables like: var xPlusOne = function(x) { return x+1; }
    • Same as: function xPlusOne(x) { return x+1; }

• Objects
  • An object in JavaScript is an associative array of property names (Strings) and values (Objects)
  • {from: “Tom”, to: “Jerry”, message: “We are good friends!” }

• Arrays
  • var numbers = [ 5, 10, 15, 20, 25];
  • var mixedValues = [ 1,3, 4.5, 5.6, "string", null, undefined, true ];
JavaScript - Data Types

• Javascript uses dynamic typing
• var x = “The answer is ” + 42;
  • The value of x is the string “The answer is 42”
• var x = “37” - 7;
  • The value of x is the number 30
• var x = “37” + 7;
  • The value of x is the string ”377”
• var x = “1.1” + “1.1”;
  • String “1.11.1”
• var x = (+“1.1”)
  • Number 1.1
• var x = (+“1.1”) + (+“1.1”);
  • Number 2.2
JavaScript - Control Flow

• C-Style `for`, `while`, `continue`, `break`, `if`/`else`, `switch/case`

```javascript
for (var i=0; i < 10; i++) {
  if (condition) {
    statement_1_runs_if_condition_is_true();
    break;
  }
  else {
    statement_2_runs_if_condition_is_false();
    continue;
  }
}
```
JavaScript - Manipulating DOM

• As mentioned, with the HTML DOM, JavaScript can access and change all the elements of an HTML document.

• But, the JavaScript APIs for DOM are complex
  • Link of JavaScript DOM methods
    • https://www.w3schools.com/js/js_htmldom.asp
  • We will learn how to use D3.js to manipulate DOM in a simple way
D3.js

• A JavaScript library
• Support visualizing data with the aid of HTML, SVG, and CSS
D3.js - Downloading and Referencing D3

• Downloading
  • Official website: https://d3js.org/

• Referencing

```html
<!DOCTYPE html>
<html>
  <head>
    <title>D3</title>
    <script type="text/javascript" src="d3/d3.min.js"></script>
  </head>
  <body>
    <script type="text/javascript">
      // Your beautiful D3 code will go here
    </script>
  </body>
</html>
```

• Referencing without downloading

```html
<script src="https://d3js.org/d3.v4.min.js"></script>
```
D3.js - Open Web Pages with D3.js

• Usually, you can view local HTML files directly in your web browser

• However, some browsers have restrictions that prevent them from loading local files via JavaScript, for security reasons
  • That means if your D3 code is trying to pull in any external data files (like CSVs or JSONs), it will fail with no good explanation
  • For this reason, it is much more reliable to load your page via a web server
  • To set up a simple local server
    • See D3 example to test your browser of our course website: http://web.cse.ohio-state.edu/~shen.94/5544/
Good Resources

• **W3School: Tutorial, Manual**
  - HTML: https://www.w3schools.com/html/default.asp
  - CSS: https://www.w3schools.com/css/default.asp
  - SVG: https://www.w3schools.com/graphics/svg_intro.asp
  - JavaScript: https://www.w3schools.com/js/default.asp

• **MDN web docs: Tutorial, Manual**
  - MDN web docs also have tutorials and manuals for HTML, CSS, and JavaScript

• **D3.js: https://d3js.org/**
Recommended Book

Interactive Data Visualization for the Web

Scott Murray