

729G87 Interaction Programming

Lecture 1 - Intro

Philipp Hock, PhD
philipp.hock@liu.se

Overview

- About the course
- Front-end development
- HTML/CSS/JavaScript
- Development Tools

- Interaction programming
 - Concepts (Component based interaction)
 - Prototyping

Objectives

- Understanding the world from a user interface (UI) perspective
- Identify and describe components used in a UI
- Implement graphical interactivity using state-of-the-art technologies
- Implement a UI component given a description
- Learn basics (vanilla) of front-end web-development

Prerequisites

- Curiosity
- Proactivity

- Basic programming skills:
 - abstraction
 - flow control
 - data types
 - OOP (quick recap)

Literature

- "Google"
- <https://www.ida.liu.se/~729G87/>
- <https://stackoverflow.com/>
- <https://developer.mozilla.org/en-US/docs/Web>

Staff

- Examiner/course leader
Philipp Hock
- Teaching assistants
Charlie Simonsson
Emma Mainza Chilufya
- Course administrator
Veronica Gunnarson

Changes since last year

- Lectures
 - More practical examples
 - More basics
- Assignments
 - Assignment 1 modified
 - starts with codecademy for css
 - Switched 1st and 3rd assignment
 - Assignment 4 modified
 - Template & description updated
 - More guidance in 1st and 2nd assignment
- Project
 - Reference analysis added
- Better integration between repository & assignments
 - Folder structure and proper linking in the repository from start
 - All templates for assignments in repository

Course page

Home base:

<https://www.ida.liu.se/~729G87/>

Assignments

- Course website has all infos:
 - <https://www.ida.liu.se/~729G87>
- Work in pairs!
- Use Gitlab & Webreg
- No Lisam!
- Fridays: Present your results (mandatory)

Sign up!

- Sign up for LAB1 and PRO1

<https://www.ida.liu.se/webreg3/729G87-2023-1/LAB1>

<https://www.ida.liu.se/webreg3/729G87-2023-1/PRO1>

- Sign up using the same pair groups for both modules

Course structure

CW 46-50
Assignments &
Project preparation

CW 51 – 2
Project

Workload

- <https://www.ida.liu.se/~729G87/>
 - [timetable](#)
- 9 Weeks, 6 ECTS = 180h => ~20h/ week
- Scheduled course sessions 4-10h/week
- 10 - 16h/week on your own

Grades: assignments

- Assignments
 - **Assignment 1 - 5:** 1 - 2 points per group submission
 - total: 10 points
 - 5 Points to pass
 - 9 for A grade
 - Passing
 - "Score 1 point in each assignment to pass"
- LAB1. Grades
 - A: ≥ 9 points
 - B: ≥ 8 points
 - C: ≥ 7 points
 - D: ≥ 6 points
 - E: ≥ 5 points
- Early assignments are easier
- Later assignments are more creative

Grades: Project

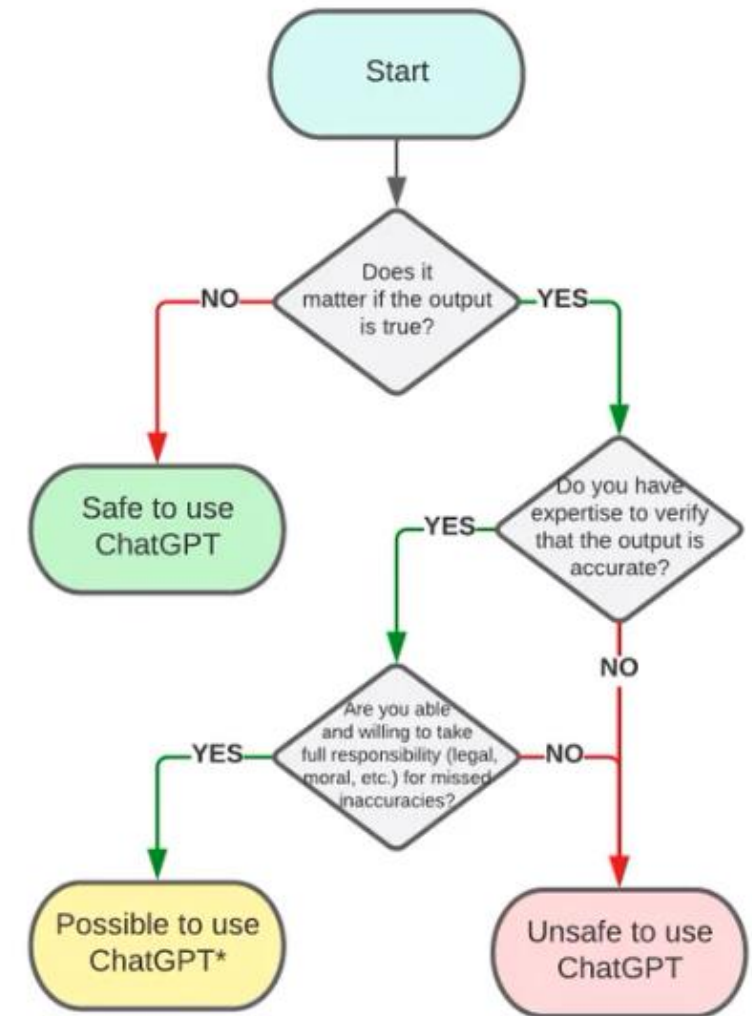
- Project deliverables
 - Specification (1-3 points)
 - Implementation (1-3 points)
- PRO1
 - A: 6 points
 - B: 5 points
 - C: 4 points
 - D: 3 points
 - E: 2 points (one from each deliverable)
- To pass PRO1, both deliverables must be passed.

