Data Visualization

INF552 - 2023 - Session 01 - exercices
Crash course HTML/CSS/SVG/DOM/JS

ÉCOLE POLYTECHNIQUE

Inria

emmanuel.pietriga@inria.fr

olivier.gladin@inria.fr

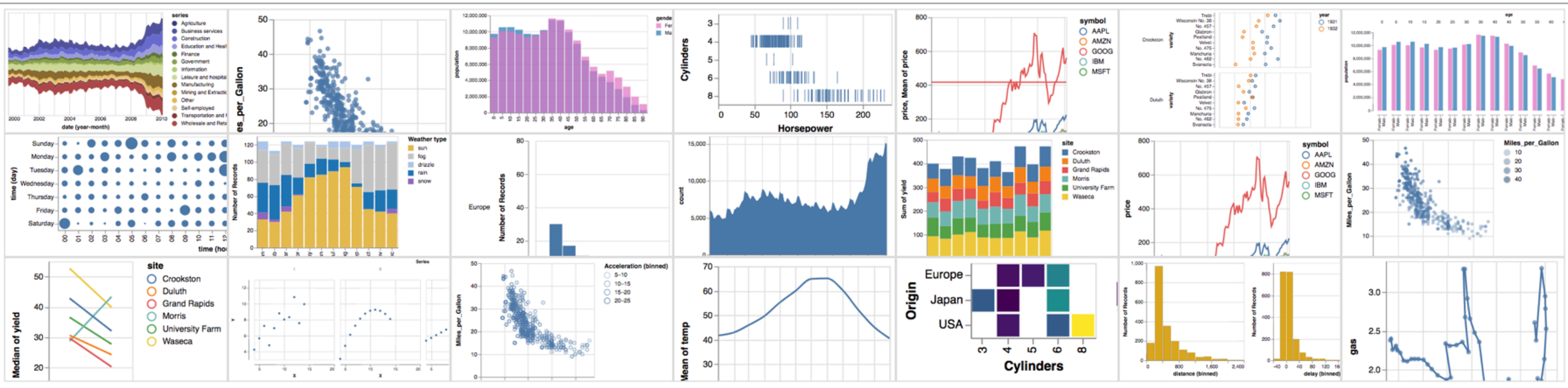
Interactive Data Visualization on the Web

- Why interactive?
  - Shneiderman's Visual Information Seeking Mantra
  - Limited number of visual variables for mapping data attributes
  - Explore alternative representations
  - Coordinated multiple views (*e.g.*, Brushing & Linking)
  - Encourages engagement

Interactive Data Visualization on the Web

- Why on the Web?
  - Reach a global audience
  - Using web-standard technologies means that the visualization can be seen by anyone regardless of the OS, browser, device (workstation, laptop, tablet, smartphone, etc.)
  - Can be done with freely-accessible tools

https://d3js.org



https://vega.github.io/vega-lite/

- Underlying technologies
  - HTML+CSS
  - JavaScript
  - DOM
  - SVG
  - CSV, JSON, …
- Goal: have basic knowledge about the languages and paradigms that will be used throughout the following exercise sessions: HTML/SVG DOM manipulation in Javascript, basic CSS rules, event-based programming with JavaScript

