CS171 - Lab 1

Learning Objectives

After completing this lab you will be able to:

- Use several web development tools (Webstorm, Chrome/Firefox developer tools and Web Inspector, and the browser integrated console)
- Set up and modify HTML documents
- Understand the difference between HTML and DOM
- Define CSS rules to style web pages (with CSS selectors)
- Include front-end frameworks like *Bootstrap*

Prerequisites

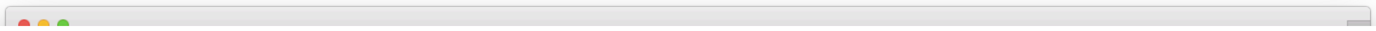
- You have installed a code editor such as *Webstorm* (https://www.jetbrains.com/webstorm/). The free classroom license can be found on Piazza.
- You have read Chapter 3 (pages 15-34) in *D3 - Interactive Data Visualization for the Web* by Scott Murray.
- You have successfully completed the pre-quiz for the first lab on Canvas.
- We encourage you to use Google Chrome or Mozilla Firefox as your primary web browser during all labs and homeworks.

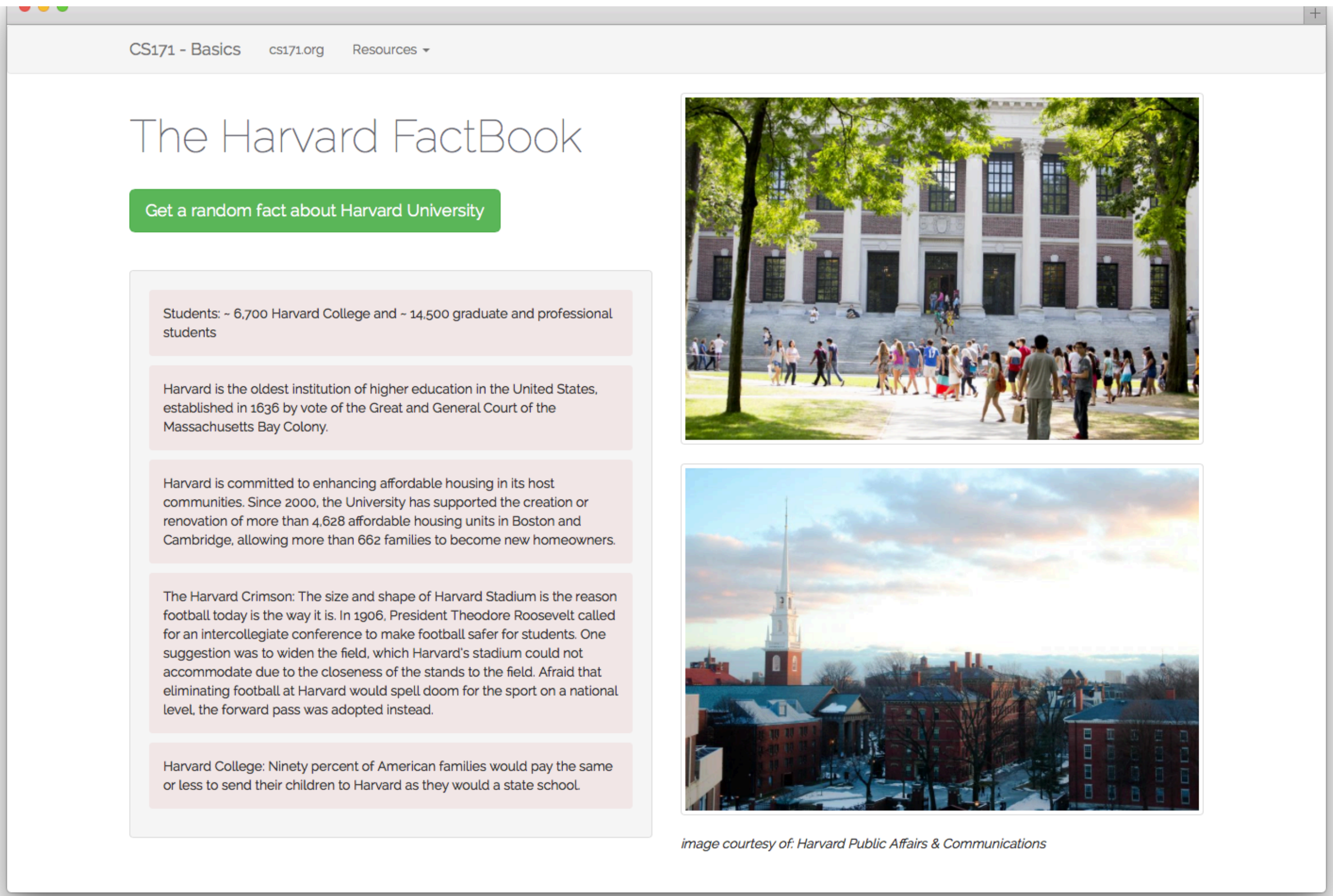
Fundamentals of Web Development

In this course we will use HTML, CSS, JavaScript and SVG to create interactive data visualizations. Before starting with extensive visualization projects and learning more about the JavaScript library D3 we will use the first two labs to get a fundamental understanding of these basic web technologies.

The next 90 minutes are split up into multiple theoretical and interactive sections. All the activities are consecutive and will result in a basic web page with random facts about Harvard. We will provide the facts and the interactive component. Your task is to set up the markup (HTML) and the style (CSS) of the website. After completion of this lab you will be well prepared for the homework and the following labs.

