Social Data Science

Core Technologies and Applications

Edgar J. Treischl

2024-10-17

Table of contents

Preface

This is work in progress. Ideas for the book:

Programming:

- Bash
- R
- Python

Data:

- SQL and other types of databases
- Webscraping
- APIs

Web:

- HTML
- CSS
- JavaScript

Products:

- Reports
- Apps
- Packages

Production:

- Version control: Git
- CI/CD: GitHub Actions
- Docker

1 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

```
1 + 1
```

```
[1] 2
```

2 HTML

This chapter introduces the basics of HTML. We start with a brief overview of HTML and discuss also some elements that can be used in an HTML document, including headings, paragraphs, lists, links, images, and tables. However, this chapter is not supposed to be a comprehensive guide to HTML. Instead it is meant to give you a basic understanding of HTML.

Why is a basis understand of HTML important for the Social Science? HTML is the language used to create web pages, but you will probably never be in the position to create a web page from scratch. However, you need a basic understanding if you want to collect data from the web, or if you want to create a web page using a content management system (CMS) like WordPress. In addition, you will need to know some HTML if you want to create a web-based survey, or if you want to create a web-based experiment. Finally, you will need to know some HTML if you want to create a web-based visualization.

This chapter is based on the book "HTML and CSS: Design and Build Websites" by Jon Duckett. This book is a great introduction to HTML and CSS, and I highly recommend it if you want to learn more about HTML and CSS.

2.1 The basics

HTML stands for HyperText Markup Language. HTML is the standard markup language for creating web pages. A markup language is a set of markup tags. HTML uses markup tags to describe web pages. HTML tags are surrounded by angle brackets, like this: `<html>`. HTML tags normally come in pairs like `<p>` and `</p>`. The first tag in a pair is the start tag, the second tag is the end tag. The end tag is written like the start tag, but with a forward slash inserted before the tag name.

All html documents must start with a `<!DOCTYPE html>` declaration. The HTML document itself begins with `<html>` and ends with `</html>`. Before the content of the HTML document, you will find the `<head>` element. The `<head>` element contains meta-information about the HTML document, like the title of the document, and links to CSS files. CSS stands for Cascading Style Sheets. CSS is used to define the style of the HTML document, which is why we will not bother how the document looks in this chapter. The header also includes links to JavaScript files, which are used to add interactivity to the HTML document. With JavaScript, you can create web-based games, web-based surveys, and web-based experiments.

The following code shows the structure a typical HTML document that we discussed, including links in the header to CSS and JavaScript files:

```
 <!-- A typical HTML document -->
<!DOCTYPE html>
<html>
<head>
  <title>Page Title</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
  <script src="script.js"></script>
</head>
<body>

  <!-- The visible part of the HTML document is between <body> and </body> -->

</body>
</html>
```

Thus, there is a lot going on in the header of an HTML document, but we will not bother with that in this chapter. The visible part of the HTML document is between `<body>` and `</body>`. The body of the HTML document contains all the content of the document, like text, images, links. To give the content of the HTML document some structure, we can use headings, paragraphs, lists, links. The `<h1>` tag defines the most important heading, the `<h2>` tag defines the second most important heading, and so on. The `<p>` tag defines a paragraph. The `<ul>` tag defines an unordered list, and the `<ol>` tag defines an ordered list. Each list item is defined with the `<li>` tag. The following code shows an example of headings, paragraphs, and lists:

```
<!-- Example of headings, paragraphs, and lists -->
<h1>This is title heading</h1>
<h2>This is another heading</h2>
<p>This is a paragraph.</p>
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ol>
```

In addition, we can also insert images or add a link in an HTML document. The `<img>` tag defines an image. The `<a>` tag defines a hyperlink to another website. The following code shows an example code:

```
<!-- Example of images and links -->
<img src="image.jpg" alt="Description of the image">
<a href="https://www.google.com">This is a link to Google</a>
```

Did you see that the img tag has an attribute called `src`? The `src` attribute specifies the URL of the image. The `alt` attribute provides an alternative text for an image. The `href` attribute specifies the URL of the page the link goes to. The text between the `<a>` and `</a>` tags will be displayed as a hyperlink. Thus, HTML tags can have attributes that provide additional information about the tag which we should keep in mind for the bigger picture.