HTML

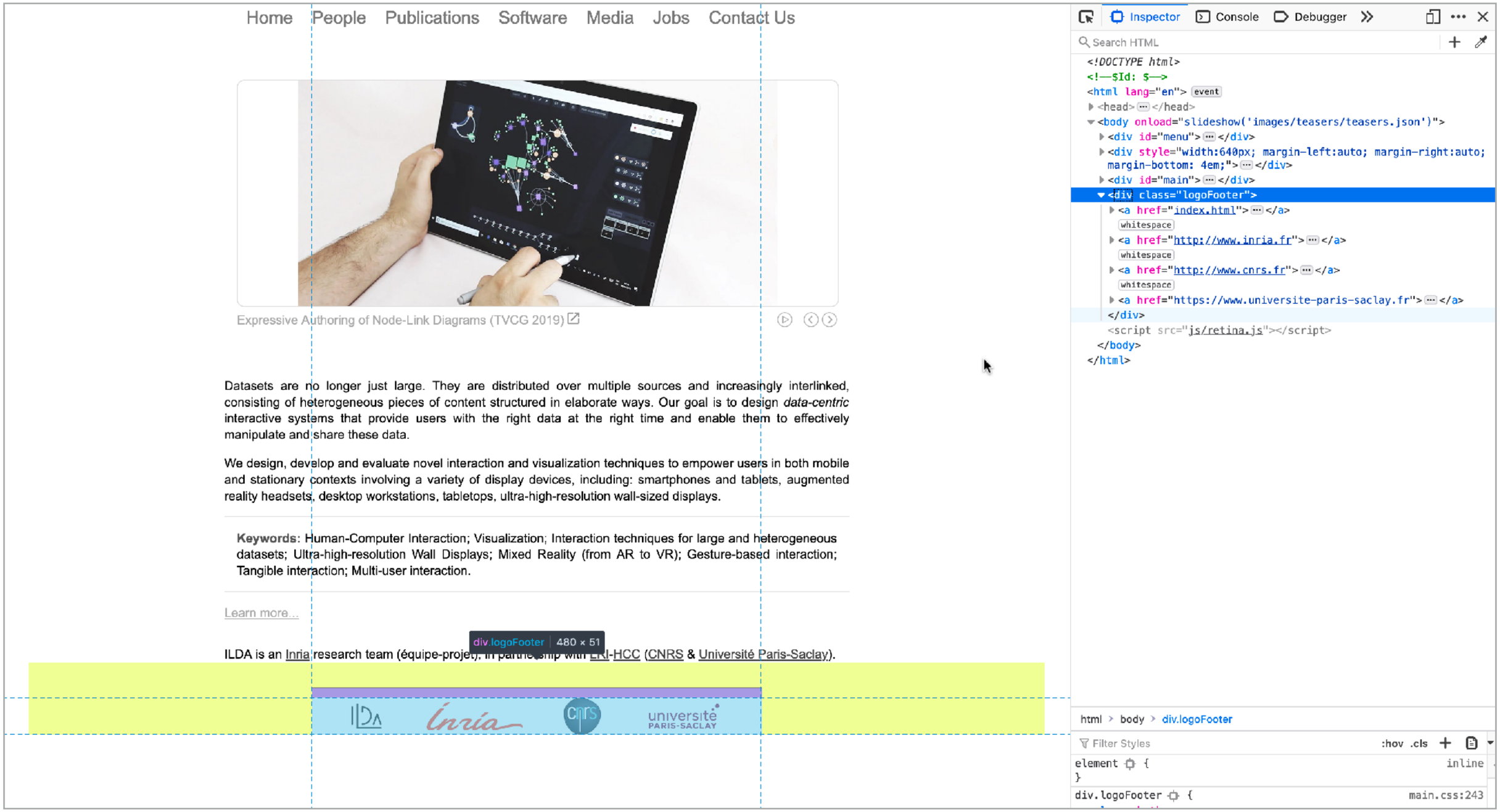
- `html` - Surrounds all HTML content in a document.
- `head` - The document head contains all metadata about the document, such as its title and any references to external stylesheets and scripts.
- `title` - The title of the document. Browsers typically display this at the top of the browser window and use this title when bookmarking a page.
- `body` - Everything not in the head should go in the body. This is the primary visible content of the page.
- `h1, h2, h3, h4` - These let you specify headings of different levels. h1 is a top-level heading, h2 is below that, and so on.
- `p` - A paragraph
- `ul, ol, li` - Unordered lists are specified with ul, most often used for bulleted lists. Ordered lists (ol) are often numbered. Both ul and ol should include li elements to specify list items.
- `em` - Indicates emphasis. Typically rendered in italics.
- `strong` - Indicates additional emphasis. Typically rendered in boldface.
- `a` - A link. Typically rendered as underlined, blue text, unless otherwise specified.
- `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.

HTML - id & class

```
<!DOCTYPE html>

<!-- $Id: index.html 5740 2017-08-10 09:13:04Z epietrig $ -->

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>ILDA - Homepage</title>
   <link rel="stylesheet" href="main.css" type="text/css" media="all" />
    <link rel="icon" type="image/png" href="favicon.ico" />
  </head>

  <body>

  <div id="menu">
    <ul>
      <li><a href="index.html" class="mainMenuItem">Home</a></li>
      <li><a href="people.html" class="mainMenuItem">People</a></li>
      <li><a href="publications.html" class="mainMenuItem">Publications</a></li>
      <li><a href="software.html" class="mainMenuItem">Software</a></li>
      <li><a href="jobs.html" class="mainMenuItem">Jobs</a></li>
      <li><a href="contact.html" class="mainMenuItem">Contact Us</a></li>
    </ul>
  </div>

  <div style="width:640px; margin-left: auto; margin-right: auto; margin-bottom:-64px; margin-top:-1.0em">
    <div id="teaser" style="margin-bottom: 0em; margin-top:0em">
      <img id="teaserImg" class="ctr" src="images/logo_ilda.svg" height="160" alt="ILDA logo" style="padding-top:40px;padding-bottom:40px"/>
    </div>
 </div>

    <div id="main">

      <p><a href="http://www.inria.fr">Inria</a> research team (équipe-projet), in partnership with <a href="http://www.lri.fr">LRI</a>-<a href="http://
www.lri.fr/equipe_en.php?eq=19&amp;lang=EN">HCC</a> (<a href="http://www.cnrs.fr">CNRS</a> &amp;amp; <a href="http://www.u-psud.fr/">Université Paris-Sud</
a>).</p>

      <p class="boxed"><strong>Keywords:</strong> Human-Computer Interaction; Interaction techniques for large datasets; Web of Data; Wall displays;
Gesture-based interaction; Multi-user interaction.</p>

      <p>In an increasing number of domains, computer users are faced with large datasets, that are often interlinked and organized according to elaborate
structures thanks to new data models such as those that are arising with the development of, e.g., the <a href="http://www.w3.org/2013/data/">Web of Data</
a>. Rather than seeing the inherent complexity of those data models as a hindrance, we aim at leveraging it to design new interactive systems that can
better assist users in their data understanding and processing tasks.</p>

      <p>These <em>Data-centric Interactive Systems</em> aim at providing users with the right information at the right time, presenting it in the most
meaningful manner, and letting users efficiently manipulate, edit and share these data with others. This entails minimizing the effort required to retrieve
and relate data from relevant sources; displaying data using presentation techniques that match the data's characteristics and the users' tasks; and
providing users with means of interacting with the data that effectively support their train of thought.</p>

    </div>

    <div class="logoFooter">
      <a href="http://www.inria.fr"><img src="images/logo_inria.png" width="111" height="40" alt="INRIA logo" /></a>
      <a href="http://www.cnrs.fr"><img src="images/logo_cnrs.png" width="40" height="40" alt="CNRS logo" /></a>
      <a href="http://www.u-psud.fr"><img src="images/logo_psud.png" width="73" height="40" alt="Univ. Paris-Sud logo" /></a>
    </div>

  </body>

</html>
```

