Week 02: React, Node, HTML

...semantics, semantics, semantics.

Agenda

- Prerequisites
- Creating a React project from a `vite` template
- React project structure and practices
- HTML5 revision & document structure
- Inline, Block elements
- Tables, Forms
- Meta tags
- Lighthouse, Inspect
- Documentation for WEB development
- Demo - dummy form for investing

Let's start!

Prerequisites

You should have `node` and `npm` commands available. **These commands are available on school Linux computers (nymfe, musa), so you can skip this step**.

When you work on your own computers:

- If you have not installed a Node.js manager yet, do so via nvm or fnm install pages. We advise you to install the manager of your choice under the Unix environment (Linux, MacOS, WSL). Windows has its own version of `nvm` - `nvm-windows` which is not maintained by the creators of `nvm` for Unix computers. We prefer you to work under WSL so both nvm/fnm will work fine there.
- If you have installed your Node.js manager (nvm/fnm) and the commands are not available, it might mean you have not installed Node yet, or your path has not been updated. Check the links above to see how to install the latest version depending on your Node.js manager. There are FAQ & troubleshooting pages for both.

Let's create a React app via the `vite` template

```
npm create vite@latest seminar-02 --template react-ts
```

or just:

```
npm create vite@latest
```

Where you can specify the underlying frontend technology & TypeScript support.

Inspect your Node project

- Check out `package.json` & included scripts
- Install dependencies included in your `package.json` via:

```
npm i
# or
npm install
```

- Check out `package-lock.json`
- Check out the project strucure:
    - `index.html` file
    - `src` & `public` folders
    - Check out where React is added injected onto the page

Run your React app

```
npm run dev
```

Possible discussion:

- How does it work? How do the files get in the browser?
- Now, try to change something on the page. What does HMR mean?
- Create a folder for pages/views and a folder for components? What's the difference between pages and components, when we represent them both as components?

JSX/TSX

JavaScript(/TypeScript) XML (formally JavaScript/TypeScript syntax eXtension) - allows writing logic & code along with the underlying markup

```
import { useState } from "react";
import "./App.css";

function App() {
  const [count, setCount] = useState(0);

  return (
    <>
      <h1>Vite + React</h1>
      <div className="card">
        <button onClick={() => setCount((count) => count + 1)}>
          count is {count}
        </button>
      </div>
    </>
  );
}

export default App;
```

For the demo, we will need to quickly revise HTML. Ready?

HTML5 revision

- HTML = HyperText Markup Language
- Used for web documents, gives meaning to the content
- **Not** a programming language
- Latest specification of HTML - **HTML5**
  - Current standard
  - New tags for semantic only (`<article>` , `<aside>` , `<footer>`)
  - Elements for multimedia (`<audio>` , `<video>`) - uses shadow dom
  - Since 2014, previous version from 1997

Document structure

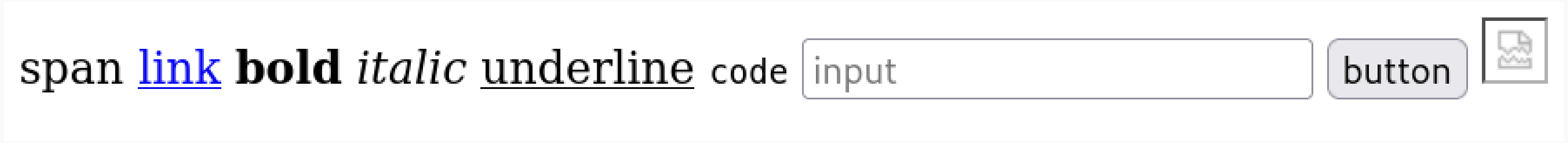
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>A simple HTML document</title>
  </head>
  <body>
    <p>Hello World!</p>
    <p></p>
  </body>
</html>
```

Document structure

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Page title</title>
    <link rel="stylesheet" href="style.css" />
  </head>

  <body>
    <header>
      <h1>Header</h1>
    </header>

    <nav>
      <ul>
        <li><a href="#home">Home</a></li>
        <li><a href="#projects">Projects</a></li>
        <li><a href="#contact">Contact</a></li>
      </ul>
    </nav>

    <main>
      <article>
        <h2>Article heading</h2>
        <p>Lorem ipsum</p>
        <p>Lorem ipsum</p>
      </article>
    </main>

    <footer>
      <p>All rights reversed.</p>
    </footer>
  </body>
</html>
```