Course Grade

- Rounded average of LAB1 and PRO1
- LAB = A & PRO = D
 - $\rightarrow (5+2)/2 = 3,5 \rightarrow 3 \rightarrow C$

Is it safe to use ChatGPT for your task?

Aleksandr Tiulkanov | January 19, 2023



* but be sure to verify each output word and sentence for accuracy and common sense

ChatGPT

- Not forbidden!
- BUT!
 - Do not just copy-paste code from ChatGPT
 - Counts as plagiarism
 - Reference when and how ChatGPT is used
 - Obvious plagiarism will have serious consequences
 - Do not verify output using ChatGPT
 - Can be helpful finding bugs
 - Can be helpful getting started
 - Can be helpful understanding things
 - Be sceptic!
 - Often solves the problem partially
 - Often does not delivers elegant/best solution

Assignments

- 5 Assignments
 - HTML & CSS
 - JavaScript
 - Advanced JS
 - HTML Components
 - Creative assignment (no groups)
- 2 – 4 computer lab sessions per assignment
 - Supervised
 - Semi-supervised
 - Unsupervised
- Done in groups of 2
- signup via webreg: <https://www.ida.liu.se/webreg3>

Submissions

- Mind the deadlines!
 - <https://www.ida.liu.se/~729G87/about/timetable/>
- Publish your assignments!
 - <https://gitlab.liu.se/729g87-ht23>
 - Git, Gitlab & CI/CD pipeline
 - All exercises in one repository
 - If not explicitly stated, javascript frameworks (e.g., jQuery, React,...) are not allowed
 - CSS Reset must be used: <https://meyerweb.com/eric/tools/css/reset/>
 - Use Javascript event-handler, not HTML attributes to call js
 - More in later lectures

Project

- Deliverables
 - Specification (1-3 points) [1 = pass]
 - Implementation (1-3 points) [1 = pass]
- Presentation necessary to pass
- Supervision regarding specification & implementation
- All deadlines online: <https://www.ida.liu.se/~729G87/>
 - Course page counts (not timeedit)

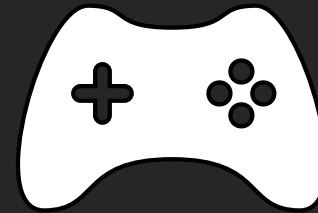
Choose one



web shop



Psychological
experiment

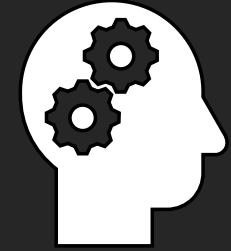


Browser Game



Web shop

- Browse items and interact with shopping cart
- Functional requirements:
 - browse items
 - items have more than one picture
 - items have a description
 - add/remove/edit items in shopping cart
 - view shopping cart
- Example subcategories: clothing store, food store ...