Box Model

To a browser, everything is a box



CSS

```css
/* $Id: main.css 1495 2015-03-12 08:23:39Z chapuis $ */

body {
  font-family: Arial, Verdana, Lucida, Helvetica, sans-serif;
  font-size:10pt;
  margin:4px 28px 28px 28px;
  text-align: justify;
  background:white;
}

div#main {
  width: 99.99%;
  max-width:48em;
  margin-left: auto;
  margin-right: auto;
}

div#accessibility {
  display:none;
}

h1,h2,h3,h4 {
  background: transparent;
  color: #777;
  text-align: left;
}

div.logoFooter {
    clear:both;
    width:360px;
    margin-left: auto;
    margin-right: auto;
    border-top:1px solid black;
    margin-top:4em;
    padding-top:10px;
    text-align:center;
}
```

- Home
- People
- Publications
- Software
- Media
- Jobs
- Contact Us

Expressive Authoring of Node-Link Diagrams (TVCG 2019)

Datasets are no longer just large. They are distributed over multiple sources and increasingly interlinked, consisting of heterogeneous pieces of content structured in elaborate ways. Our goal is to design *data-centric* interactive systems that provide users with the right data at the right time and enable them to effectively manipulate and share these data.

We design, develop and evaluate novel interaction and visualization techniques to empower users in both mobile and stationary contexts involving a variety of display devices, including: smartphones and tablets, augmented reality headsets, desktop workstations, tabletops, ultra-high-resolution wall-sized displays.

Keywords: Human-Computer Interaction; Visualization; Interaction techniques for large and heterogeneous datasets; Ultra-high-resolution Wall Displays; Mixed Reality (from AR to VR); Gesture-based interaction; Tangible interaction; Multi-user interaction.

Learn more...

ILDA is an Inria research team (équipe-projet), in partnership with LRI-HCC (CNRS & Université Paris-Saclay).

Without any CSS stylesheet

Home People Publications Software Media Jobs Contact Us

Expressive Authoring of Node-Link Diagrams (TVCG 2019)

Datasets are no longer just large. They are distributed over multiple sources and increasingly interlinked, consisting of heterogeneous pieces of content structured in elaborate ways. Our goal is to design *data-centric* interactive systems that provide users with the right data at the right time and enable them to effectively manipulate and share these data.

We design, develop and evaluate novel interaction and visualization techniques to empower users in both mobile and stationary contexts involving a variety of display devices, including: smartphones and tablets, augmented reality headsets, desktop workstations, tabletops, ultra-high-resolution wall-sized displays.

Keywords: Human-Computer Interaction; Visualization; Interaction techniques for large and heterogeneous datasets; Ultra-high-resolution Wall Displays; Mixed Reality (from AR to VR); Gesture-based interaction; Tangible interaction; Multi-user interaction.

Learn more...

ILDA is an Inria research team (équipe-projet), in partnership with LRI-HCC (CNRS & Université Paris-Saclay).

With a CSS stylesheet

CSS selectors

Type selectors

```
h1          /* Selects all level 1 headings <h1>                    */
p           /* Selects all <p> elements                             */
strong      /* Selects all <strong> elements                        */
div         /* Selects all <div> elements                           */
```

Descendant selectors

```
h1 em       /* Selects <em> elements contained in an <h1>           */
div p       /* Selects <p> elements contained in a <div>            */
```

ID selectors

```
#header     /* Selects unique element whose ID is 'header'          */
#nav        /* Selects unique element whose ID is 'nav'             */
```

CSS selectors

Class selectors

```
.caption   /* Selects all elements with class 'caption'              */
.label     /* Selects all elements with class 'label'                */

.axis.x    /* Selects all elements with both classes 'axis' and 'x' */
.axis.y    /* Selects all elements with both classes 'axis' and 'y' */
```

More advanced examples

```
div.foo    /* Selects all <div> elements with class 'foo'            */
#button.on /* Selects unique element whose ID is 'button'
              but only if it has class 'on'                          */
```

CSS property/value pairs

```
div {
    margin: 10px;
    padding: 25px;
    background-color: #EEE;
    color: #444;
    font-family: Helvetica, Arial, sans-serif;
}
```

https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

JavaScript

Constants and Variables

```
const aNumber = 5;

const aString = "foo";

const aBoolean = true;
```

```
let aNumber = 5;

let aString = "foo";

let aBoolean = true;

aNumber = 10;
```

The older `var` keyword also works (but has a somewhat different scope).