The result of the first lab may look like the following screenshot. Most of the design decisions are up to you - so feel free to be creative and come up with your own ideas.





The Harvard FactBook

Students: ~ 6,700 Harvard College and ~ 14,500 graduate and professional students

Harvard is the oldest institution of higher education in the United States, established in 1636 by vote of the Great and General Court of the Massachusetts Bay Colony.

Harvard is committed to enhancing affordable housing in its host communities. Since 2000, the University has supported the creation or renovation of more than 4,628 affordable housing units in Boston and Cambridge, allowing more than 662 families to become new homeowners.

The Harvard Crimson: The size and shape of Harvard Stadium is the reason football today is the way it is. In 1906, President Theodore Roosevelt called for an intercollegiate conference to make football safer for students. One suggestion was to widen the field, which Harvard's stadium could not accommodate due to the closeness of the stands to the field. Afraid that eliminating football at Harvard would spell doom for the sport on a national level, the forward pass was adopted instead.

Harvard College: Ninety percent of American families would pay the same or less to send their children to Harvard as they would a state school.

image courtesy of: Harvard Public Affairs & Communications

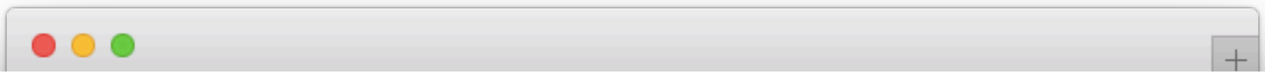
HTML (Hypertext Markup Language)

