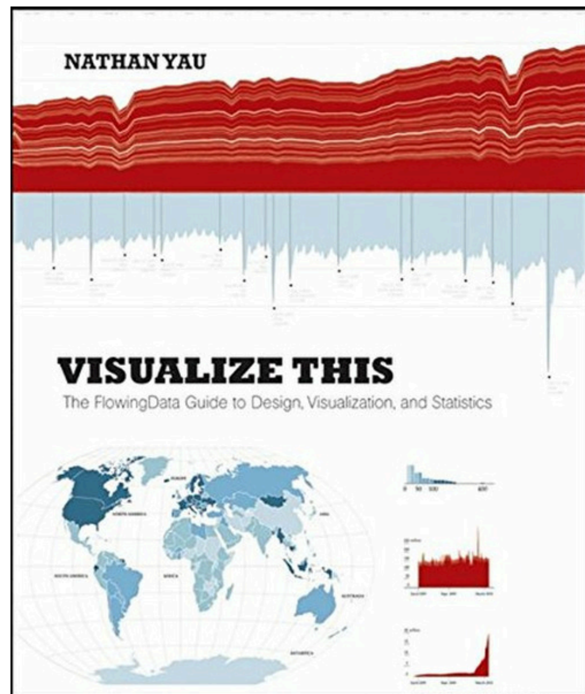
Data Science II

- Retrieving Data from the Web -

Prof. Dr. Eduard Kromer
Summer Semester 2024
University of Applied Sciences Landshut

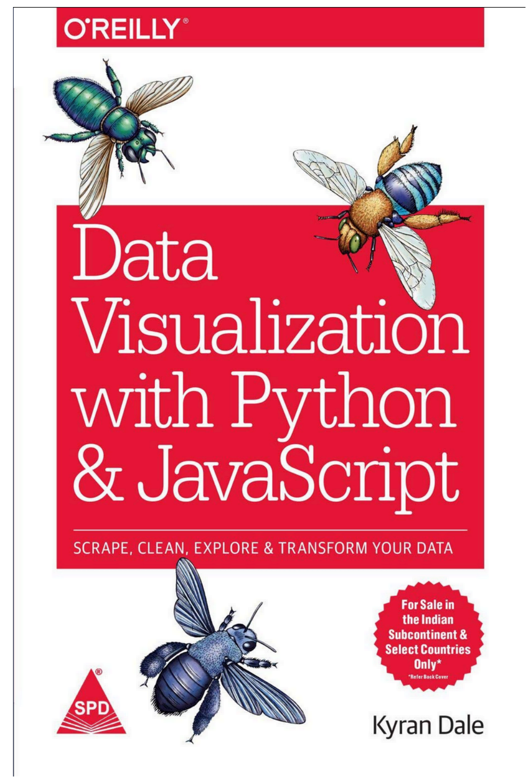
“Before you start working on the visual part of any visualization, you actually need data. The data is what makes a visualization interesting.”

–Nathan Yau - in Visualize This. The FlowingData Guide to Design, Visualization, and Statistics


NATHAN YAU
VISUALIZE THIS
The FlowingData Guide to Design, Visualization, and Statistics

“A fundamental part of the data visualizer’s skill set is getting the right dataset in as clean a form possible. And more often than not these days, this involves getting it off the Web”

–Kyran Dale in “Data Visualization with Python & JavaScript”, O’Reilly 2016


O'REILLY
Data Visualization with Python & JavaScript
SCRAPE, CLEAN, EXPLORE & TRANSFORM YOUR DATA
SPD
Kyran Dale

HTML + CSS

The Web

- the web: a collection of conversations between web servers and web clients
- clients contact servers with requests and servers respond with data
- web servers are internet-connected computers running server software (server web documents as requested)
- local servers run on the same computer you're working on (local: here, remote: somewhere else)

The Web

- web browsers, like Firefox or Chrome, are web clients
- every web page can be identified by its URL (Uniform Resource Locator) or URI (Uniform Resource Identifier):
- a complete URL consists of four parts:

    1. indication of the communication protocol, like HTTP or HTTPS
    2. the domain name of the resource, such as `haw-landshut.de`
    3. the port number (which port to use for the connection to the server)
    4. any additional locating information, such as the path to the requested file, or any query parameters

HTTP

- HTTP — Hypertext Transfer Protocol: the most common protocol for transferring web content from server to client
- HTTPS — the “S” stands for Secure — used when information should be encrypted in transit