JavaScript

Data structures

```
let anArray = [1,2,4,8,16,32,64];
    // anArray[3] returns 8

let anotherArray = ["foo", "bar", "x", "y"];
    // anotherArray[3] returns “y”

let arrayMixingTypes = [1, true, "foo"];
    // not recommended
```

JavaScript

Data structures

```
let anObject = {attr1: true, attr2: 12, attr3: "foo"};

anObject.attr1 // returns true
anObject.attr2 // returns 12

anObject.attr || anObject[attr]  // (use 2nd option if attr is num, like anObject[1976])
```

- You can create arrays of objects
- You can assign arrays as object attribute values

JavaScript

Data structures

- JSON looks the same, with property names double quoted:

```
let anObject = {"attr1": true, "attr2": 12, "attr3": "foo"};
```

JavaScript

Opeators and Control Structures

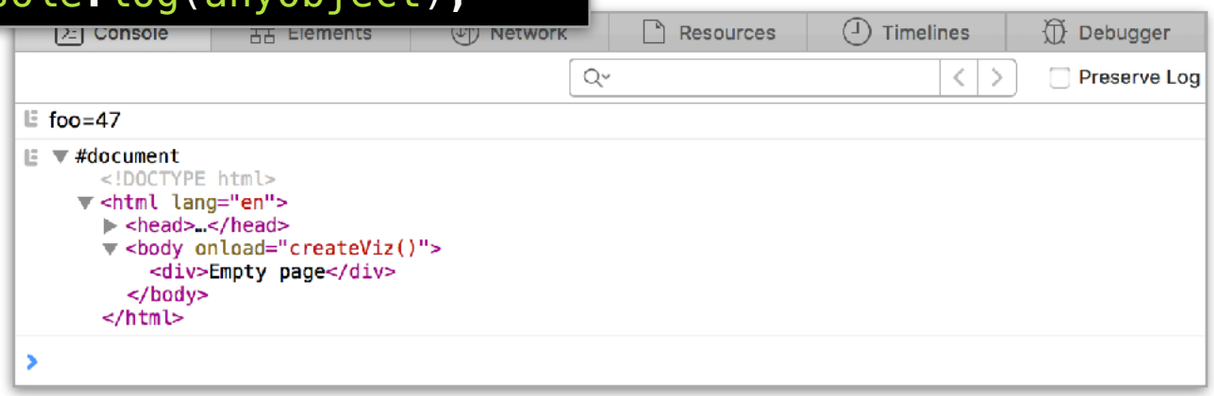
div mod

- Mathematical operators: `+ - * / %`
- Comparison operators: `== != < > <= >=`
- Control structures:

```
if (test){}
else {}

switch(foo){
    case val1:{break;}
    case val2:{break;}
    default:{}
}

for (init;test;update){}

while(test){}
```

Debug/text output:

```
console.log("foo="+foo);
console.log(`foo=${foo}`);
console.log(anyObject);
```



JavaScript

Functions

```
let oneThird = function(num) {
    return num / 3;
};

let res = oneThird(10);

/* comments */
// comments
```

JavaScript - Referencing Scripts

```
<head>
    <title>Page Title</title>
    <script type="text/javascript" src="myscript.js"></script>
</head>
```

JavaScript

- Loosely typed language (use `typeof` to get the type)
- Avoid polluting the global namespace (`window`)
  - Create an empty global object and add attributes to it

`https://developer.mozilla.org/en-US/docs/Web/JavaScript`

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>INF552 - Exercice 1</title>
        <!-- https://d3js.org/ -->
        <script type="text/javascript" src="js/d3.v7.min.js"></script>
   </head>

    <body>
        <script type="text/javascript">
            let foo = 47;
            console.log(foo);
        </script>
    </body>

</html>
```

JS code inside **<script>** element directly in the HTML body

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>INF552 - Exercice 1</title>
        <!-- https://d3js.org/ -->
        <script type="text/javascript" src="js/d3.v7.min.js"></script>
        <script type="text/javascript" src="js/ex01.js"></script>
    </head>

    <body onload="createViz()">
    </body>

</html>
```

Cleaner: JS code in a separate file referenced from a **`<script>`** element in the HTML header.