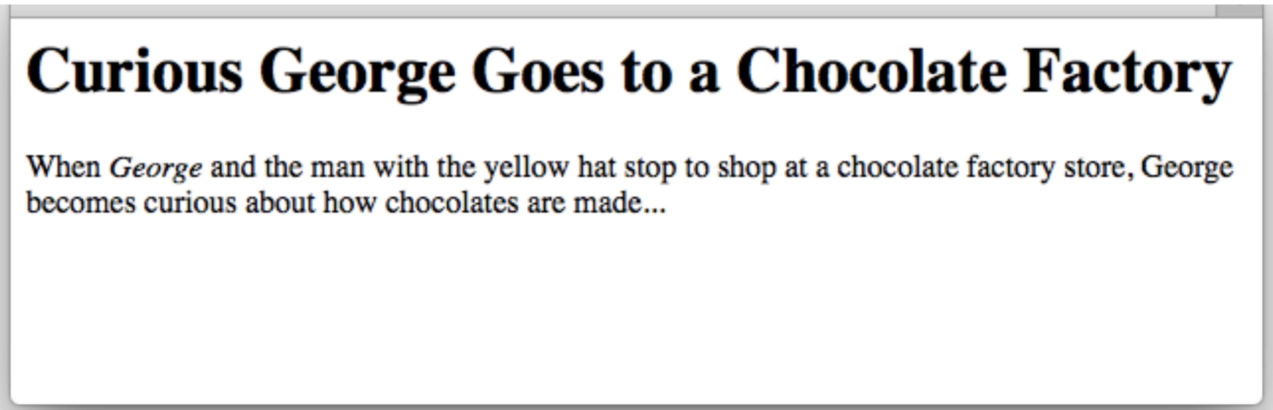
HTML is used to structure content for web browsers. It enables us to create a markup by adding additional elements (i.e., tags) to the content. Therefore we can differentiate, for example, between the headline and the body of a story.

A brief example:

```
<h1>Curious George Goes to a Chocolate Factory</h1>
<p>
    When <i>George</i> and the man with the yellow hat stop to shop at a chocolate
    factory store, George becomes curious about how chocolates are made...
</p>
```

Result:





Curious George Goes to a Chocolate Factory

When *George* and the man with the yellow hat stop to shop at a chocolate factory store, George becomes curious about how chocolates are made...

By surrounding a portion of content with a pair of tags we give the raw text a visual structure and make the information more accessible.

A comprehensive and well structured list of HTML elements can be found at MDN.

HTML Boilerplate

Every HTML5 document requires a little bit of boilerplate code that you should just copy and past every time you create a new file. A boilerplate is a piece of code that is usually copied with little or no alteration, much like a template, to speed up the creation of new files. In the case of HTML5 this includes several HTML tags (e.g. head, html, ...) that don't have visual equivalents on the website, but that are necessary to define the document's metadata.