Psychological experiment

- Get inspired by <https://www.psychtoolkit.org/experiment-library/>
- Classical experiments are
 - Stroop task
 - N-back task
 - Go/No-go task
 - ...
- Such experiments are intended to create certain stimuli
 - Increase cognitive load
 - Induce stress
 - Increase memory workload
 - ...
- In each experiment, performance is measured and later analyzed
- Experiment must include a performance evaluation (statistics of how participants performed on the task)
- Difficulty to implement differ
 - Get feedback from teaching assistant/me before starting

Purple

no penalty

4s

Penalty

8s

Black



Purple



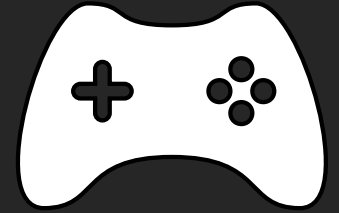
Blue

Yellow

Trials

5

Time 0:00*Time penalty* 0:00*Total time* 0:00



Game

- Games can be very simple but can become very complex to implement!
 - Initial feedback from teaching assistant/me is mandatory!
- Some games require server-side logic.
 - You have no access to custom server-side logic
 - Do not implement such games
- Probably the most complex choice
 - Rewarded by lots of fun implementing and playing
- Classic games are
 - Pong
 - Snake
 - Blackjack
 - Breakout
 - Tetris
 - You can also create your own game
- Add a custom feature that is not in the original game

