

Data Visualization (CSCI 627/490)

Web Programming

Dr. David Koop

JavaScript in one slide

- Interpreted and Dynamically-typed Programming Language
- Statements end with semi-colons, normal blocking with brackets
- Variables: `var a = 0; let b = 2; const c = 4;`
- Operators: `+, -, *, /, [ ]`
- Control Statements: `if (<expr>) {...} else {...}, switch`
- Loops: `for, while, do-while`
- Arrays: `var a = [1,2,3]; a[99] = 100; console.log(a.length);`
- Functions: `function myFunction(a,b) { return a + b; }`
- Objects: `var obj; obj.x = 3; obj.y = 5;`
 - Prototypes for instance functions
- Comments are `/* Comment */` or `// Single-line Comment`

Including JavaScript in HTML

- Use the script tag
- Can either inline JavaScript or load it from an external file
 - `<script type="text/javascript">`
 `a = 5, b = 8;`
 `c = a * b + b - a;`
 `</script>`
 `<script type="text/javascript" src="script.js"/>`
- Script tag can reference local or **remote** external javascript files
- The order the javascript is in is the order it is executed
- Example: in the above, `script.js` can access the variables `a`, `b`, and `c`

JavaScript Objects

- `var student = {name: "John Smith", id: "000012345", class: "Senior", hometown: "Peoria, IL, USA"};`
- Objects contain multiple values: key-value pairs called **properties**
- Accessing properties via dot-notation: `student.name`
- Always works via bracket-notation: `student["name"]`
- May also contain functions:
 - `var student = {firstName: "John",
 lastName: "Smith",
 fullName: function() { return this.firstName +
 " " + this.lastName; }};`
 - `student.fullName()`

Functional Programming

- Data: `var a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6];`
- How would I subtract one from each item?
 - `a.map(function(d) { return d-1; })`
- How would I find only the values `>= 10`?
 - `a.filter(function(d) { return d >= 10; })`
- How would I sum the array?
 - `a.reduce(function(s,d) { return s + d; })`
- How would I create a reversed version of the array?
 - `b = []; a.forEach(function(d) { b.unshift(d); });`
 - ...Or `a.reverse()` // modifies in place
- Arrow functions shorten such calls: `a.map(d => d-1);`
`a.filter(d => d >= 10); a.reduce((s,d) => s+d);`