DOM (Document Object Model)

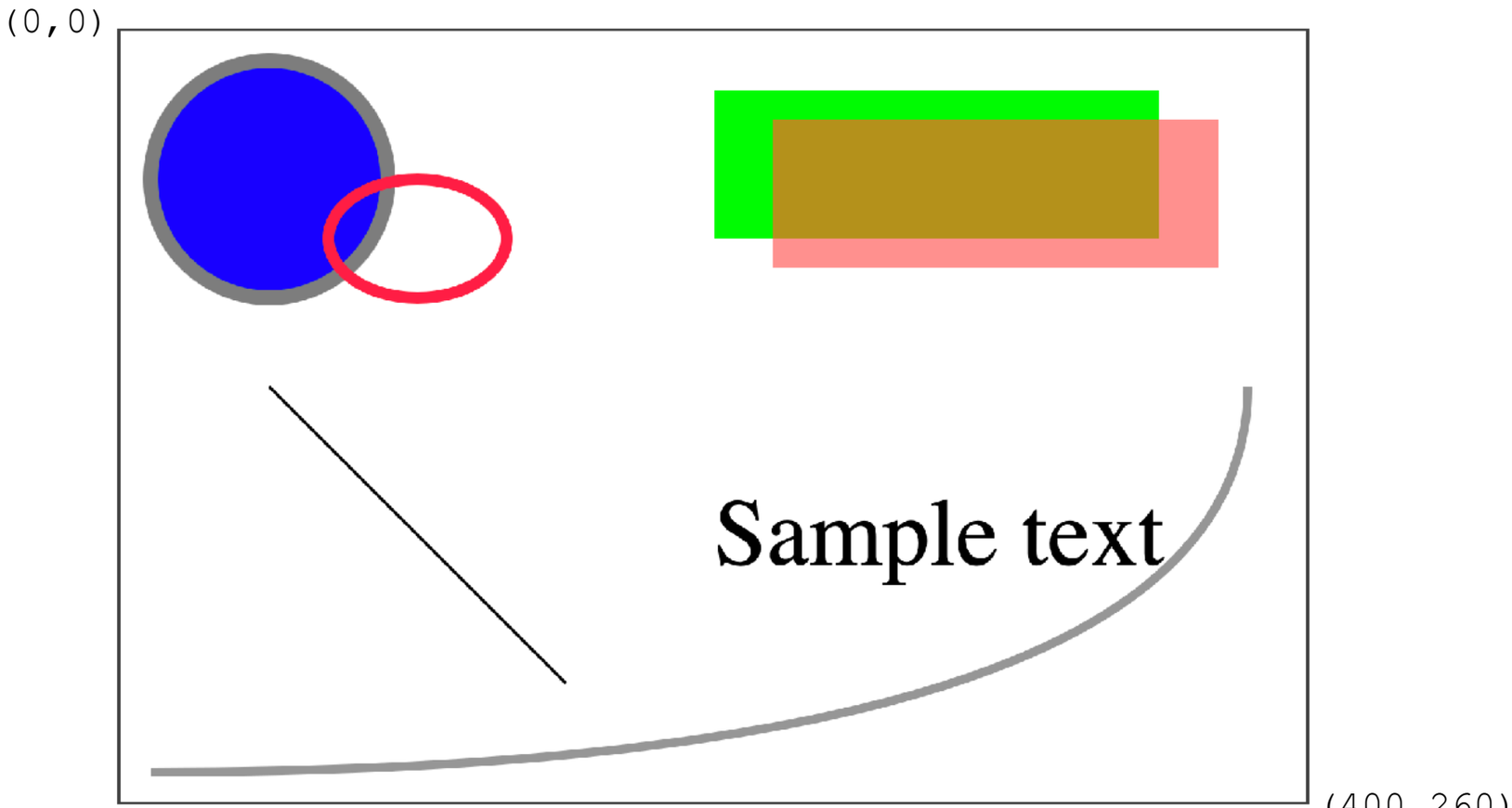
- Add and remove elements from the HTML/SVG tree structure
- Set attributes on nodes
- Useful methods: `document.createElement, document.createElementNS, document.createTextNode, Element.appendChild, Element.setAttribute, document.getElementsByTagName, document.getElementById, document.querySelector…`