The Process of a Website Visit

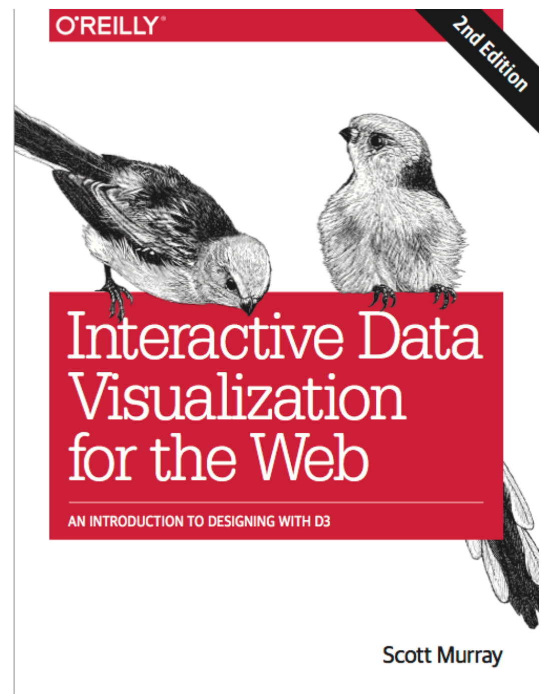
1. user runs browser and types in URL (if no protocol was specified, HTTP is assumed and “http://“ is prepended

2. browser attempts to contact the server behind the URL across the network, via port 80, the default port for HTTP

3. server acknowledges connection and taking requests

4. browser sends a request for the page that lives behind the URL the user typed in

5. server sends HTML content for that page


O'REILLY
2nd Edition
Interactive Data Visualization for the Web
AN INTRODUCTION TO DESIGNING WITH D3
Scott Murray

HOCHSCHULE LANDSHUT

The Process of a Website Visit

6. as client browser receives HTML, it discovers references to other files needed to assemble and display the entire page, including CSS stylesheets and image files and contacts the same server again, once per file

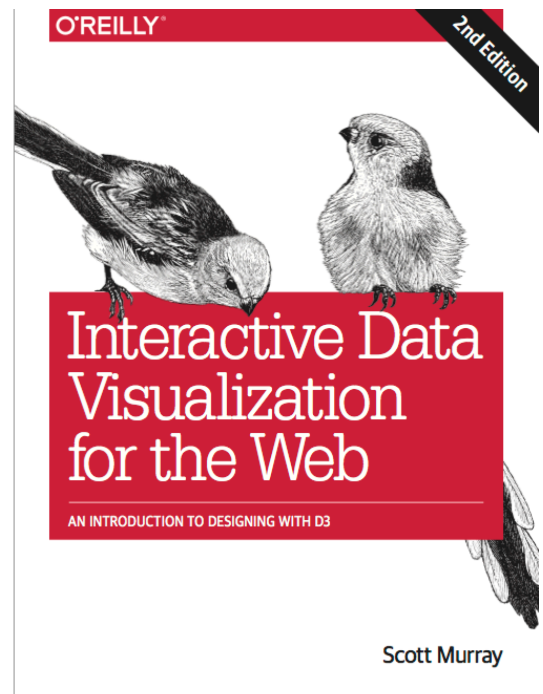
7. server responds, dispatching each file as needed

8. all web documents have been transferred over and the client renders the content

