CS 171

Lab 1

Welcome to the first lab of CS171!

Our labs are designed as work-books in the style of a self-guided tutorial. We ask you to read and work through the given example problems, and to hand in the code of your completed lab at the end of each week, together with your homework.

We embrace the concept of learning by doing. To truly master new programming and development skills, you have to spend the time to figure things out and to try different approaches and examples. However, you are not alone in this! CS171 staff is available for any questions that pop up along the way. We encourage you to pester them with questions, but at the same time, make sure that you come to the lab prepared and ready to code!

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 educational license can be obtained here. (You are free to use your own IDE, but we will only officially support Webstorm.)
- You have read Chapter 3 (up to page 36) in *D3 - Interactive Data Visualization for the Web* (Second Edition) by Scott Murray.
- We encourage you to use Google Chrome or Mozilla Firefox as your primary web browser during all labs and homeworks. Those are the browsers we will use for grading.

- We also encourage you to take a look at Alex Lex's interactive lecture on html at the University of Utah.

Setup

During the next weeks, you will be working through the book *D3 - Interactive Data Visualization for the Web* (Second Edition) by Scott Murray. The book provides a lot of sample code (see page 5 of the book, *Using Sample Code*).

- Download and extract the sample code for the book now. It can be found here.
- Set up a directory on your computer for the sample code and remember its location.
- Starting next week, while working through the book, you should look at and run the sample code. It will help you prepare for labs and homeworks!

For todays lab, you should prepare the following:

- Create a directory for your HTML project
- You can already create subdirectories "css" and "js".
- You will add all remaining files while working through the activities of this lab.

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.