Okay, we know how a HTML file is structured, and we know some elements that can be used in an HTML document. What else do we need to know for a basic understanding about HTML? Say we want to use an image instead of the text as a link? We can use the `<img>` tag inside the `<a>` tag. The following code shows an example:

```
<!-- Example of an image as a link -->
<a href="https://www.google.com">
  <img src="image.jpg" alt="Description of the image">
</a>
```

Finally, we need to discuss how we can harvest data from the web. Maybe, the data is provided as a table on a website. The `<table>` tag is used to create a table in an HTML document. The `<tr>` tag defines a row in a table. The `<td>` tag defines a cell in a table. The following code shows an example of a table:

```
<!-- Example of a table -->
<table>
  <tr>
    <td>Row 1, Cell 1</td>
    <td>Row 1, Cell 2</td>
  </tr>
  <tr>
    <td>Row 2, Cell 1</td>
    <td>Row 2, Cell 2</td>
  </tr>
</table>
```

How can we extract the data from the table? We can use a programming language like Python to extract the data from the table. We can use the `requests` library to download the HTML

document, and we can use the `BeautifulSoup` library to parse the HTML document. The following code shows an example of how we can extract the data from a table:

```
import requests
from bs4 import BeautifulSoup

url = 'https://www.example.com'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

tables = soup.find_all('table')

for table in tables:
  rows = table.find_all('tr')
  for row in rows:
    cells = row.find_all('td')
    for cell in cells:
      print(cell.text)
```

In R, we can use the `rvest` package to extract the data from the table. The following code shows an example of how we can extract the data from a table in R:

```
library(rvest)
```

```
Warning: package 'rvest' was built under R version 4.2.3
```

```
url <- 'https://edgar-treischl.github.io/PracticeR/articles/web_only/webscraping.html'
html <- read_html(url)
```

```
html |>
  html_element("table") |>
  html_table() |>
  head()
```

```
# A tibble: 6 x 11
  Country  `2011` `2012` `2013` `2014` `2015` `2016` `2017` `2018` `2019` `2020`
  <chr>     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 Austral~    5.1    5.2    5.7    6.1    6.1    5.7    5.6    5.3    5.2    6.5
2 Austria     4.6    4.9    5.3    5.6    5.7    6      5.5    4.8    4.5    5.4
3 Belgium     7.1    7.5    8.4    8.5    8.5    7.8    7.1    5.9    5.4    5.5
```

```
4 Canada      7.6    7.3    7.1    6.9    6.9    7.1    6.4    5.9    5.7    9.6
5 Chile       7.1    6.5    6.1    6.5    6.3    6.7    7      7.4    7.2   10.8
6 Colombia   10.9   10.4    9.7    9.2    9      9.3    9.4    9.7   10.5   16.1
```

2.2 CSS

CSS stands for Cascading Style Sheets. CSS is used to define the style of the HTML document. The style of the HTML document includes the layout, the colors, the fonts, and the spacing. CSS is used to separate the content of the HTML document from the style of the HTML document. Why do we seperate the content from the style? By separating the content from the style, we can change the style of the HTML document without changing the content. This is useful if we want to change the style of the HTML document for different devices, like a desktop computer, a tablet, or a smartphone. In addition, we only need to change the style in one place, which makes it easier to maintain the HTML document.

Let me give a simple example. Say we want to color all paragraphs red and all headings blue. Each time we insert a paragraph or a heading in the HTML document, we could specify the color with the `style` attribute. The next console shows how we can specify the color manually:

```
<!-- Example of inline sytle (CSS) -->
<h1 style="color: blue;">This is a heading</h1>
<p style="color: red;">This is a paragraph.</p>
<p>This is another paragraph.</p>
```

Did you realize that the last paragraph will not be displayed in red? Inline CSS is bothersome and definetly not the best way to specify the style of an HTML document. Instead, we can use the `<style>` tag to define the style of the entire HTML document. The `<style>` tag is placed in the `<head>` element. The following code shows an example of how we can color all paragraphs red and all headings blue:

```
<!-- Example of CSS -->
<!DOCTYPE html>
<html>
<head>
  <style>
    p {
      color: green;
    }

    p.important {
```

```
      color: red;
    }

    h1 {
      color: blue;
    }
  </style>
</head>
```

The CSS code is placed between the `<style>` and `</style>` tags. The CSS code is written in a selector and declaration block. The selector points to the HTML element you want to style. The declaration block contains one or more declarations separated by semicolons. Each declaration includes a CSS property name and a value, separated by a colon.

Did you see that I included a class in the CSS code? Paragraphs with the class `important` will be displayed in red, while all other paragraphs will be displayed in green. The class is defined with a dot in front of the class name. Thus, if the HTML code contains a paragraph with the class `important`, the paragraph will be displayed in red:

```
<!-- Example of CSS with class -->
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
  <p class="important">This is an important paragraph.</p>
  <p id="unique">Look, a unicorn.</p>
</body>
</html>
```

In addition to classes, we can also use the ID to style HTML elements. The ID is defined with a hash in front of the ID name. The ID is used to uniquely identify an element in the HTML document. The following code shows the corresponding CSS code to select an element with a specific ID:

```
<style>
  #unique {
      color: purple;
    }
</style>
```

In a real world application, we would not use the `<style>` tag to define the style of the HTML document. Instead, we would use an external CSS file which contains the style of the HTML document. Look out for link tag in the header of a HTML document. The link tag is used to link an external CSS file to the HTML document.

```html
<!-- Example of an external CSS file -->
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
```

Thus, look out for the CSS file if you want to change the style of a HTML document.

References

Knuth, Donald E. 1984. “Literate Programming.” *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.