You should be familiar with this structure:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
</body>
</html>
```

Classes and IDs

Classes and IDs are extremely useful attributes, as they can be referenced to identify specific pieces of content. Your CSS and JavaScript code will rely heavily on classes and IDs to identify elements.

(1) `<div>Interactive Data Visualization</div>`

This element has no attributes, so the only possible way to identify the element is by its tag name *div*. If there are multiple div-tags in one page, which happens frequently, selecting just the above div element becomes a problem.

(2) `<div id="book-123">The Value of Visualization</div>`

To solve this problem, we can give the element a unique ID: *book-123*. However, there can be only one ID per element and each ID value can be used only once per page!

(3) `<div class="book content">Visualization Analysis and Design</div>`

Any attribute or styling information that needs to be applied to multiple elements on a page should be done with a *class*. In the above example, we have assigned the div element to the class *book*, that allows us to select all html containers of the type *book*. At the same time, we have assigned the div element to the class *content*.

Elements can be assigned multiple classes, simply by separating them with a space.

Activity 1

1. **Create a new HTML file `basics.html` in your code editor**

   In Webstorm you will have to create a new project, and then add a new basics.html file to your project. You can then not only edit your file in Webstorm, but also view the html in a browser by running basics.html from within Webstorm (right-click on basics.html -> run). Internally, Webstorm will start its own web server for serving your basics.html page, which will become important once we start including D3 elements into our html pages.

2. **Copy the HTML boilerplate into your empty file**

3. **Add some structure to the *body* of your new document and try different HTML elements. The content should be appropriate for the HTML tags you are using (e.g., a headline vs. a paragraph).**

   Make sure to include at least:

   - A top-level headline
   - An empty div-container (will be filled with facts later)
   - A hyperlink to any other page
   - An image
   - A button (will trigger the search for a new fact)

   Open your file `basics.html` in a web browser to see the results.

4. **Add the following classes and IDs to the elements**

4. **Add the following classes and IDs to the elements**

    - Add the ID `content` to the div container
    - Add the ID `cs171-basics` to the button
    - Add the classes `btn` and `btn-primary` to the button

The DOM

The Document Object Model (DOM) is a programming interface for HTML, XML and SVG documents. It provides a hierarchical structured representation of the document (a tree) and it defines a way that the structure can be accessed from programs so that they can change the document structure, style and content. Or in other words, it is a model that the browser generates, when it parses the HTML document.

The difference between HTML and DOM should be more understandable after the following interactive exercise.

Activity 2

Web Developer Tools

Fortunately every modern-day web browser has built-in *developer tools* that expose the current state of the DOM and help us to better understand what is going on. In this exercise we will use the *Web Inspector* to view the DOM tree of our document.

1. **Add the following line at the bottom of your previously created `basics.html` (inside the `<body></body>`) to include an external JavaScript file that we prepared for this exercise.**

    ```
    <script src="http://www.cs171.org/2016/assets/scripts/dom-example.js"></script>
    ```

    We are providing a ready-made script which will listen to your button (*ID: cs171-basics*). It will automatically deliver random facts if you click on the button.

2. **Open *basics.html* in your web browser and navigate to *Developer Tools****

    We encourage you to use Google Chrome or Mozilla Firefox.

    - Chrome: View → Developer → Developer Tools
    - Firefox: Tools → Web Developer → Page Source

    Make sure to check out the keyboard shortcut of your system to open the developer tools!

3. **Inspecting the DOM with the *Web Inspector****

    In the default mode, the *Web Inspector* should be docked to the bottom of the window and split the page

In the default mode, the *Web Inspector* should be docked to the bottom of the window and split the page horizontally.

We can see something that looks like the source code of the HTML document that you wrote in your editor. Some tags are probably collapsed. Actually, you are **not** viewing the raw content of your HTML document. What you are seeing is the visual representation of the DOM tree (after all scripts have run and potentially modified the original html source)!

The HTML you write is parsed by the browser and turned into the DOM. In simple cases it will look like your raw HTML, but if any JavaScript code has been executed, the current DOM may be vastly different.

4. **Update the DOM: Click on the previously created button!**

   Every time you click on the button, a function in the external JavaScript file that we provided for you will be triggered that adds a new paragraph (random fact) to the DOM tree. You can see the new elements in the browser window and the current state of the DOM in the *Web Inspector*.

   Your `basics.html` remains unchanged. We have only modified the DOM with JavaScript, not the actual document. Your modifications will be discarded if you reload the page.

5. **Update the DOM: Delete nodes from the DOM tree**

   You can also use the *Web Inspector* to modify your DOM directly. You can edit the content, add attributes or delete nodes. Try it out!

   The developer tools of modern browsers usually include many other tools to make the life of a web developer easier. In the next activity we will use the Web Inspector to modify CSS and next week we will use the JavaScript Console for debugging.

Cascading Style Sheets (CSS) - *Making things pretty!*

With HTML you define the structure and content of the page and with CSS you set its style - things like fonts, colors, margins, backgrounds etc.

A stylesheet will usually consist of a list of CSS rules that are inserted in a `<style>` block in your HTML header or more often stored in an external file and included via the below line of code. Make sure to include an external style sheet also in the HTML header (inside the `<head></head>` elements of your html file).