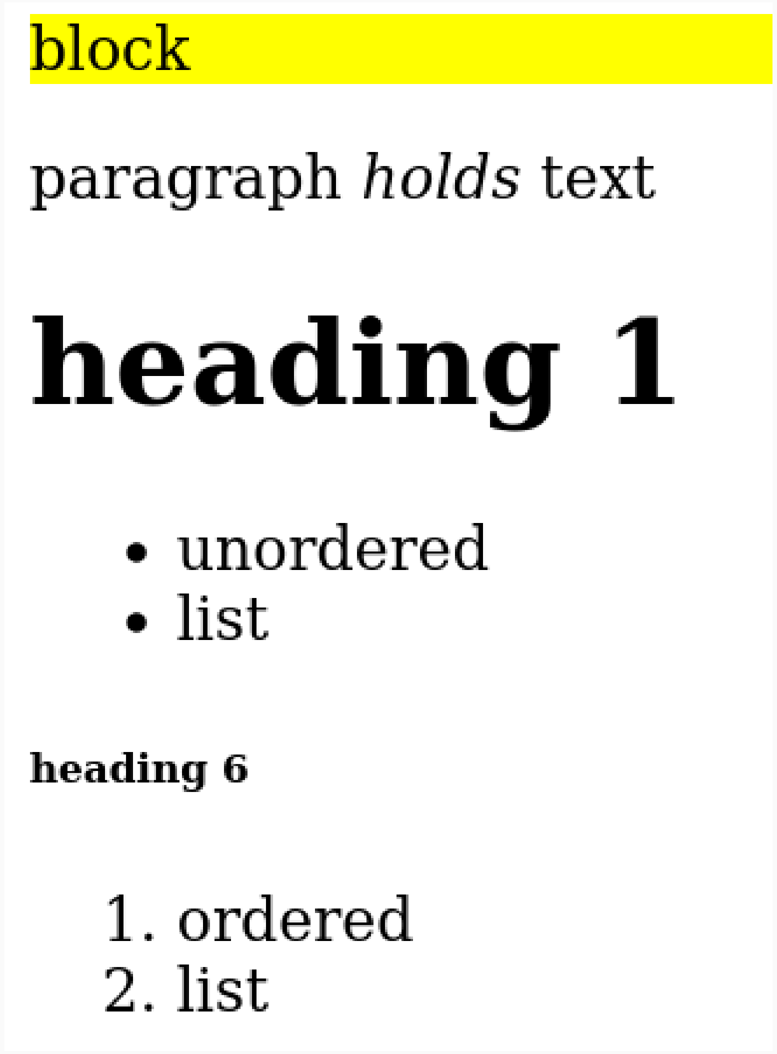
Common inline elements

```html
<span>span</span>
<a href="http://foo.com">link</a>
<b>bold</b>
<i>italic</i>
<u>underline</u>
<code>code</code>
<input placeholder="Insert text here" type="text" />
<button type="button">button</button>
<img src="https://notfound" alt="404" />
```



Block elements

```html
<div>block</div>
<p>paragraph <i>holds</i> text</p>
<h1>heading 1</h1>
<h6>heading 6</h6>
<ul>
  <li>unordered</li>
  <li>list</li>
</ul>
<ol>
  <li>ordered</li>
  <li>list</li>
</ol>
```