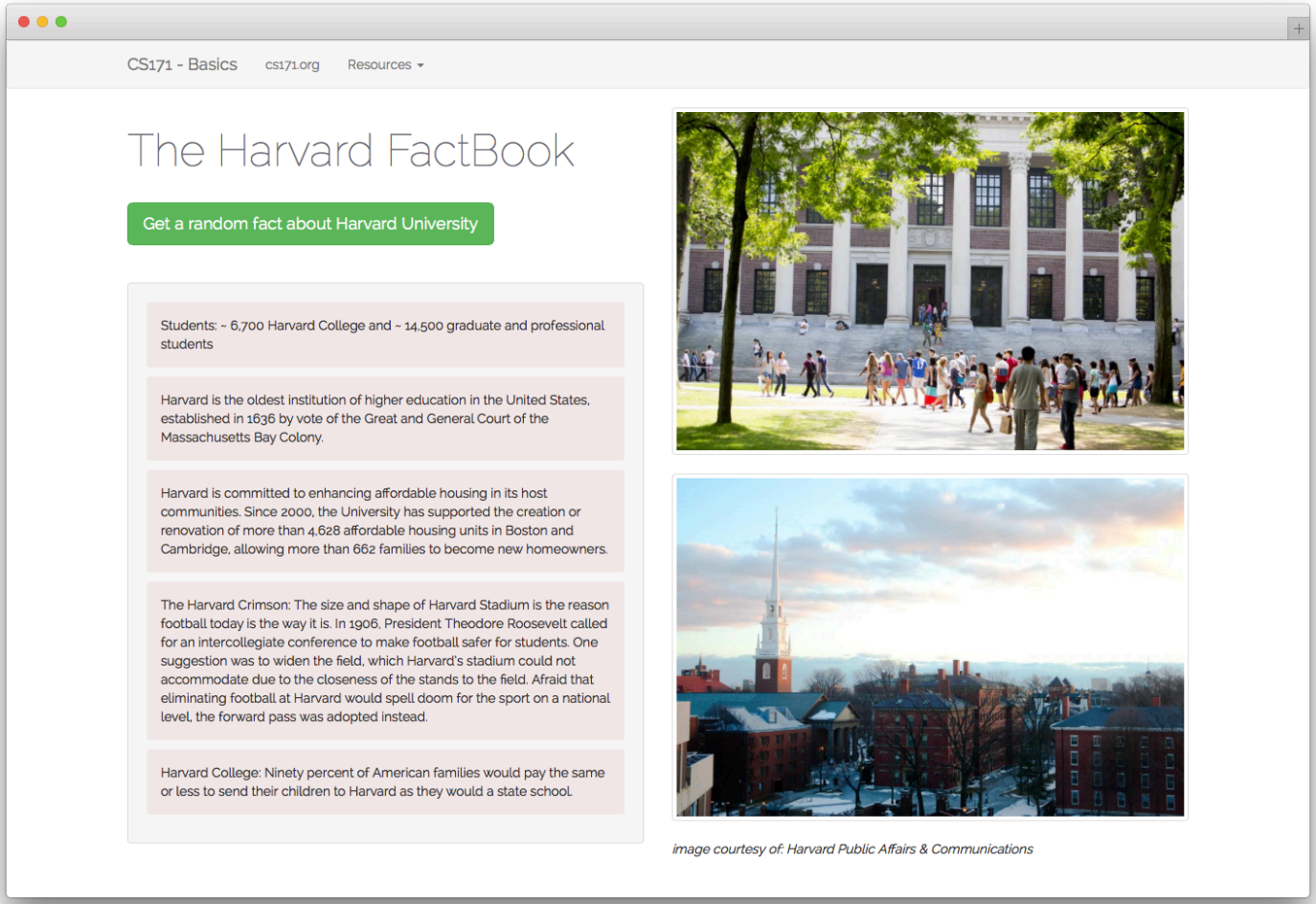


image courtesy of: Harvard Public Affairs & Communications

HTML (Hypertext Markup Language)

As you have read, HTML is used to structure content for web browsers. It enables us to differentiate between content and structure of a document.

A brief HTML 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 as shown in a web browser:

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 get familiar with this structure:

```
<!DOCTYPE HTML>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
This is where the main content of the page goes.
</body>
</html>
```

CSS Selectors: Classes and IDs

Classes and IDs are extremely helpful for selecting specific HTML elements. Your CSS and JavaScript code will rely heavily on classes and IDs to identify elements.

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

In the above HTML snippet, the div-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, each element can only have a single ID, 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 (empty 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**

   - 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

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. **Create a new folder `js` in your project, download the file `dom-example.js` and save it in your newly created folder**

   http://www.cs171.org/2018/assets/scripts/lab1/dom-example.js

2. **Include the external JavaScript file that you just downloaded in your HTML document. Add the following line at the bottom of your previously created `basics.html` (inside the `<body></body>`)**

   ```
   <script src="js/dom-example.js"></script>
   ```

This script will listen to your button (*ID: cs171-basics*). It will automatically deliver random facts if you click on the button.

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

   We strongly encourage you to use Google Chrome or Mozilla Firefox, these are the browsers we will use for grading. On Mac OS, you can navigate to the Developer Tools using:

   - Chrome: View → Developer → Developer Tools
   - Firefox: Tools → Web Developer → Inspector

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

4. **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 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 this will look like your raw HTML, but if any JavaScript code has been executed, the current DOM may be different, as JavaScript commands can add, remove, and adjust the DOM dynamically.

5. **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.

   If you have trouble getting this step to work you should double check that you have added the appropriate IDs to your HTML tags!

6. **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 and delete some paragraphs!

> 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.
>
> **From now on, whenever you are programming HTML, CSS, JavaScript, or D3, you should always have the developer console open. This helps you in debugging and figuring out what is going on in the code!**

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 either 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 always in the HTML header (inside the `<head></head>` elements of your HTML file).

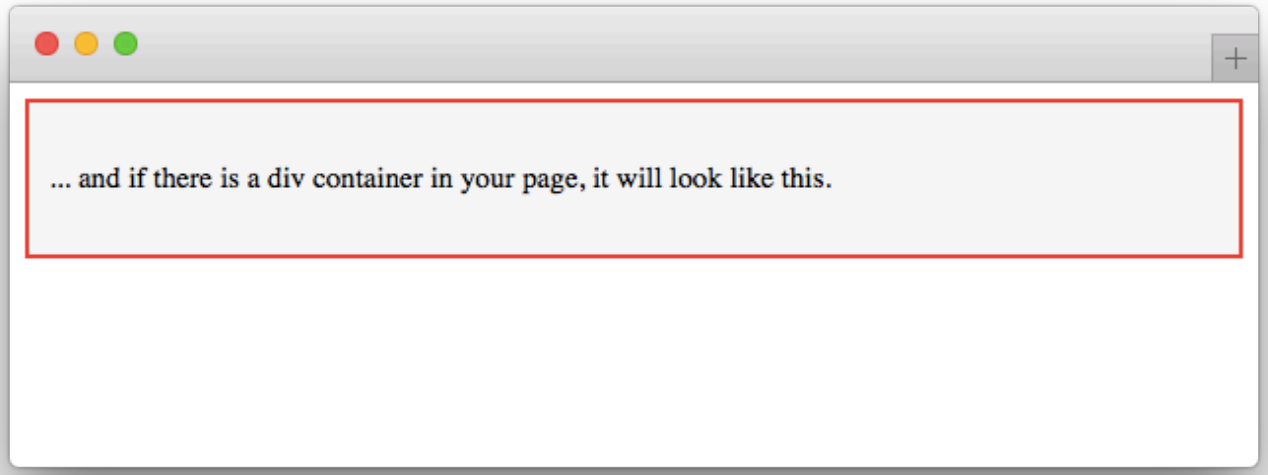
```
<link rel="stylesheet" href="css/style.css">
```