```
<link rel="stylesheet" href="css/style.css">
```

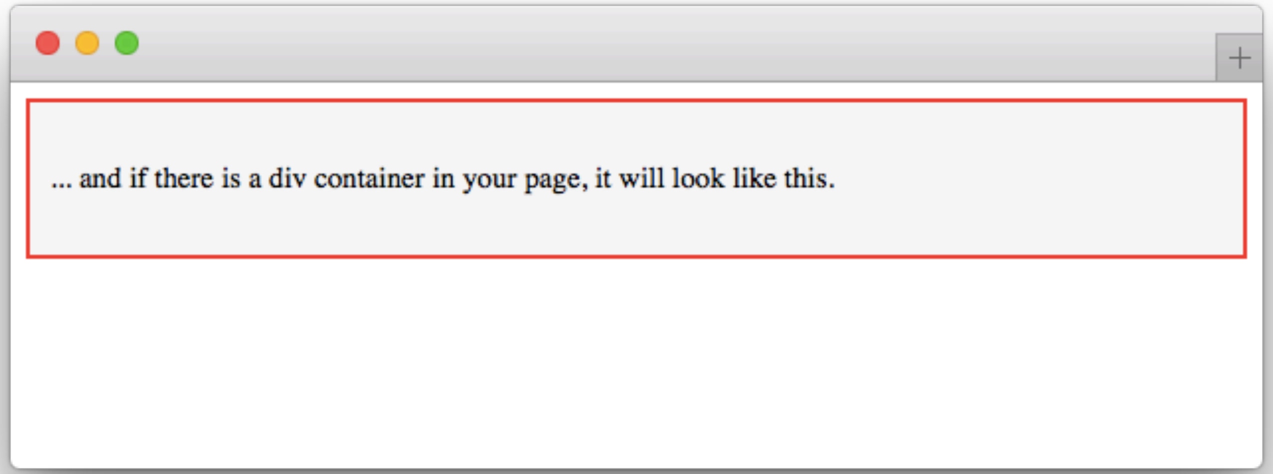
... This assumes that you have a separate file `style.css` in the folder `css`.

CSS styles consist of selectors and properties. Selectors are followed by properties, grouped in curly brackets. A property and its value are separated by a colon, and the line is terminated with a semicolon.

property and its value are separated by a colon, and the line is terminated with a semicolon.

A simple rule in CSS can look like the following:

```
div {
    font-size: 15px;
    padding: 30px 10px;
    background-color: #F7F7F7;
    color: black;
    border: 2px solid red;
}
```


... and if there is a div container in your page, it will look like this.

Similar to the introduction to HTML we are providing only a few examples. If you are searching for specific style properties in the future we recommend [Mozilla's Developer Platform] or [w3schools.com]. Some CSS properties are needed quite often, so you will memorize them quickly.

In the above example we have assigned our CSS rule to all div containers but if we want to style specific elements we can use the selectors *id* and *class*.

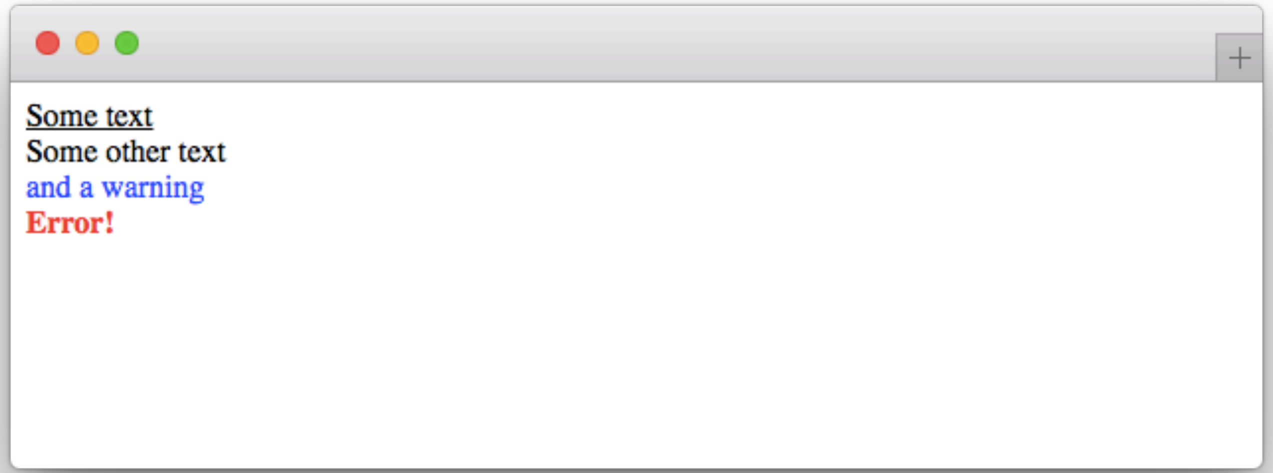
As you can see in the example below, IDs are preceded with a hash mark (CSS style 1) and class names are preceded with a period (CSS style 2). You can also use descendant selectors to address nested tags (CSS style 3).

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <title>Simple CSS</title>
    <style>
        #article-1 {
            text-decoration: underline;
        }
        .error {
            font-weight: bold;
            color: red;
        }
        .article .warning {
          color: blue;
        }
    </style>
</head>
<body>

<div class="article" id="article-1">Some text</div>

<div class="article">
    Some other text
    <div class="warning">and a warning</div>
</div>

<div class="article error">Error!</div>

</body>
</html>
```