`https://developer.mozilla.org/en-US/docs/Web/API/Document`

`https://developer.mozilla.org/en-US/docs/Web/API/Element`

`https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector`

SVG - Scalable Vector Graphics



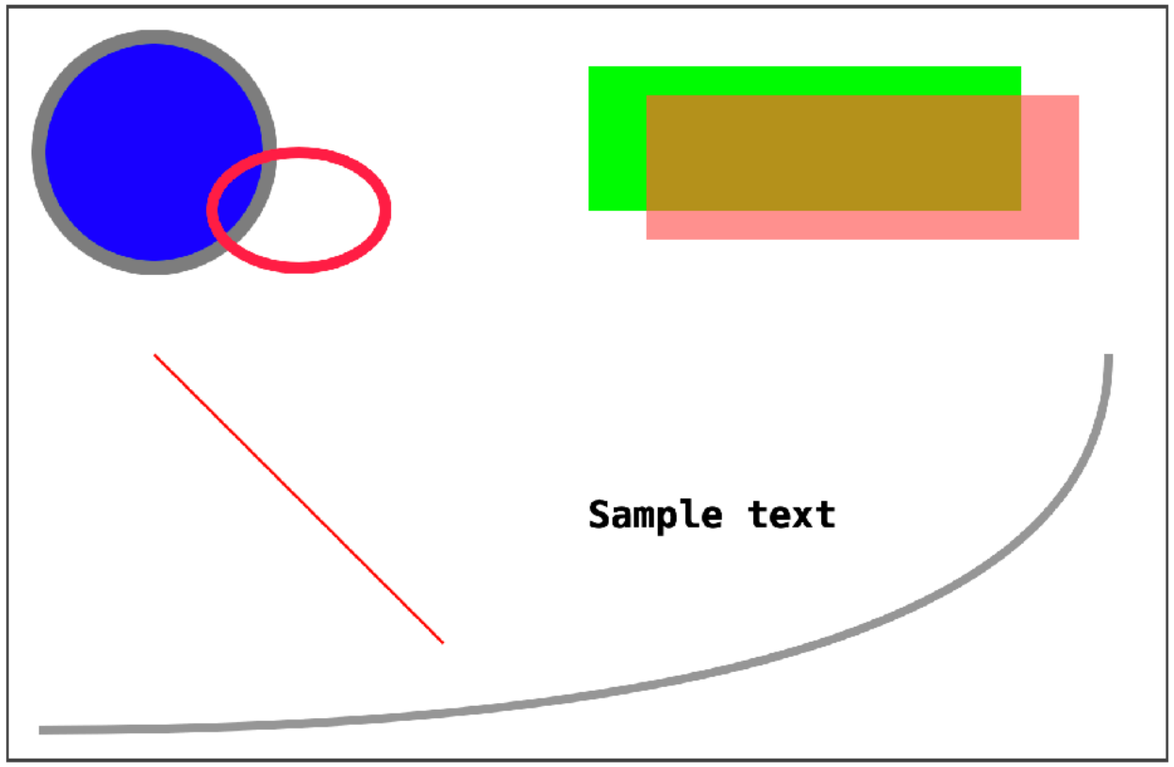

```
<svg width="400" height="260">
    <!-- blue circle with a 5px-gray border-->
    <circle cx="50" cy="50" r="40" fill="blue" stroke="gray" stroke-width="5"/>
    <!-- ellipse with a 4px redish border and no fill color-->
    <ellipse cx="100" cy="70" rx="30" ry="20" fill="none" stroke="#FF2244" stroke-width="4"/>
    <!-- two rectangles partially overlapping, the one above (which us red) is semi-transparent-->
    <rect x="200" y="20" width="150" height="50" fill="#0F0"/>
    <rect x="220" y="30" width="150" height="50" fill="#F00" opacity=".5"/>
    <!-- simple black line -->
    <line x1="50" y1="120" x2="150" y2="220" stroke="black"/>
    <!-- simple text element -->
    <text x="200" y="180">Sample text</text>
    <!-- a quadratic bézier curve -->
    <path fill="none" stroke="#999" stroke-width="3" d="M10,250 Q380,250 380,120" />
</svg>
```

SVG - Scalable Vector Graphics

We can also style elements using CSS rules, which apply to all elements that match the selector:

```
circle {
    fill:blue;
    stroke:gray;
    stroke-width:5;
}

ellipse {
    fill:none;
    stroke:#FF2244;
    stroke-width:4;
}
```

```
line {
    stroke:black;
}

text {
    font-family:monospace;
    font-weight:bold;
}

path {
    fill:none;
    stroke:#999;
    stroke-width:3;
}
```