    - it first parses the HTML, to understand the structure of the content

    - then it reviews the CSS selectors, applying any properties to matched elements

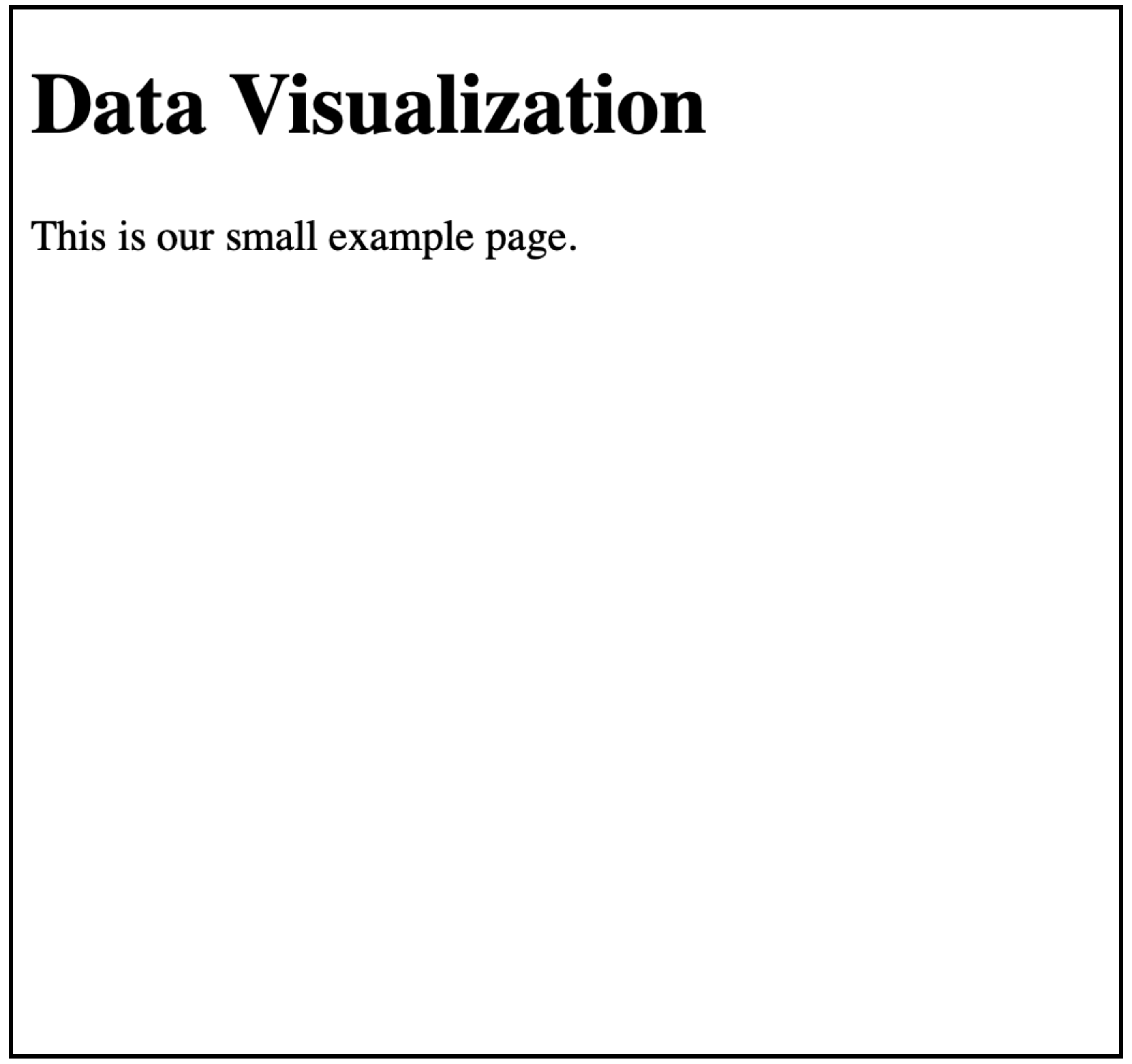
    - finally it plugs in any image files and executes any JavaScript code


O'REILLY
2nd Edition
Interactive Data Visualization for the Web
AN INTRODUCTION TO DESIGNING WITH D3
Scott Murray

HTML

- HTML — Hypertext Markup Language is used to structure content for web browsers
- HTML is stored in plain-text files with the *.html* suffix

```
<html>
  <head>
    <title>Data Science II</title>
  </head>
  <body>
    <h1>Data Visualization</h1>
    <p>This is our small example page.</p>
  </body>
</html>
```



Data Visualization

This is our small example page.

Core Function of HTML

- the core function of HTML is to enable you to “mark up” content
- you can provide raw text content with visual structure
- marking up: adding tags to create elements
  - tags start with `<` and end with `>`, as in `<p>` (paragraph of text)
  - tags usually occur in pairs; closing tags are indicated with a slash `</p>`

```
<p>This is our <em>small</em> example page.</p>
```

Core Function of HTML

- some tags never occur in pairs, such as the `img` element (references an image file)

```
<img src="photo.jpg">
```

- comments: `<!-- your comment here -->`

HTML Common Elements

- for the complete listing see the Mozilla Developer Network documentation
- `<!DOCTYPE html>`: standard document type declaration; first thing in the document
- `html`: surrounds all HTML content in a document
- `head`: document head (contains all metadata about the document, e.g. title and references to external stylesheets and scripts)
- `title`: title of the document; usually displayed at the top of the browser window
- `body`: primary visible content of the page; everything not in the head should go in the body
- `h1,h2,h3,h4`: headings of different levels