block

paragraph *holds* text

heading 1

- unordered
- list

heading 6

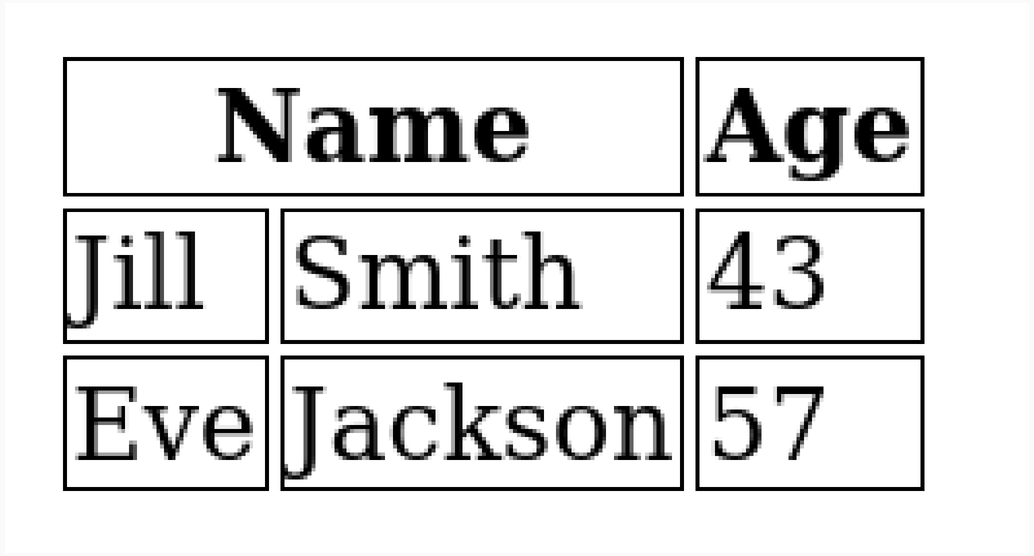
1. ordered
2. list

Block vs inline elements

	Block	Inline
width	parent block	content only
break line	yes	no
children	any	inline or text

Tables

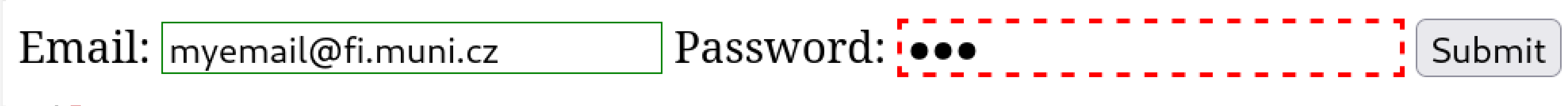
```
<table border="1">
  <thead>
    <tr>
      <th colspan="2" scope="col">Name</th>
      <th scope="col">Age</th>
    </tr>
  </thead>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>43</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>57</td>
  </tr>
</table>
```



Name		Age
Jill	Smith	43
Eve	Jackson	57

Forms

```
<form action="/path/to/handler" method="post">
  <label for="email">Email:</label>
  <input
    type="email"
    name="email"
    id="email"
    placeholder="john@example.com"
    required
  />

  <label for="password">Password:</label>
  <input
    type="password"
    name="password"
    id="password"
    pattern="[0-9a-fA-F]{4,50}"
    required
  />
```



```
</form>
```