Result:



Some text
Some other text
and a warning
Error!

Activity 3

In this activity you will use CSS to add custom styles to your HTML file.

1. **Create an external CSS file `style.css` and include it in your HTML document `basics.html`**

2. **Add several CSS rules to your stylesheet. You can choose the design parameters freely (i.e., decisions about fonts, colors or scales are up to you) but make sure to include at least:**

    - *ID Selector* (e.g. *#content*)
    - *Class Selector*
    - *Descendant Selector* for the dynamically created Harvard facts (paragraphs)
    - Custom *Hover-Effect* for hyperlinks
    - *Padding* and *Margin* properties
    - *Color* or *Background-Color* properties

    You can play around with different CSS parameters, but don't worry too much right now about making your webpage look beautiful, we will come back to the design at the end of the lab.

3. **Inspecting the CSS with the *Web Inspector****

    If you are working on a more sophisticated problem it can be very useful to analyze your CSS rules with the *Web Inspector*. The CSS styles in the right panel match the currently selected DOM element.

    - Click on different lines in the *Elements*-panel to see the respective CSS properties. The rules are collected from inline styles, attached stylesheets and user agent stylesheets. User agent stylesheets are the browser's default properties, such as font-size or margin.

    - Rules that are specified later in a CSS file generally override rules that were specified earlier in the file, but not always. The true logic has to do with the specificity of each selector. The *div.content* selector would override the *div* rule even if it were listed first, simply because it is a more specific selector.

      The order of the CSS rules in the right panel helps you to identify the importance of the individual styles.

    - Similar to the DOM tree, you can also modify, add and remove CSS rules and properties. This is a quick and easy way to try different styles directly in the browser (debugging), but keep in mind that the changes will be discarded if you reload the page. Try it out and modify your CSS in the browser!

    - Maybe you have already recognized it. If you include a specific color in your CSS properties the *Web Inspector* shows you a small button, linked to a color picker. This little tool can help you to find the desired color codes. Add a new CSS rule in the *Web Inspector* and change the font color of the headline!

HTML, CSS & JS Frameworks

Rather than coding from scratch, frameworks enable you to utilize ready made blocks of code to help you get started. They give you a solid foundation for what a typical web project requires and usually they are also flexible enough for customization.

We have chosen ***Bootstrap*** as an example open source HTML, JS and CSS framework. It is one of the most widely used frameworks, it is easy to understand and it provides a great documentation with many examples.

The question whether a framework can be useful depends on the individual project and on the developer. Therefore, it is up to you to decide if you want to use it in your homeworks or projects.