Next steps

- Wireframe specification
- Implementation
- Presentation

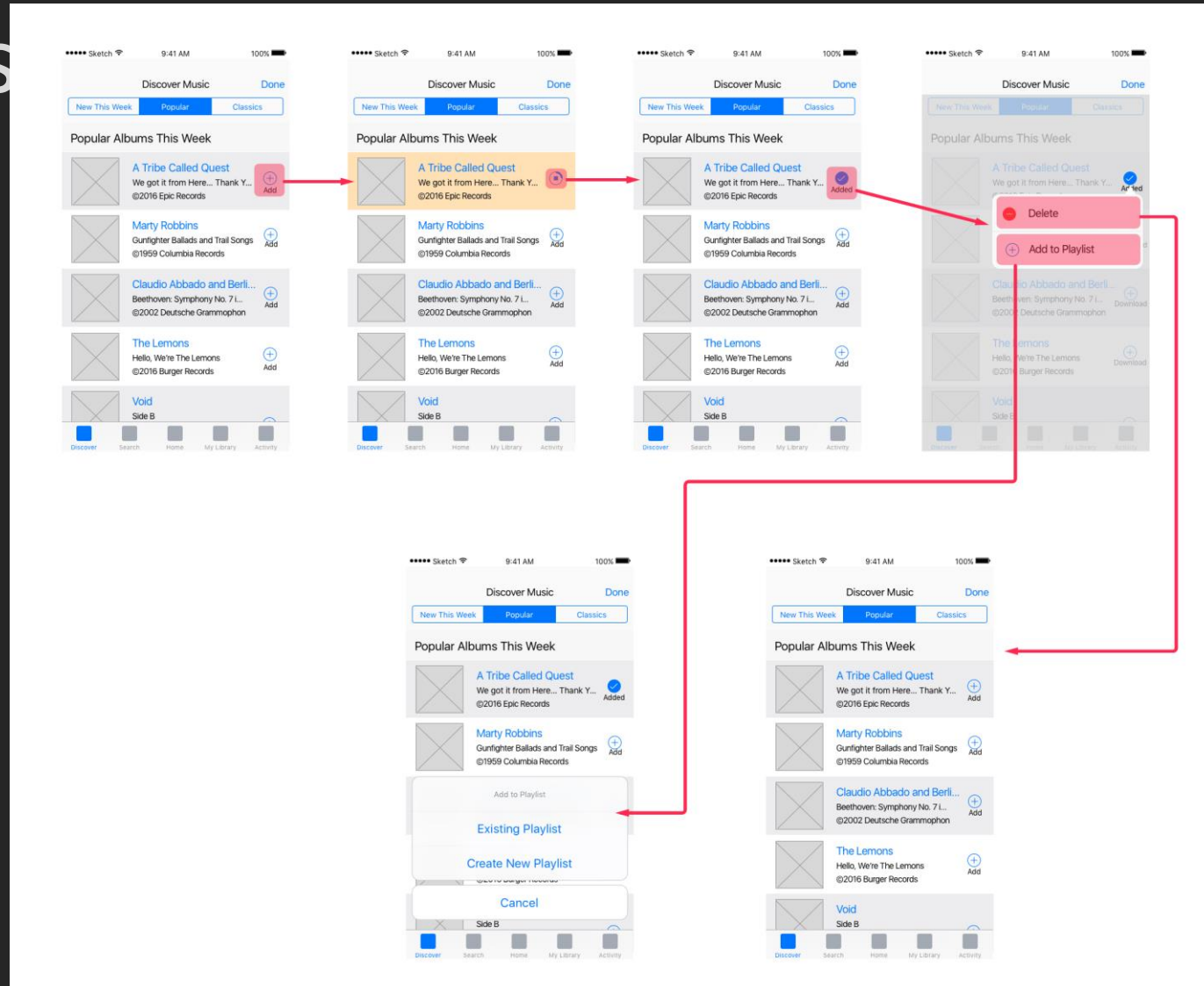
Reference Analysis

- Find references that has similar ideas/concepts
 - Add screenshots and links
- See description in on course page
 - <https://www.ida.liu.se/~729G87/>

Wireflow

- Create wireflow sketches for all functionality related to the functional requirements of your chosen category
- Consists of
 - visual prototype
 - Interactions visualized with arrows
- Detailed instructions on the course page
- More in dedicated lecture