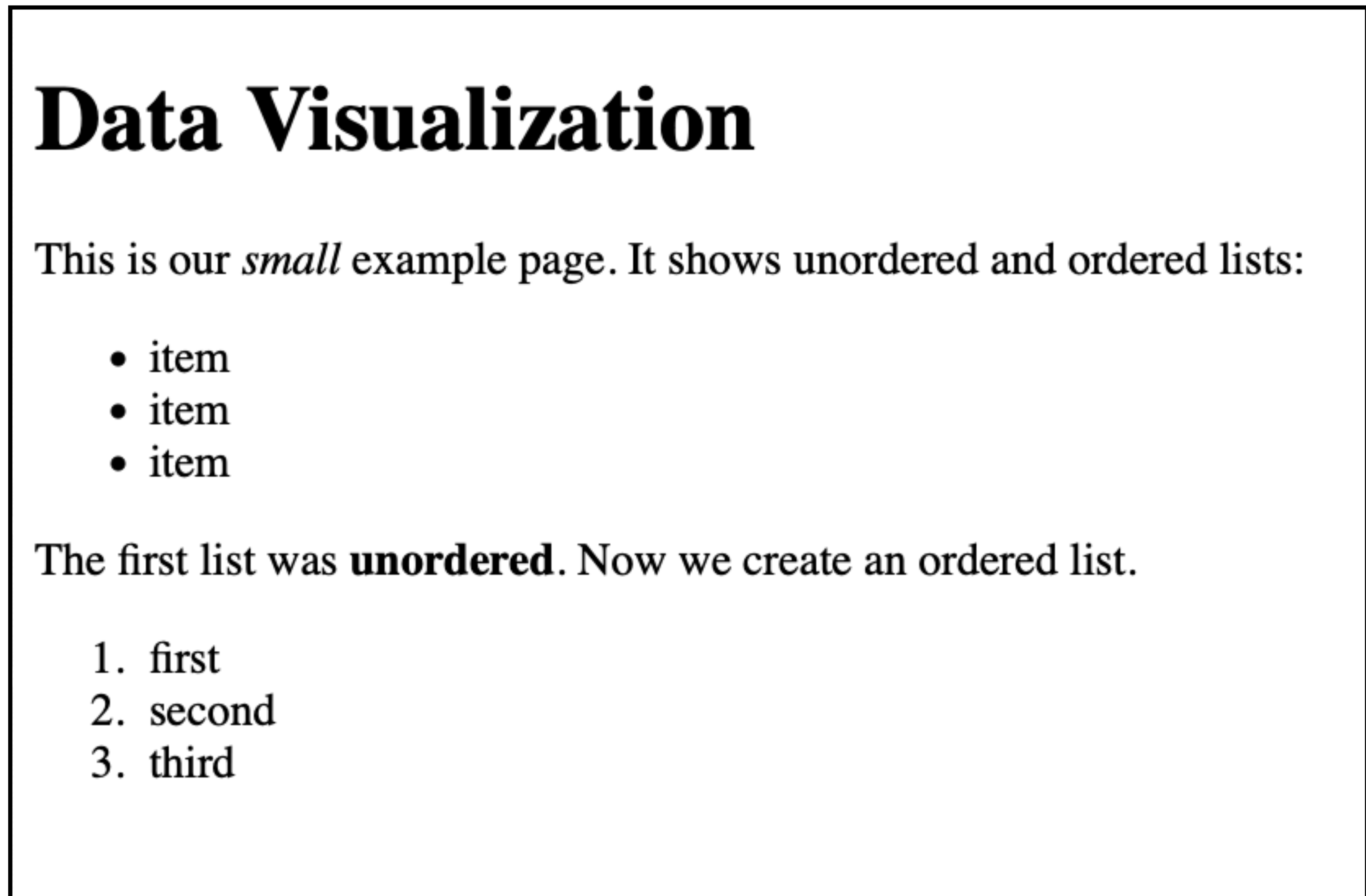
HTML Common Elements

- `p`: paragraph
- `ul, ol, li`: unordered lists (bulleted lists); ordered lists; both `ul` and `ol` should include `li` elements to specify list items
- `em`: emphasis; typically rendered in *italics*
- `strong`: additional emphasis; typically rendered in **boldface**
- `a`: a link; typically rendered as underlined, blue text
- `span`: an arbitrary span of text; typically within a larger containing element like `p`.
- `div`: an arbitrary division within the document; used for grouping and containing related elements