Meta tags

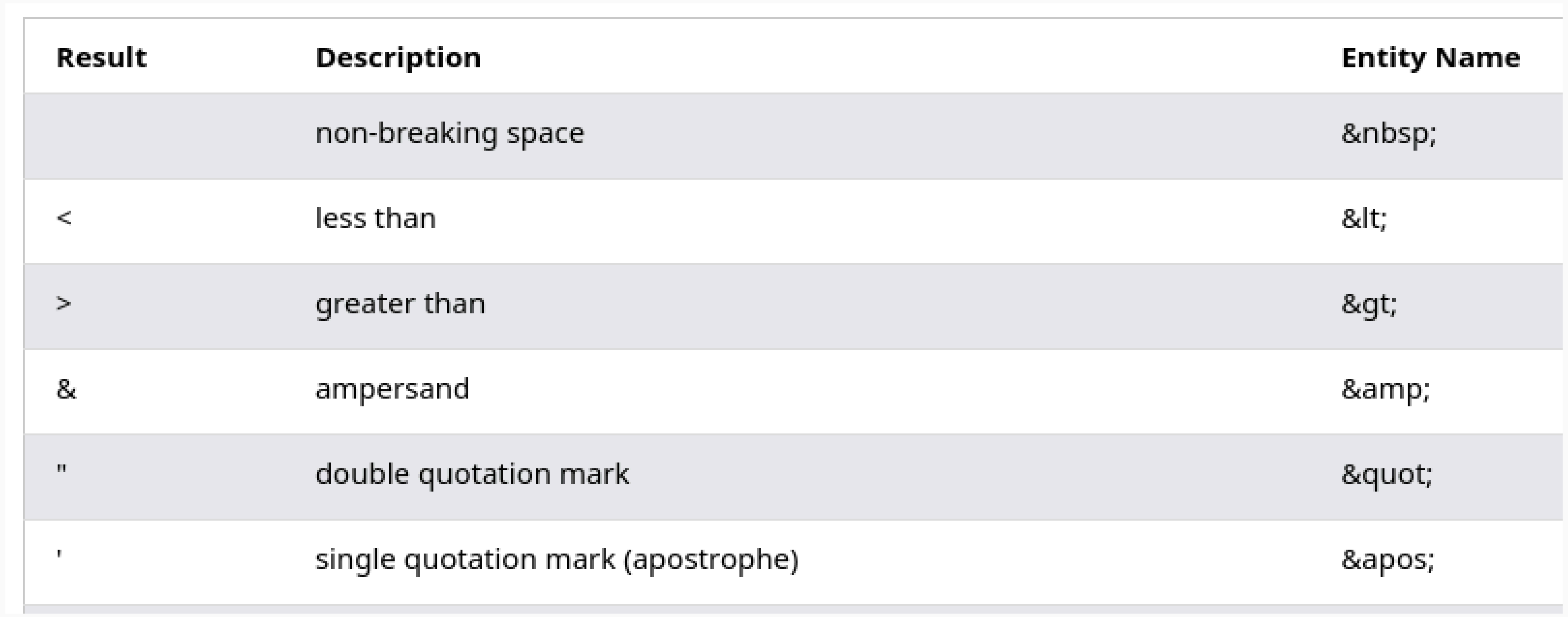
- Not rendered, information for search engines (SEO)
- Page title and description in search results
- `og:` properties for social media previews
- title, favicon in tab, color

```html
<head>
  <title>My page</title>
  <meta charset="UTF-8" />

  <!-- SEO -->
  <meta name="description" content="Free Web tutorials" />
  <meta name="keywords" content="HTML, CSS, JavaScript" />
  <meta name="author" content="John Doe" />

  <!-- Open Graph protocol / social media previews -->
  <meta property="og:title" content="The Rock" />
  <meta property="og:type" content="video.movie" />
  <meta property="og:url" content="https://www.imdb.com/title/tt0117500/" />
  <meta
    property="og:image"
    content="https://ia.media-imdb.com/images/rock.jpg"
  />

  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="icon" type="image/svg" href="http://example.com/image.svg" />
</head>
```

Reserved characters in HTML

If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags.

Character entities are used to display reserved characters in HTML.



Result	Description	Entity Name
	non-breaking space	&nbsp;
<	less than	&lt;
>	greater than	&gt;
&	ampersand	&amp;
"	double quotation mark	&quot;
'	single quotation mark (apostrophe)	&apos;