Examples



<https://alvarotrigo.com/blog/wireflows/>

<https://www.nngroup.com/articles/wireflows/>

Git & Gitlab

Version control | shared working | publishing

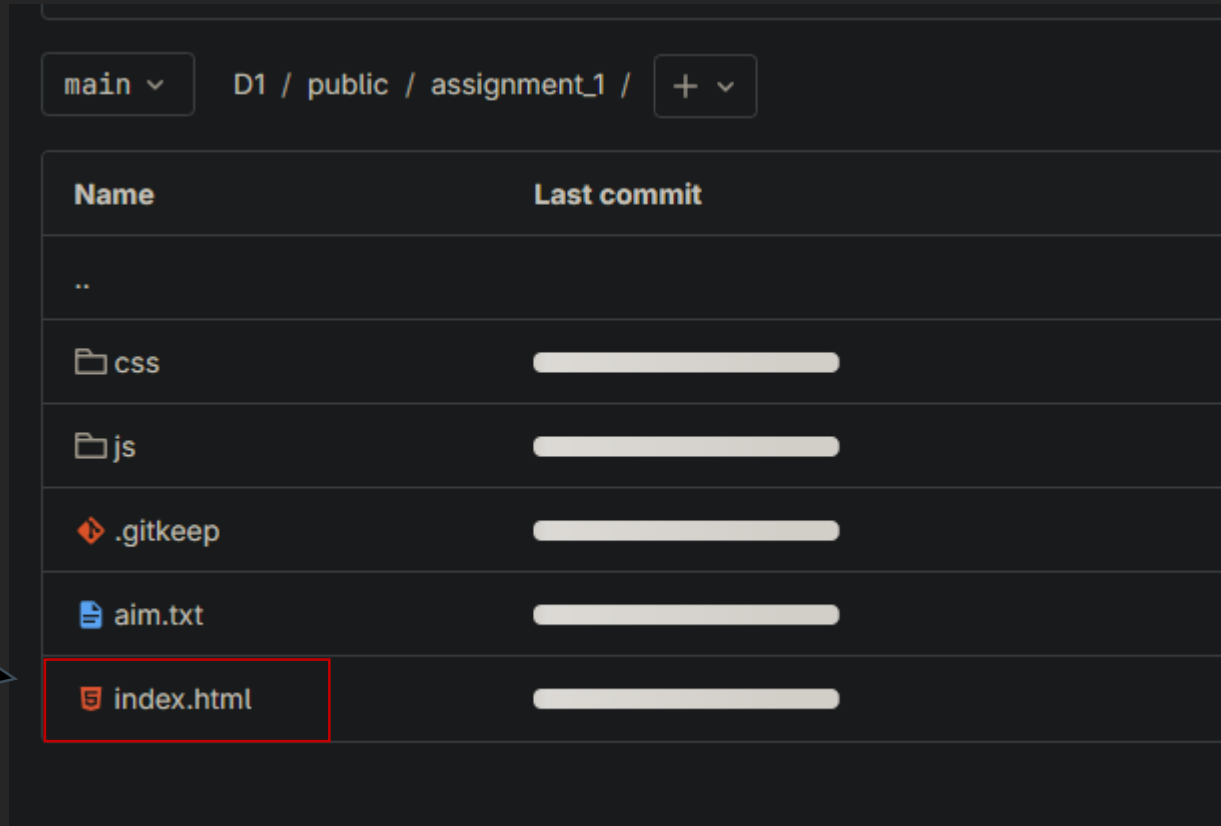
Publish content @ Liu.se

- <https://www.ida.liu.se/~729G87/>
- Gitlab makes ./public folder accessible
- Depending on the project policies
- Basic template will be provided
 - Automatically publishes assignment
 - Assignments are in subfolder
 - Test your published code
 - Local code sometimes works but published does not
 - Mostly references/links issues (external stylesheets/javascript files)

URL & index.html

<https://729g87.gitlab-pages.liu.se/submissions-group-xx-template/>

If a link has a folder as target instead of a file, the server will return the index.html, if it is present. If not, a 404 page will be displayed



The screenshot shows a GitLab repository interface. At the top, there is a breadcrumb navigation: "main" (with a dropdown arrow), "D1 / public / assignment_1 /" (with a plus and dropdown arrow). Below this is a table with two columns: "Name" and "Last commit". The table lists the following items:

Name	Last commit
..	
css	
js	
.gitkeep	
aim.txt	
index.html	

The "index.html" row is highlighted with a red border. Each row in the table has a corresponding progress bar in the "Last commit" column.

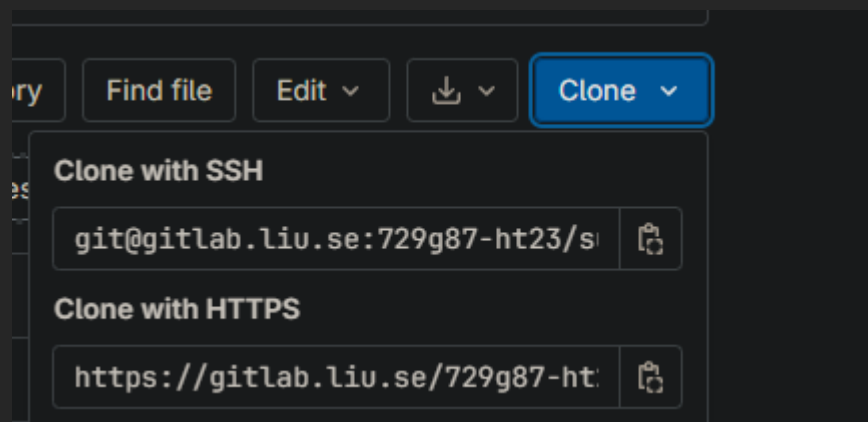
Git & Assignment workflow

- Clone your project repo
 - <https://gitlab.liu.se/729g87/HT2024>
 - Clone your team repo (e.g. E1)
 - All assignment in public folder



Clone (download) a remote repo

- Go to gitlab repo page (e.g. <https://gitlab.liu.se/729g87/HT2023/D1>)
- Remote repo
 - In a terminal, type:
 - `git clone <url>`
 - You can clone the ssh version or the https version
 - ssh version requires ssh-key



Local repo

- After cloning, the repository is on your local computer
- After making changes, you add your changes to the stage
- Stage = “a list” of all files that should be committed
- `> git add filename`
 - Or: `> git add -A .` // adds all files that has been changed
- Commit = save the current version of staged files as snapshot

- Then you need to upload your changes to gitlab:
- `> git push`

- Important:
- After cloning your repository, you need to pull the newest version before you start working!
- `> git pull`

Workflow

- > git clone (once)

- Then each time you work on your repo:
- > git pull (do not forget this!)
- Make changes to your assignment
- > git add -A .
- > git commit -m “a meaningful message what you did”
- > git push