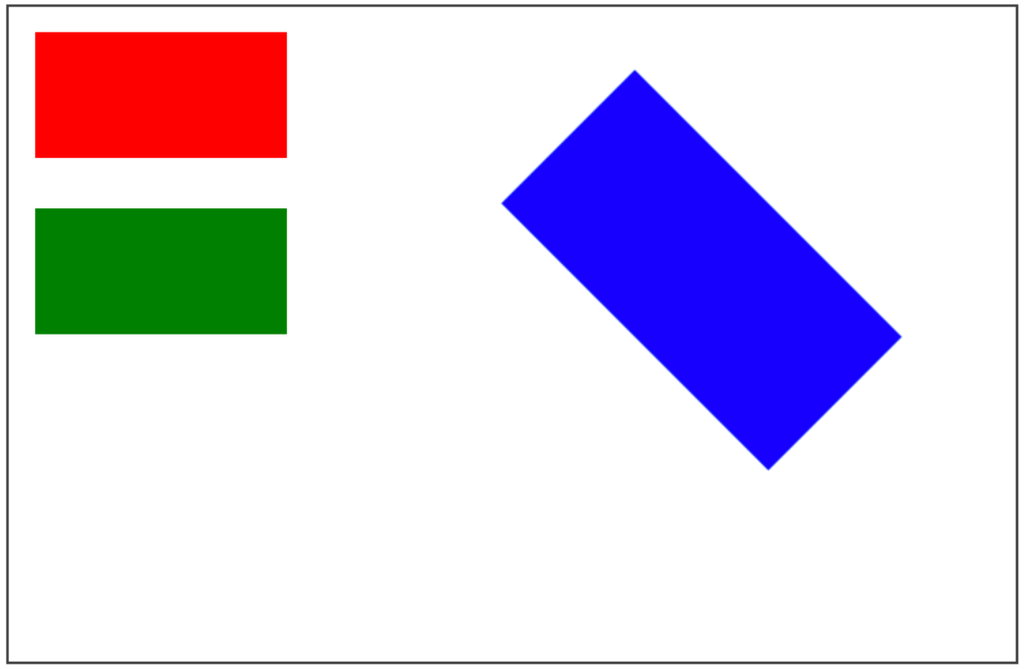



```
<svg width="400" height="260">
   <circle cx="50" cy="50" r="40" />
   <ellipse cx="100" cy="70" rx="30" ry="20"/>
   <rect x="200" y="20" width="150" height="50" style="fill:#0F0"/>
   <rect x="220" y="30" width="150" height="50" style="fill:#F00;opacity:.5"/>
   <line x1="50" y1="120" x2="150" y2="220" style="stroke:red"/>
   <text x="200" y="180">Sample text</text>
   <path d="M10,250 Q380,250 380,120" />
</svg>
```

`style` attributes declared on individual elements override CSS rules when in conflict, as illustrated with the `<line>`, which is now red instead of black

SVG - Scalable Vector Graphics

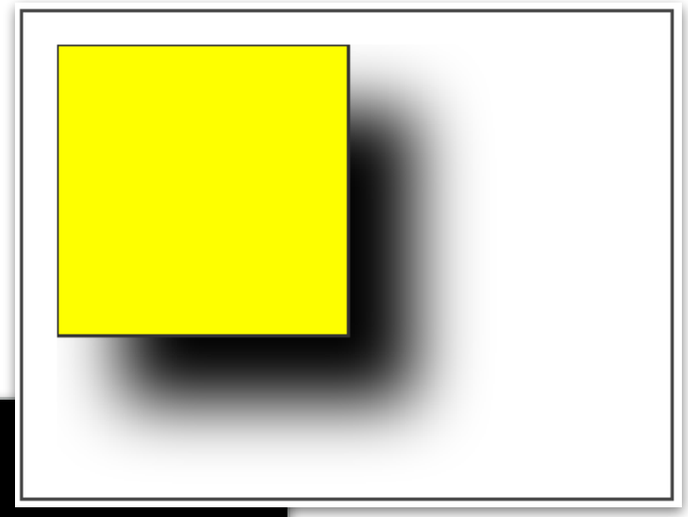
Affine Transforms



```
<rect x="10" y="10" width="100" height="50" style="fill:red"/>
<rect x="0" y="0" width="100" height="50" style="fill:green"
      transform="translate(10,80)" />
<rect x="0" y="0" width="100" height="50" style="fill:blue"
      transform="translate(10,150) scale(1.5) rotate(45 180 150)"/>
```

SVG - Scalable Vector Graphics

Many more possibilities, including, *e.g.*, filters:



```
<svg width="200" height="150">
    <defs>
      <filter id="ds" x="0" y="0" width="200%" height="200%">
        <feOffset result="offOut" in="SourceAlpha" dx="20" dy="20" />
        <feGaussianBlur result="blurOut" in="offOut" stdDeviation="10" />
        <feBlend in="SourceGraphic" in2="blurOut" mode="normal" />
      </filter>
    </defs>
    <rect x="10" y="10" width="90" height="90" fill="yellow" stroke="#333"
          filter="url(#ds)" />
</svg>
```

Detailed SVG documentation:

`https://developer.mozilla.org/en-US/docs/Web/SVG/Element`

XML Namespaces

- Elements from different XML vocabularies can be inserted in the same tree structure.

- In our case: an SVG canvas will be inserted in an HTML document.

- The SVG vocabulary resides in a different namespace. When creating SVG elements using the DOM API, explicitly set the namespace:

```
let mainDiv = document.getElementById("main");
let svgEl = document.createElementNS("http://www.w3.org/2000/svg", "svg");
svgEl.setAttribute("width", 480);
svgEl.setAttribute("height", 480);
mainDiv.appendChild(svgEl);
```