HOCHSCHULE LANDSHUT

HTML Common Elements

```
<html>
  <head>
    <title>Data Science II</title>
  </head>
  <body>
    <h1>Data Visualization</h1>
    <p>This is our <em>small</em> example page. It shows unordered and ordered lists:</p>
    <ul>
      <li>item</li>
      <li>item</li>
      <li>item</li>
    </ul>
    <p>The first list was <strong>unordered</strong>. Now we create an ordered list.</p>
    <ol>
      <li>first</li>
      <li>second</li>
      <li>third</li>
    </ol>
  </body>
</html>
```



Data Visualization

This is our *small* example page. It shows unordered and ordered lists:

- item
- item
- item

The first list was **unordered**. Now we create an ordered list.

1. first
2. second
3. third

Attributes

- we assign all HTML elements attributes by including property/value pairs in the opening tag

```
<tagname property="value"></tagname>
```

- different kinds of elements can be assigned different attributes

```
<a href="https://www.haw-landshut.de/">Hochschule Landshut</a>
```

Classes and IDs

- classes and IDs are very helpful as they can be referenced later to *identify specific pieces of content*
  - the CSS and JavaScript code will rely heavily on classes and IDs to identify elements
  - we can assign elements multiple classes by separating them with a space

```
<p class="great">This is a great paragraph.</p>
<p class="great">This is also a great paragraph.</p>
<p class="great brilliant">This is a great and brilliant paragraph.</p>
```

- IDs are used in the same way, but there can be only one ID per element and each ID value can be used only once on the page

```
<div id="content">
  <div id="visualization"></div>
  <div id="button"></div>
</div>
```

DOM

- DOM - Document Object Model: the hierarchical structure of HTML
- each pair of bracketed tags is an element and we refer to elements' relationships to each other in human terms: parent, child, sibling, ancestor, descendant
- web browsers parse the DOM to make sense of the page's content
- with our code we can navigate the DOM's hierarchy and apply styles and actions to its elements

```
<html>
  <body>
    <h1>Data Science</h1>
    <p></p>
  </body>
</html>
```

CSS

- CSS - Cascading Style Sheets are used to style the visual presentation of DOM elements
- CSS styles consist of selectors and properties
- you can apply the same properties to multiple selectors at once by separating the selectors with a comma

```css
body {
  background-color: whitesmoke;
  color: blueviolet;
}
```

```css
p,
li {
  font-size: 12px;
  line-height: 14px;
  color: orange;
}
```

Selectors

- selectors identify specific elements to which styles will be applied
- type selectors: they match DOM elements with the same name
    - `h1`, `p`, `strong`, `em`, `div`, …
- descendant selectors: match elements that are contained by another element
    - `h1 em` — selects em elements contained in an `h1`
    - `div p` — selects p elements contained in a `div`
- class selectors: match elements of any type that have been assigned a specific class
- id selectors: match the single element with a given ID

Referencing Styles

- embed the CSS in your HTML

```
<html>
  <head>
    <style type="text/css">
      p {
        font-size: 24px;
        font-weight: bold;
        background-color: red;
        color: white;
      }
    </style>
  </head>
  <body>
    <p>This an awesome red paragraph!!!</p>
  </body>
</html>
```

- reference an external stylesheet from the HTML

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <p>This is awesome orange text!!!</p>
  </body>
</html>
```

Referencing Styles

- we can also attach style rules inline directly to elements in the HTML
- you can do this by adding a style attribute to any element

```
<p style="color: blue; font-size: 24px; font-style: italic;">This is awesome blue text!!!</p>
```

Inheritance and Cascading

- many style properties are inherited by an element's decendants unless otherwise specified
- inheritance is a great feature of CSS, as children adopt the styles of their parents
- Why are they called Cascading Style Sheets?
  - selector matches cascade from top to down
    - when more than one selector applies to an element, the later rule generally overrides the earlier one

Specificity

- later rules generally override earlier ones, but not always

- the true logic has to do with the specificity of each selector

- if two selectors have the same specificity, then the later one will be applied

- advice: start with general selectors on top, and work your way down to more specific ones