Git

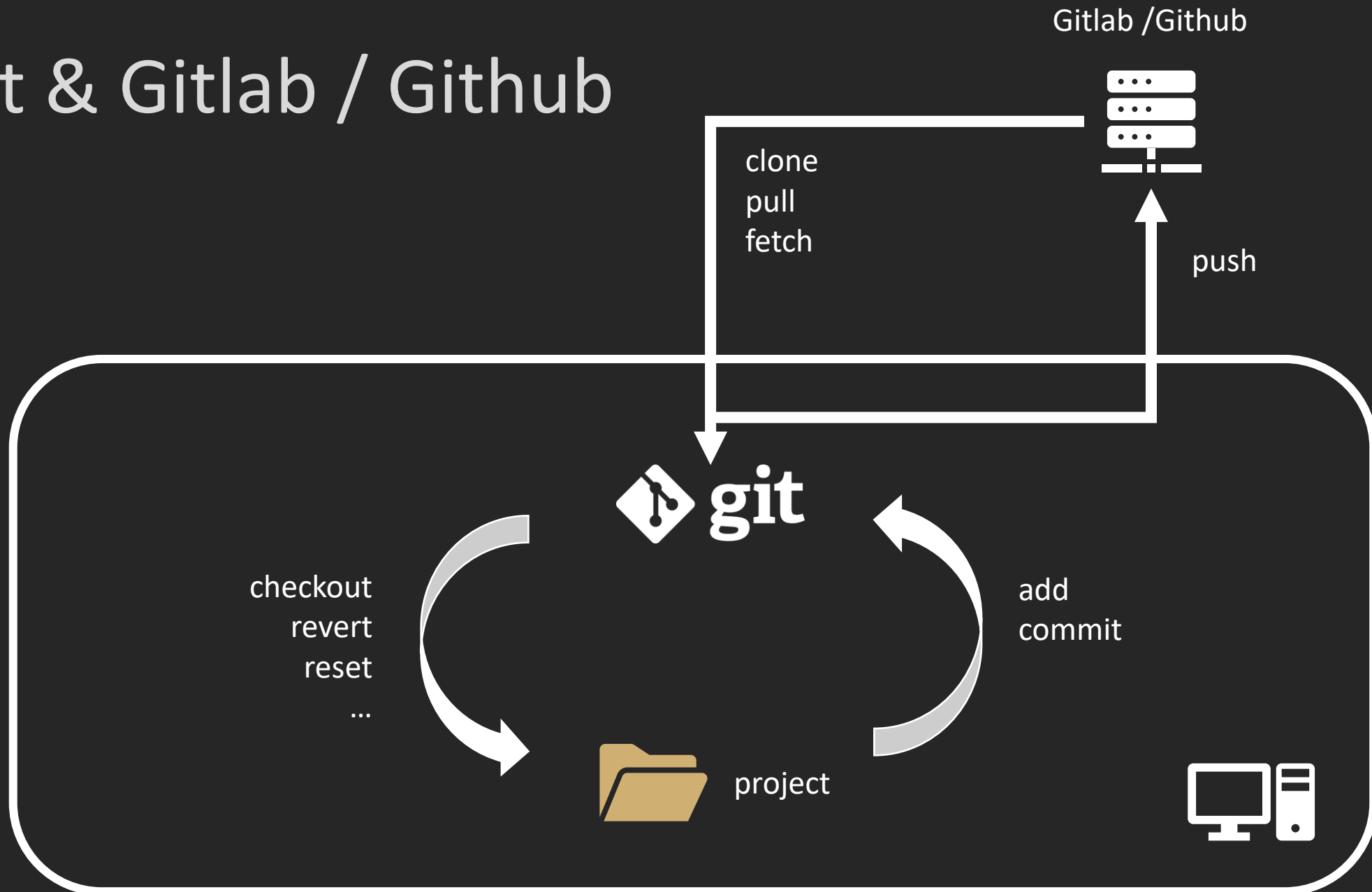
- If done right, all you need is
 - git clone
 - git add
 - git commit
 - git push
 - git pull

- Also has GUIs
 - <https://git-scm.com/downloads/guis>
 - <https://gitextensions.github.io/>
 - <https://murmele.github.io/Gittyup/>

From gitlab to your computer and back

- `git clone` = copies remote repo to local computer
- `git pull` = gets newest changes from remote repo
 - `> git pull origin main` (sometimes main, not master)
 - Always do `git pull` before working -> avoid merge conflicts
- `git push` = upload local changes to remote repo
 - `> git push origin main` (sometimes master, not main)
 - You can only push if you add and commit your work locally!