Closures in JavaScript

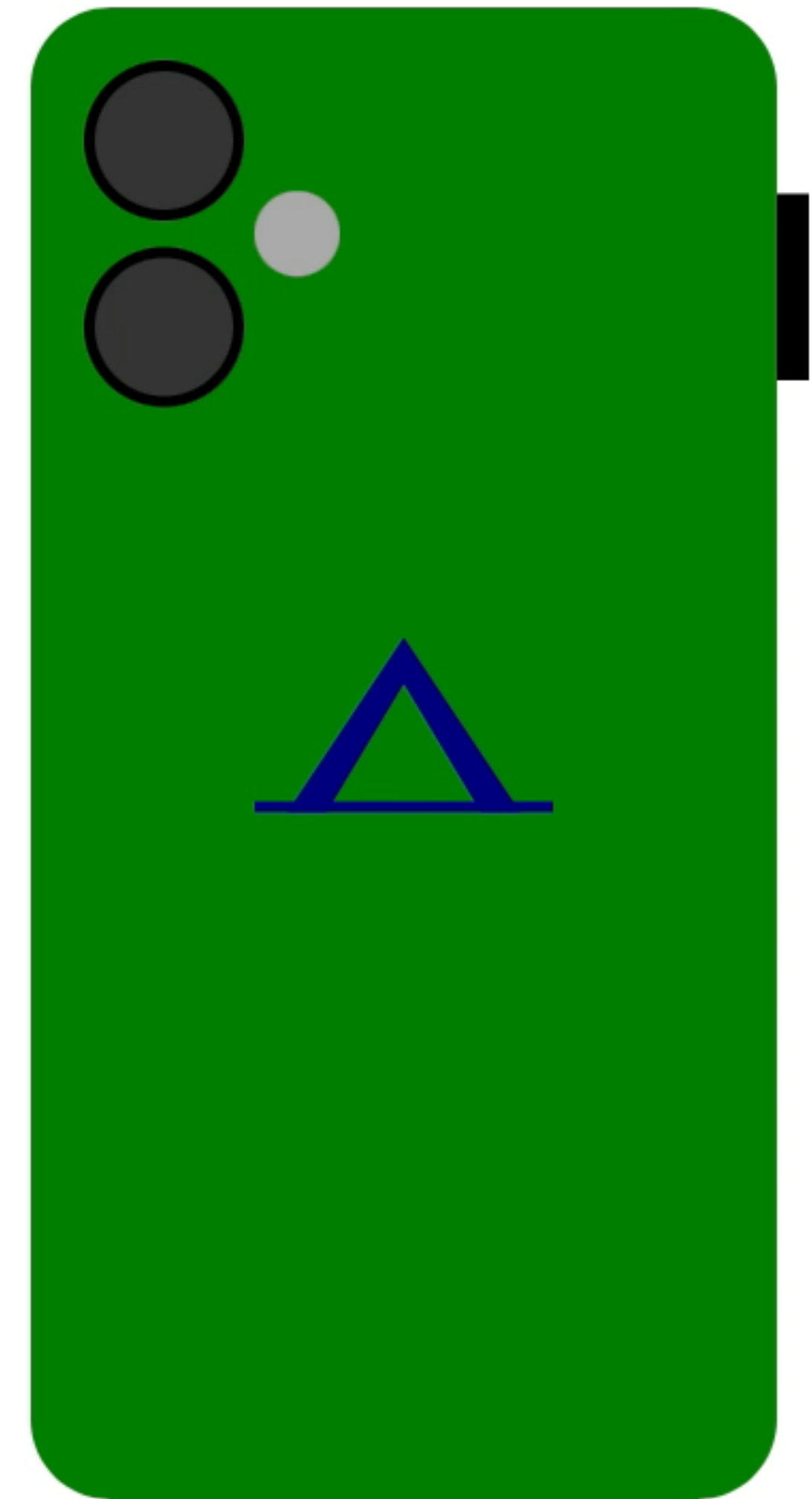
- Functions can return functions with some values set
- Allows assignment of some of the values
- Closures are functions that "remember their environments" [MDN]

```
function makeAdder(x) {  
    return function(y) {  
        return x + y;  
    };  
}  
var add5 = makeAdder(5);  
var add10 = makeAdder(10);  
  
console.log(add5(2)); // 7  
console.log(add10(2)); // 12
```