```
p {
  color: blue;
}

p.highlight {
  color: black;
  background-color: yellow;
}
```

HTML & CSS Tutorials

- https://developer.mozilla.org/en-US/docs/Learn/HTML
- https://htmldog.com/guides/html/beginner/

Python Ecosystem
- requests & Beautiful Soup -

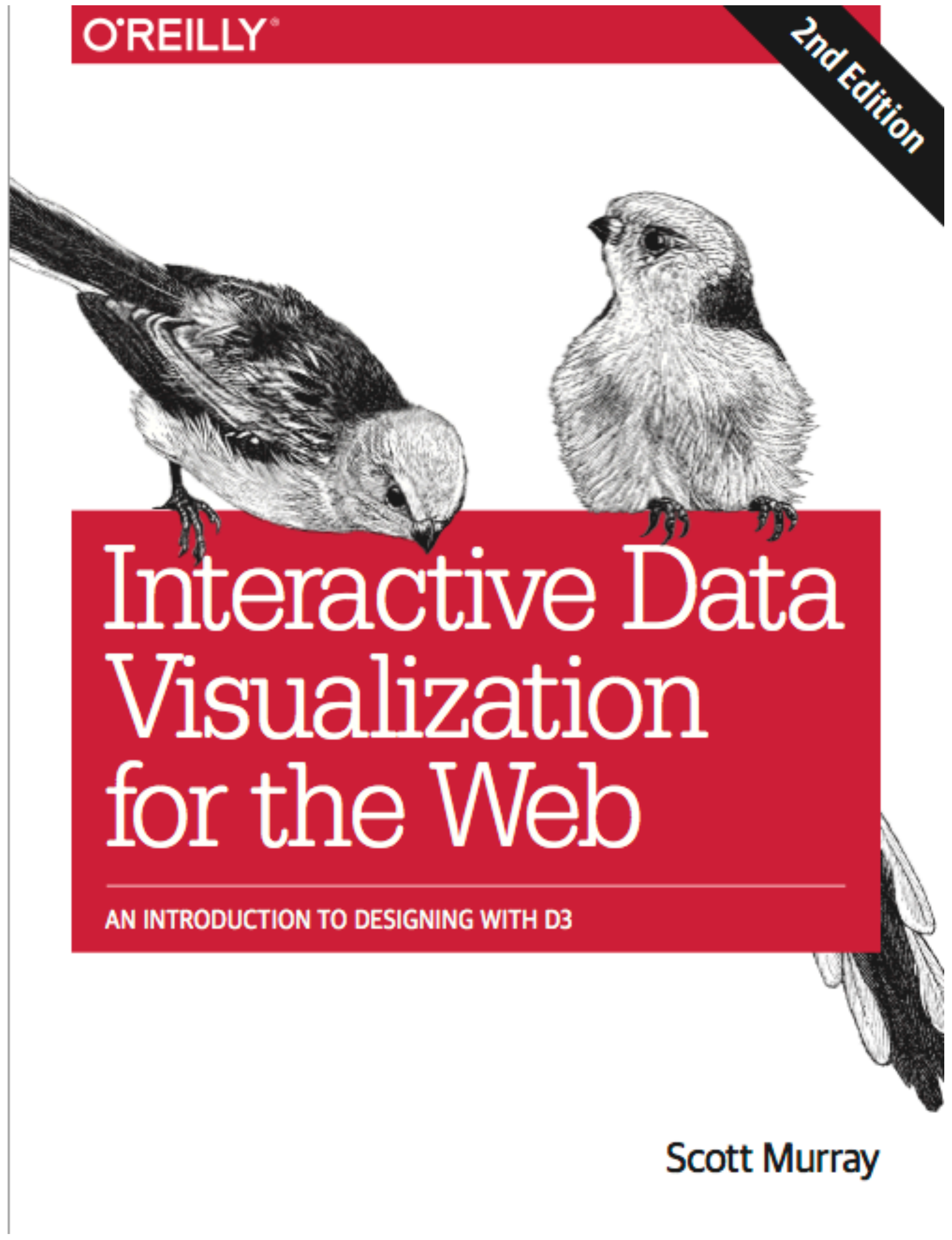
Web Scraping

- *web scraping* is the term for using a program to download and process content from the Web
- `requests`: Python library for downloading files and web pages from the internet