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

CSS styles consist of selectors and properties, grouped in curly brackets. A 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;
}
```



If you are searching for an exhaustive list of style properties we recommend Mozilla's Developer Platform or w3schools.com. Some CSS properties are needed quite often, so you should try to memorize them.

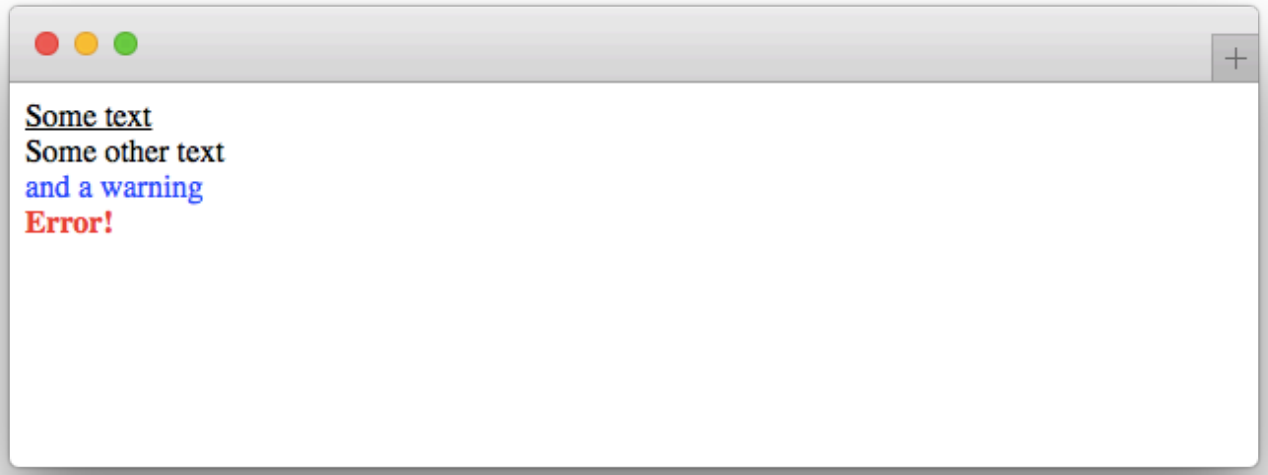
In the above example we have assigned our CSS rule to all div containers but if we want to style specific elements we can use the selectors *id* and *class*.

As you can see in the example below, IDs are preceded with a hash mark (*#article-1*) and class names are preceded with a period (*.error*). You can also use descendant selectors to address nested tags (*.article .warning*).

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:



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 (e.g., decisions about fonts, font size, colors or scales) 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.

- Be mindful that 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 in the web inspector. 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!

- 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.

In CS171 we use ***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 the Bootstrap JavaScript library in your project.

1. **Download Bootstrap**

   http://getbootstrap.com/

   Choose the first option (compiled CSS and JS), "Download Bootstrap". This will download a zip file.

2. **Extract the zip file 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 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 may be problems with some components.

   Insert these lines at the bottom of your `body` -block:

   ```
   <script src="https://code.jquery.com/jquery-latest.min.js"></script>
   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper.m
   in.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.

Note that we are linking to online versions of the jquery and popper Javascript libraries, alternatively, we could download local copies into the project directory, and link to them.

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

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

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

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

    Adding a dropdown menu can be done 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 example code in your `basics.html`. Try the dropdown-menu in your browser afterwards. Adding new Bootstrap elements to your HTML page is often very quick and easy, by using the example code on the bootstrap webpage.

6. **Override Bootstrap styles**

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

    - We assume that you do not know the CSS selectors or properties for changing the background color and the hover-state yet. Search for it online, either in google or at W3 school. Example search term: CSS background color.

    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:

- uikit
- Foundation

- PureCSS

Bonus Activities (optional!)

These bonus activities are not required, but we recommend that you try them!

1. **Beautify basics.html**

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

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 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 lab

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!

If you are done early, please help others around your table. If your entire table is done, feel free to get started on the homework.

Please submit your completed lab 1 together with your homework 1 (in Canvas)! More detailed submission instructions will be given in the homework.

See you next week!

Resources

- Chapter 3 (up to page 36) in *D3 - Interactive Data Visualization for the Web* (Second Edition) by Scott Murray
- https://developer.mozilla.org/en-US/docs/Web
- University of Utah's visualization course. Interactive introduction to html: http://dataviscourse.net/2015/lectures/lecture-html/