Here is a summary of the main aspects of *Bootstrap*:

- **Open source** HTML, CSS, and JS framework
- Provides a **base styling** for common used HTML elements
- The **grid system** helps you to create multi-column and nested layouts, especially if your website should work on different devices
- Extensive list of **pre-styled components** (navigation, dropdown-menu, forms, tables, icons ...)
- **Customizable**: All CSS rules can be overridden by your own rules
- **Compatible** with the latest versions of all major browsers

---

Activity 4

In the last activity you will download and include Bootstrap in your project. You can try different Bootstrap components and modify styles.

1. **Download Bootstrap**

   http://getbootstrap.com/

   (Compiled and minified CSS, JavaScript, and fonts.

2. **Extract it and copy the sub-folders into your current project directory**

3. **Include the Bootstrap files in your `basics.html`**

   In your header link to the bootstrap minified CSS: `bootstrap.min.css`

Every time you work with CSS libraries/frameworks you should insert them right before your own stylesheets. Thereby your rules are prioritized and you can override the default properties of the libraries.

> *The purpose of minification is to increase the speed of websites, by removing spacing, indentation, newlines and comments. These elements are not required to successfully run CSS in a browser. The best practice of many developers is to maintain a regular version of the CSS file and when rolling out the project, the stylesheet is getting transformed to the optimized version.*

Bootstrap provides some JavaScript components too. In this lab our focus is on CSS but you should include the *Bootstrap JavaScript* and the required *jQuery JavaScript library* too, otherwise there are maybe problems with some components. Just insert these lines at the bottom of your `body` -block:

```
<script src="http://code.jquery.com/jquery-latest.min.js"></script>
<script src="js/bootstrap.min.js"></script>
```

As with CSS files, every time you work with JavaScript libraries/frameworks you should insert them right before your own JavaScript files.

4. **Reload `basics.html` in your browser**

   - Can you notice any changes?
   - Open the *Web Inspector* and check the different styles

5. **Insert a *Dropdown-Button* in your `basics.html`**

   A single button which triggers a small dropdown menu with multiple options.

   It is not a very complex problem and you could also solve it with a few lines of HTML and CSS, but today we are using Bootstrap for it. Sometimes it can be very useful to start with existing components, especially when building prototypes.

   Go to the official Bootstrap website and skim over the different styles and pre-configured components:

   http://getbootstrap.com/components/

   Search for the *Dropdown-Button* and copy the HTML code in your `basics.html` . Try the dropdown-menu in your browser afterwards.

6. **Override Bootstrap styles**

   - Add custom rules to change the *Background-Color* and the Hover-State of the previously created button in your `style.css` .

   This allows us to to make full use of Bootstrap's potential. We can use boilerplates for different components, add custom content and override some styles subsequently.

Keep in mind: If you have to override too many styles, it can be easier to work without a framework.

Bootstrap is a very popular front-end framework for web projects but there are also alternatives like for example:

- uikit
- Foundation
- PureCSS

Bonus Activities

You can choose either of the below bonus activities:

1. **Beautify basics.html**

   Play around with different CSS styles. Think of the design of webpages you like and try to recreate that look.

2. **Try other Bootstrap components**

   Go back to the Bootstrap website and look at their examples. Include some other Bootstrap components in your `basics.html` file.

   Just copy and paste the respective boilerplate codes and play around with the styles.

3. **Add a grid layout to your webpage**

   Take a look at some [grid examples])(http://getbootstrap.com/css/#grid) of Bootstrap. Use the first screenshot in this document as guideline to create a grid with the textual facts on the left and images on the right.

Submission of 1-minute paper

You have now completed the activities of Lab 1 and should have a basic understanding of html, css, and how you can style your webpage based on css selectors and libraries like Bootstrap. This knowledge will enable you to complete homework 1!

At the end of each lab we will ask you to submit a short 1-minute paper to help us and yourself to track your progress throughout the course. Please fill out the paper now!

See you next week!

Resources

- Chapters 1-3 (p. 34) in *D3 - Interactive Data Visualization for the Web* by Scott Murray
- https://developer.mozilla.org/en-US/docs/Web
- http://dataviscourse.net/2015/lectures/lecture-html/