  - https://docs.python-requests.org/en/latest/
- `Beautiful Soup`: Python library for parsing HTML
  - mamba install -c anaconda beautifulsoup4
  - https://beautiful-soup-4.readthedocs.io/en/latest/
  - **read**: https://beautiful-soup-4.readthedocs.io/en/latest/#making-the-soup

Scraping Extreme Temperature Data with requests and Beautiful Soup

Jump to the Jupyter notebook `1_web_scraping.ipynb`

Literature



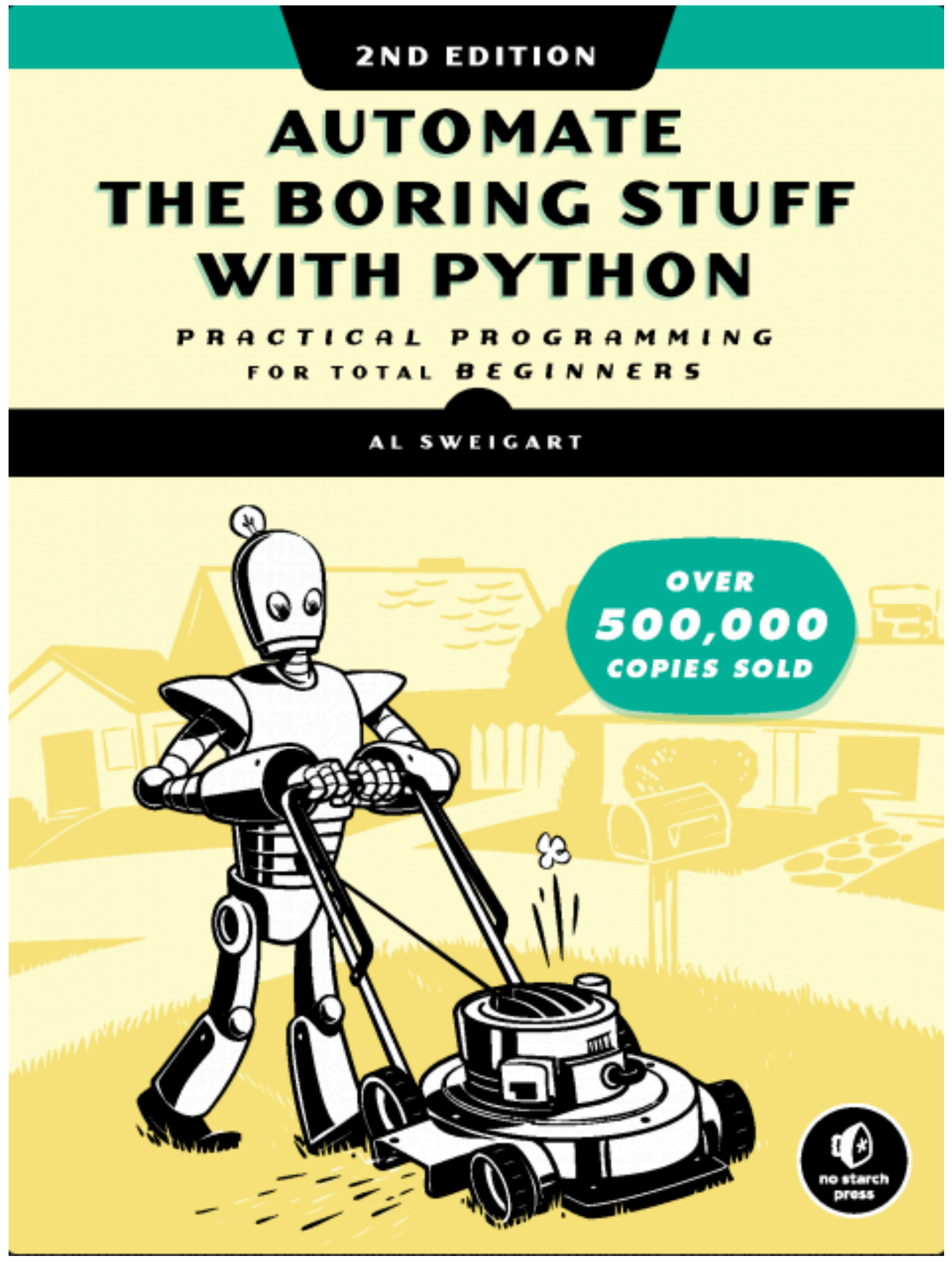
O'REILLY

2nd Edition