- Notebook

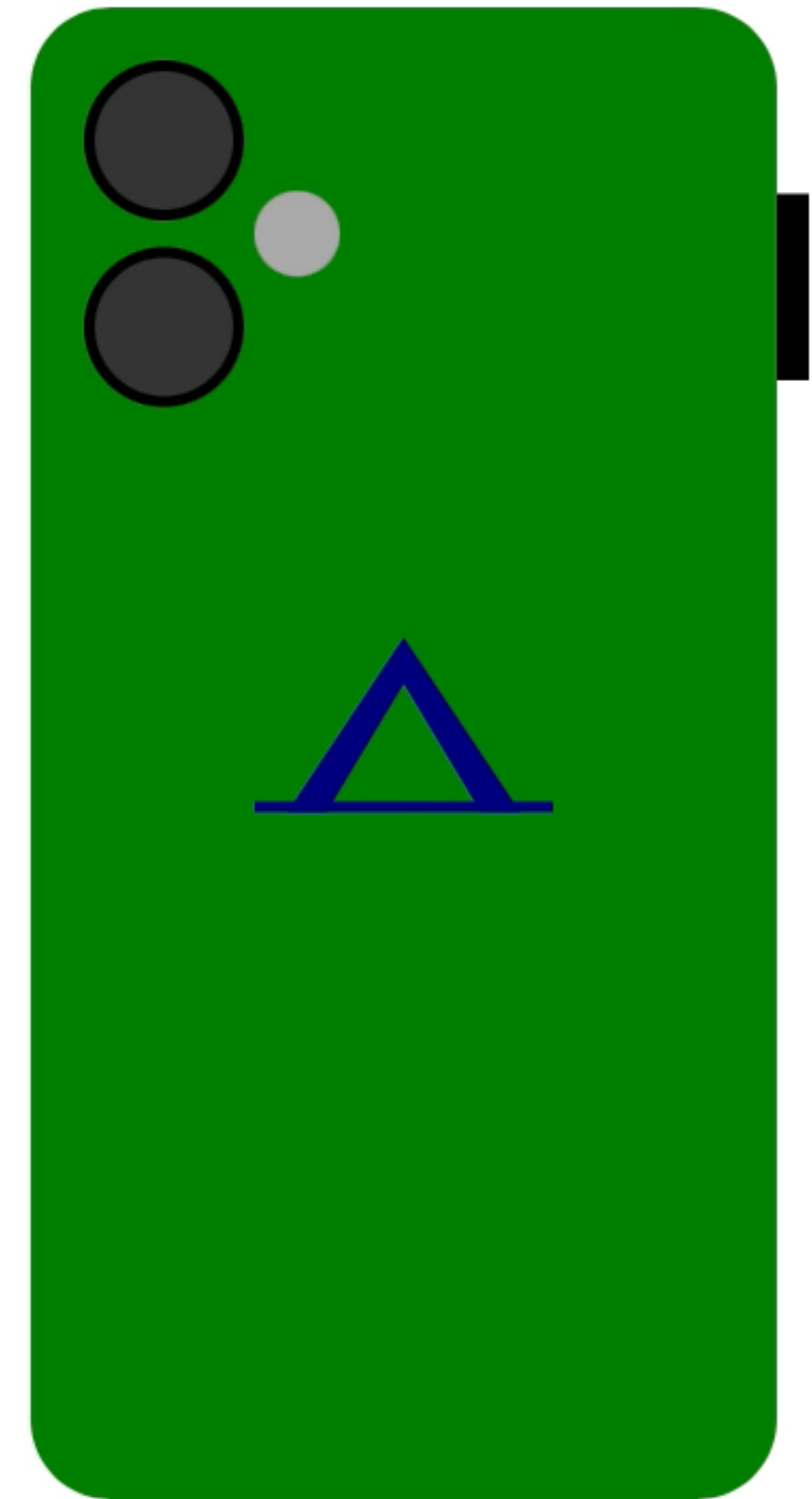
Assignment 1

- Write HTML, CSS, and SVG
- Text markup and styling (information)
- Drawing markup and styling (camera)
- Draw Bar chart using Plot library
- Due tonight
- Observable Sharing Process



Assignment 1

- Write HTML, CSS, and SVG
- Text markup and styling (information)
- Drawing markup and styling (camera)
- Draw Bar chart using Plot library
- Due tonight
- Observable Sharing Process



Women in STEM Spring Lecture Series

- Thursday, Feb. 19
- Lecture Series hosted in collaboration with the college and libraries
- All are welcome



**WOMEN IN
STEM**

Thursday, Feb. 19

Karen Samonds, Ph.D.
Professor of Biological Sciences at NIU
“Unearthing Stories: A woman’s journey through fossils, fieldwork, and family”

12:30 - 1:30 p.m.
Founders Memorial Library Staff Lounge
RSVP encouraged but not required.

This event is open to all NIU students, faculty, staff and community members. While we center on the experiences of women in STEM, people of all gender identities and expressions are welcome.



 NORTHERN ILLINOIS UNIVERSITY
Division of Academic Affairs
Office of the Executive Vice President and Provost

Hosted in collaboration with
College of Liberal Arts and Sciences and University Libraries.

This publication is approved for distribution by the NIU Division of Enrollment Management, Marketing and Communications. Display through 2-26-2026. Clearinghouse No. 21428403.