Lighthouse demo

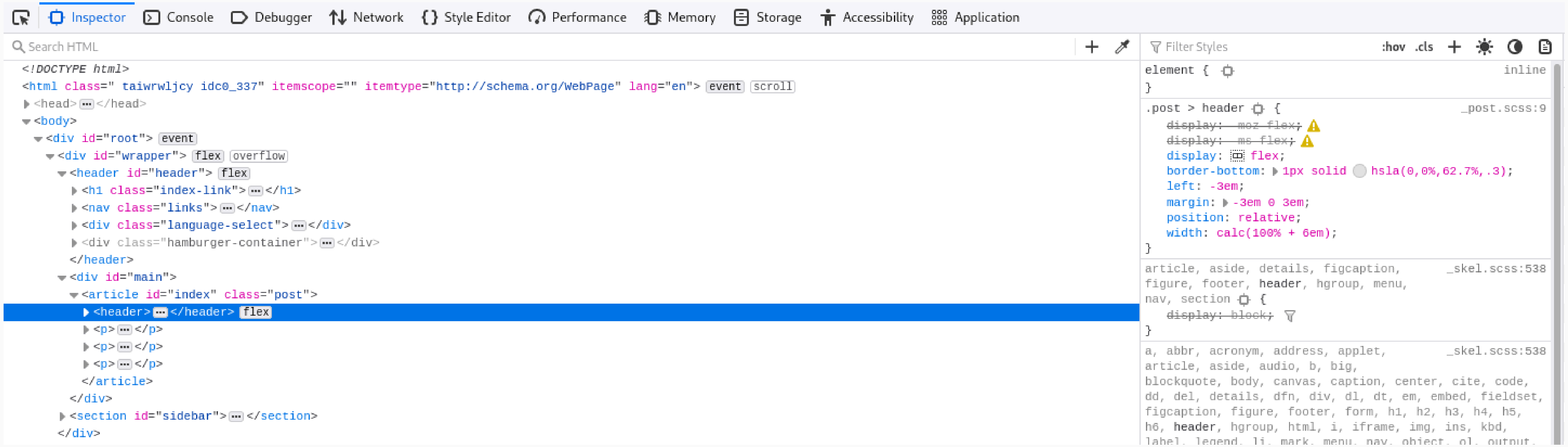
https://pagespeed.web.dev

Documentation you should use

Mozilla developer docs- gold mine for HTML and JavaScript related information
React.js docs - Docs for React - we will talk more about it throughout the semester
Google's web development page - Another great resource. Use in conjunction with Lighthouse to see whether your page is up to standard.