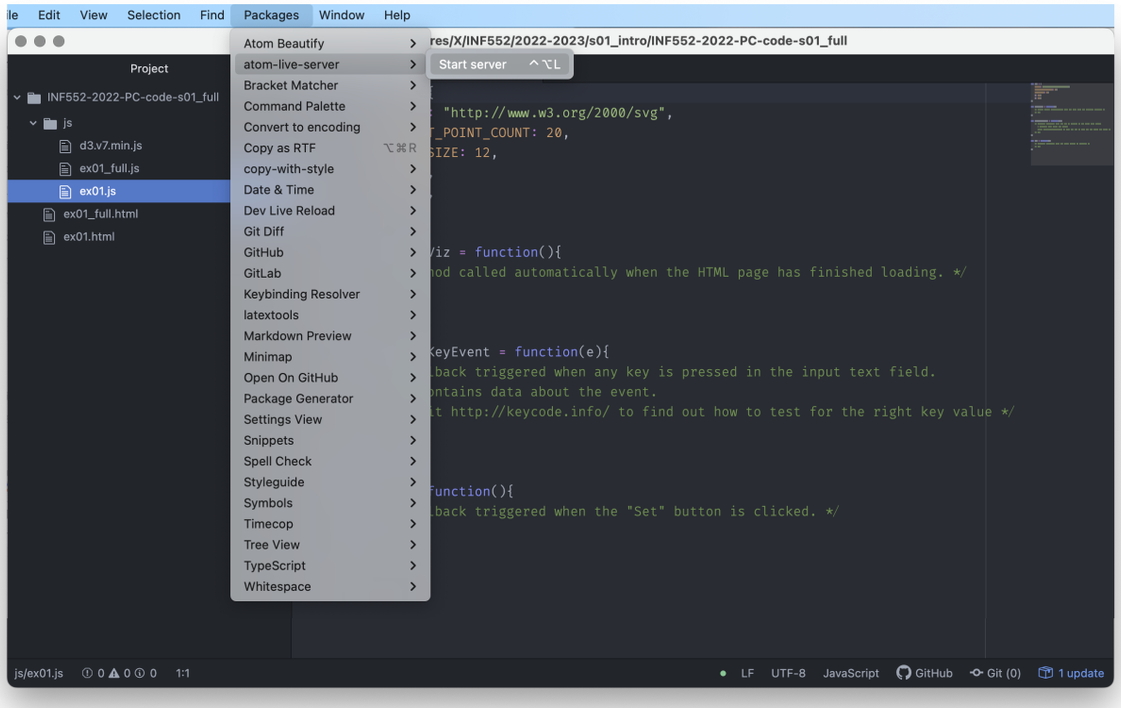
Use a (local) Web server

- Depending on your browser settings, you might not be able to load external resources (data files, etc.) if you use the **`file://`** protocol.

- Start a simple Web server in the relevant directory

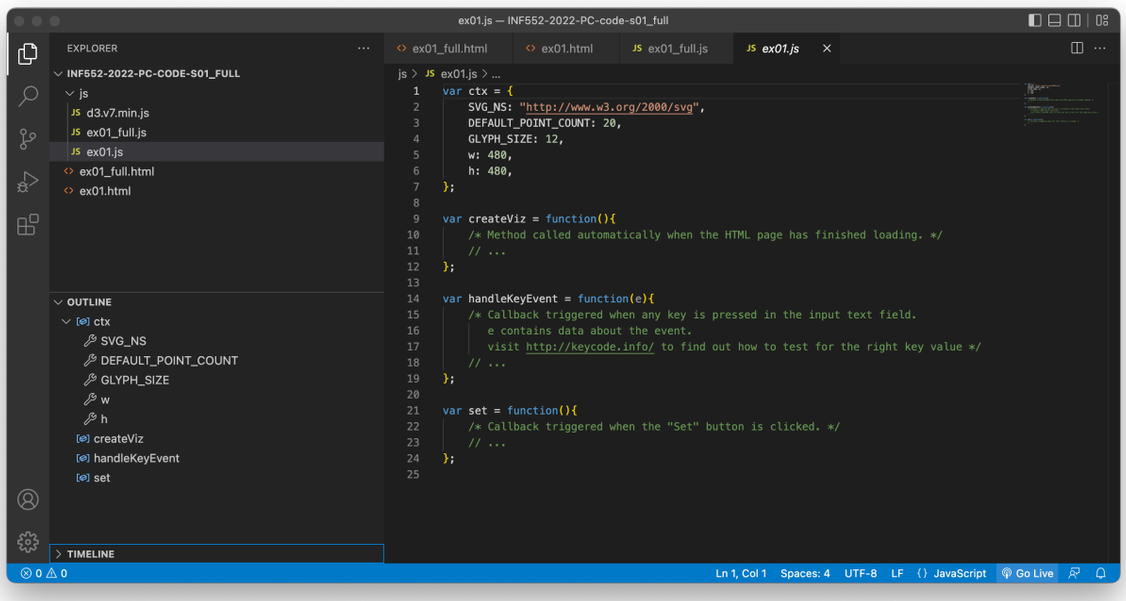
```
cd INF552-2023-PC-code-s01/
python3 -m http.server [portNumber]
```

Access from **`http://localhost:portNumber/`**



- or use an integrated solution such as atom-liver-server or VSCode Live Server

```
https://atom.io/packages/atom-live-server

https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer
```



- Edit **`.js`** and **`.html`** files in a text editor (such as, *e.g.*, Atom)

- Always use relative URLs