Spring Lecture Series

Function Chaining in JavaScript

- When programming functionally, it is useful to chain functions
- No intermediate variables!
- Often more readable code
- jQuery Example:
 - `$("#myElt").css("color", "blue").height(200).width(320)`
- Used a lot in Web programming, especially D3
- Can return the same object or a new object
- Lazy chaining keeps track of functions to be applied but will apply them later (e.g. when the page loads)

Manipulating the DOM with JavaScript

- Key global variables:
 - `window`: Global namespace
 - `document`: Current document
 - `document.getElementById(...)`: Get **one** element via its id
 - `document.querySelector(...)`: Get **one** element via selector
 - `document.querySelectorAll(...)`: Get **all** matching elements via selector
- HTML is parsed into an in-memory document (DOM)
- Can access and **modify** information stored in the DOM
- Can add information to the DOM

Example: JavaScript and the DOM

- Start with no real content, just divs:

```
<div id="firstSection"></div>  
<div id="secondSection"></div>  
<div id="finalSection"></div>
```

- Get existing elements:

- `document.querySelector/querySelectorAll`
- `document.getElementById`

- Programmatically add elements:

- `document.createElement`
- `document.createTextNode`
- `Element.appendChild`
- `Element.setAttribute`

Bears

Chicago, IL

2025-2026 NFC North Champions



What will happen next year?

Example (continued): Using Data to Build Content

Bears

Chicago, IL

2025-2026 NFC North

Champions

- September 8: Loss (24-27)
- September 14: Loss (21-52)
- September 21: Win (31-14)
- September 28: Win (25-24)
- October 13: Win (25-24)
- October 19: Win (26-14)
- October 26: Loss (16-30)
- November 2: Win (47-42)
- November 9: Win (24-20)
- November 16: Win (19-17)
- November 23: Win (31-28)
- November 28: Win (24-15)
- December 7: Loss (21-28)
- December 14: Win (31-3)
- December 20: Win (22-16)
- December 28: Loss (38-42)
- January 4: Loss (16-19)
- January 10: Win (31-27)
- January 18: Loss (17-20)

- We can loop through data to add content to a web page (schedule and results)
- Data:

```
[{"date": "September 8", "opponent": "Minnesota Vikings", "home": true, "win": false, "score": "24-27"}, ... ]
```
- Can use `forEach` to iterate through each game and build content
- Or, for...of loop:

```
for (const game of data)
```
- Notebook

Observable's HTML Templating

- Allows JavaScript expressions to be **inlined** in HTML (or SVG content)
- Use `${...}`
- Example:
 - [JavaScript] `name = "Prof. Koop"`
 - [HTML] `<p>Hello, my name is ${name}</p>`

Using Observable's HTML Templating

```
<div id="firstSection">
  <h1>Bears</h1><p>Chicago, IL</p>
</div>
<div id="secondSection">
  <h2>2018-2019 NFC North Champions</h2>
</div>
<div id="finalSection">
  ${scores.map( (game) => html`<p>${game.date} :
  ${game.win ? "Win" : "Loss"} (${game.score})</p>` ) }
  </img>
  <p>What will happen this year?</p>
</div>
```

Notebook

Creating SVG figures via JavaScript

- SVG elements can be accessed and modified just like HTML elements
- Create a new SVG programmatically and add it into a page:
 - ```
var divElt = document.getElementById("chart");
var svg = document.createElementNS(
 "http://www.w3.org/2000/svg", "svg");
divElt.appendChild(svg);
```
- You can assign attributes:
  - ```
svg.setAttribute("height", 400);  
svg.setAttribute("width", 600);  
svgCircle.setAttribute("r", 50);
```

Manipulating SVG via JavaScript

- SVG can be navigated just like the DOM
- Example:

```
function addEltToSVG(svg, name, attrs) {  
    var element = document.createElementNS(  
        "http://www.w3.org/2000/svg", name);  
    if (attrs === undefined) attrs = {};  
    for (var key in attrs) {  
        element.setAttribute(key, attrs[key]);  
    }  
    svg.appendChild(element);  
}  
  
mysvg = document.getElementById("mysvg");  
addEltToSVG(mysvg, "rect", {"x": 50, "y": 50,  
    "width": 40, "height": 40,  
    "fill": "blue"});
```

- Notebook

SVG Manipulation Example

- Draw a horizontal bar chart

- `var a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6];`

- Steps?

SVG Manipulation Example

- Draw a horizontal bar chart
 - `var a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6];`
- Steps:
 - Programmatically create SVG
 - Create individual rectangle for each item
- Notebook

...or Use Templating

- Same with SVG as with HTML
- Notebook