Inspect demo - optional

CSS seminar will use this technique more

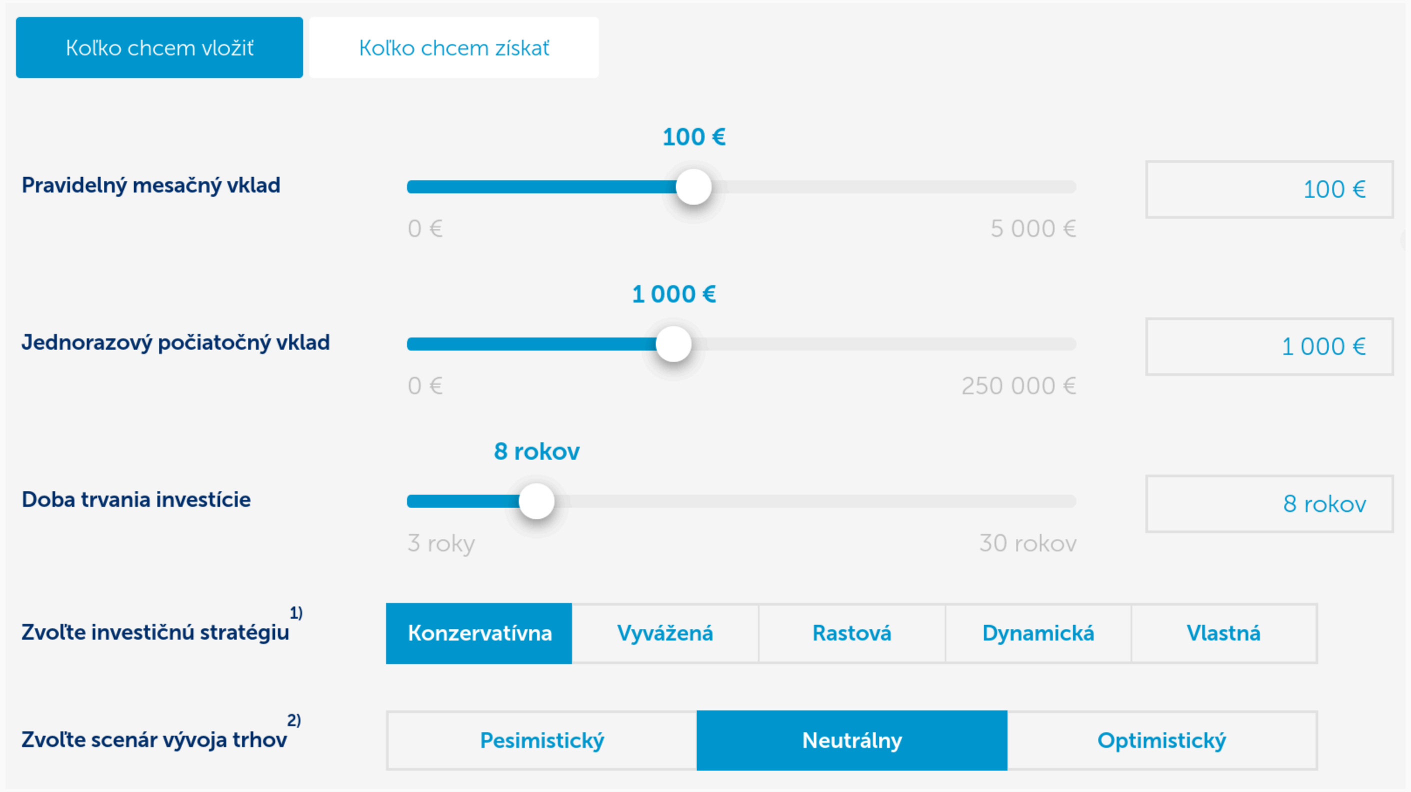


```
<!DOCTYPE html>
<html class=" taiwrwljcy idc0_337" itemscope="" itemtype="http://schema.org/WebPage" lang="en"> event scroll
  <head> ··· </head>
  <body>
    <div id="root"> event
      <div id="wrapper"> flex overflow
        <header id="header"> flex
          <h1 class="index-link"> ··· </h1>
          <nav class="links"> ··· </nav>
          <div class="language-select"> ··· </div>
          <div class="hamburger-container"> ··· </div>
        </header>
        <div id="main">
          <article id="index" class="post">
            <header> ··· </header> flex
            <p> ··· </p>
            <p> ··· </p>
            <p> ··· </p>
          </article>
        </div>
        <section id="sidebar"> ··· </section>
      </div>
```

```
element {                                                  inline
}
.post > header {                                    _post.scss:9
    ~~display: -moz-flex;~~ ⚠
    ~~display: -ms-flex;~~ ⚠
    display: flex;
    border-bottom: 1px solid hsla(0,0%,62.7%,.3);
    left: -3em;
    margin: -3em 0 3em;
    position: relative;
    width: calc(100% + 6em);
}
article, aside, details, figcaption,             _skel.scss:538
figure, footer, header, hgroup, menu,
nav, section {
    ~~display: block;~~
}
a, abbr, acronym, address, applet,               _skel.scss:538
article, aside, audio, b, big,
blockquote, body, canvas, caption, center, cite, code,
dd, del, details, dfn, div, dl, dt, em, embed, fieldset,
figcaption, figure, footer, form, h1, h2, h3, h4, h5,
h6, header, hgroup, html, i, iframe, img, ins, kbd,
label, legend, li, mark, menu, nav, object, ol, output,
```

Demo – create markup for this simple form

Don't forget to use correct **semantic elements**! We advise you to open the slides on your PC to not have to rely on the projected image from your tutor. **No styling is required! (yet)**



[✓ Koľko chcem vložiť] Koľko chcem získať

Pravidelný mesačný vklad — 100 € (0 € – 5 000 €) — 100 €

Jednorazový počiatočný vklad — 1 000 € (0 € – 250 000 €) — 1 000 €

Doba trvania investície — 8 rokov (3 roky – 30 rokov) — 8 rokov

Zvoľte investičnú stratégiu 1) — [✓ Konzervatívna] Vyvážená Rastová Dynamická Vlastná

Zvoľte scenár vývoja trhov 2) — Pesimistický [✓ Neutrálny] Optimistický

- Identify the elements
- Work your way from the whole form to individual elements making it up
- Split it into components -> think about the difference between a component and an HTML element

Demo - create markup for this simple form

Bonus:

- Add another field where the user can select multiple elements.
- Add a field where user can write their email into.
- [Advanced]: **Use flexbox/grid** to get the **layout** close to the one shown on this picture*

** Your tutor might show this to you as a sneak peek into layouting, which will be one of the two major parts of the next week's seminar (the other being styling).*

Questions?

Thank you for your attention!