Interactive Data Visualization for the Web

AN INTRODUCTION TO DESIGNING WITH D3

Scott Murray



2ND EDITION

AUTOMATE THE BORING STUFF WITH PYTHON

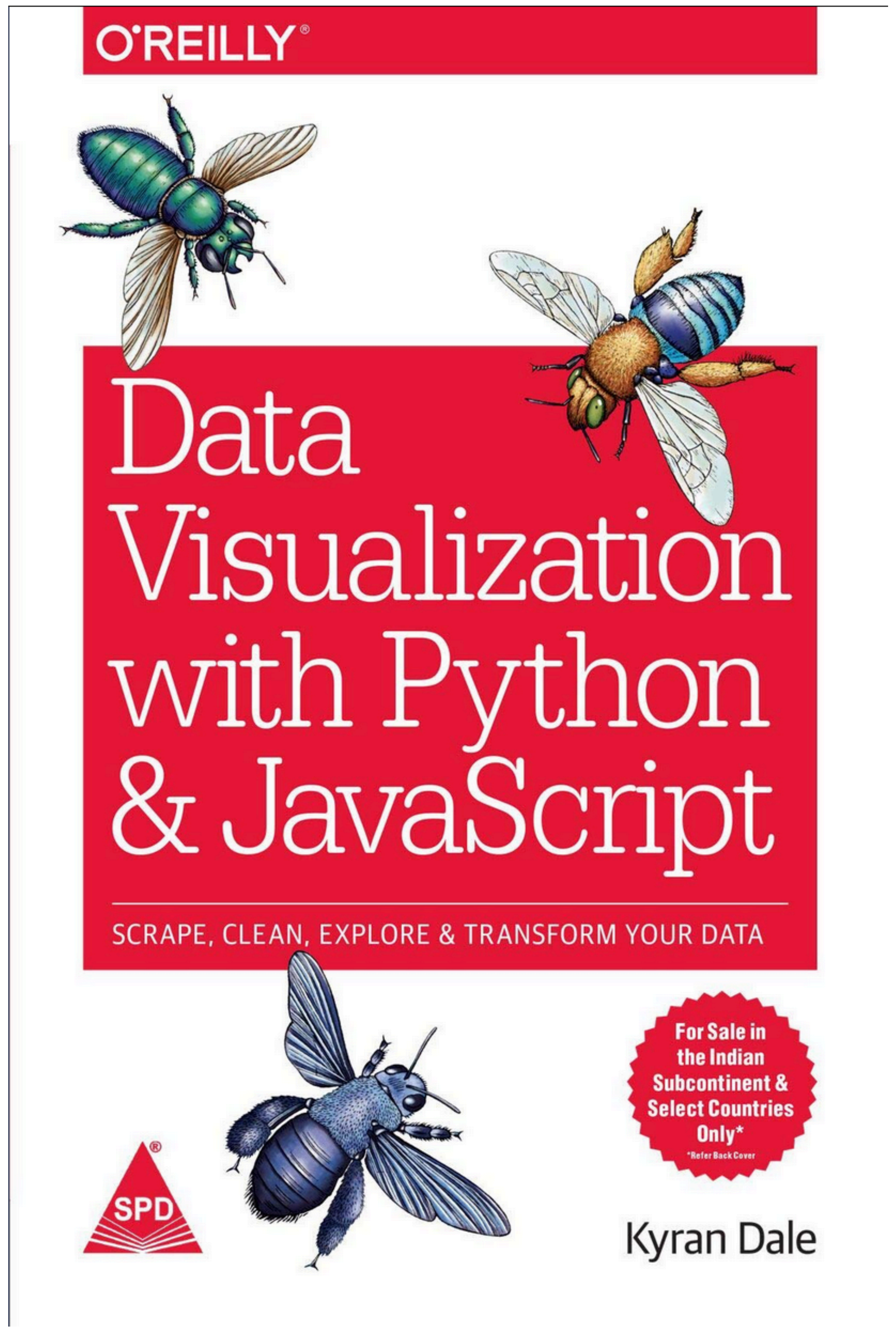
PRACTICAL PROGRAMMING FOR TOTAL BEGINNERS

AL SWEIGART

OVER 500,000 COPIES SOLD

no starch press

https://nostarch.com/automatestuff2



O'REILLY

Data Visualization with Python & JavaScript

SCRAPE, CLEAN, EXPLORE & TRANSFORM YOUR DATA

For Sale in the Indian Subcontinent & Select Countries Only*

SPD

Kyran Dale