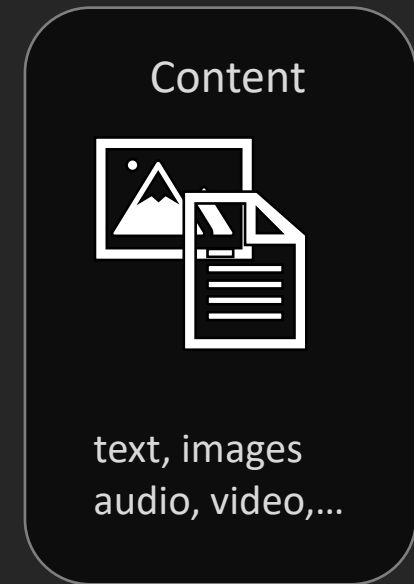
Git & Gitlab / Github



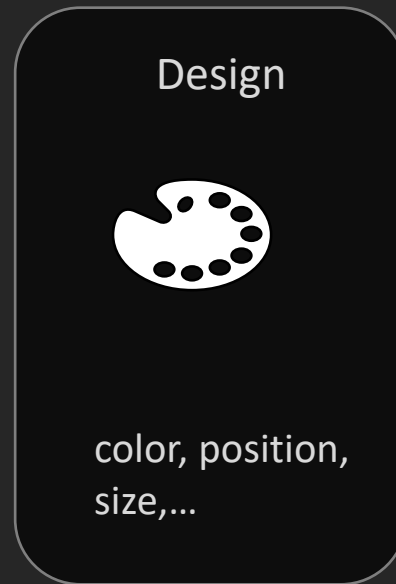
Recap

- Quick demo on gitlab

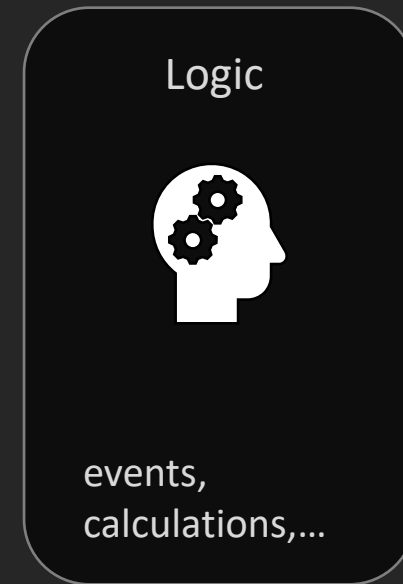
Basic HTML Eco System



HTML



CSS



JavaScript

In this lecture

- Devtools
- HTML
- CSS
 - Import
 - Selectors
 - Layout
 - Positioning
 - Flexbox
- Optional
 - Responsiveness
 - Frameworks
 - Animations

Development tools

Browser DevTools

F12

Inspektor Konsole Debugger Netzwerkanalyse Stilbearbeitung Laufzeitanalyse Speicher Web-Speicher Barrierefreiheit Anwendung

HTML durchsuchen

```
<!DOCTYPE html>
<!--
Copyright 2012 Mozilla Foundation Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License
at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES
OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. Adobe CMap resources are covered by their own
copyright but the same license: Copyright 1990-2015 Adobe Systems Incorporated. See https://github.com/adobe-type-tools/cmap-resources
-->
<html dir="ltr" mozdisallowselectionprint="" style="--viewer-container-height: 386px;">
<head>
</head>
<body tabindex="1">
  <div id="outerContainer">
    <!--outerContainer-->
    <div id="printContainer"></div>
  </body>
</html>
```

Stile filtern

```
Element { }
body {
  background-color: var(--body-bg-color);
  scrollbar-color: var(--scrollbar-color) var(--scrollbar-bg-color);
}
html, body {
  height: 100%;
  width: 100%;
}
* {
  padding: 0;
  margin: 0;
}
Geerbt von html
Element {
  --viewer-container-height: 386px;
}
@media (prefers-color-scheme: dark)
:root {
  --main-color: rgba(249, 249, 250, 1);
  --body-bg-color: rgba(42, 42, 46, 1);
  --progressBar-color: rgba(0, 96, 223, 1);
  --progressBar-bg-color: rgba(40, 40, 43, 1);
  --progressBar-blend-color: rgba(20, 68, 133, 1);
  --scrollbar-color: rgba(121, 121, 123, 1);
  --scrollbar-bg-color: rgba(35, 35, 39, 1);
  --toolbar-icon-bg-color: rgba(255, 255, 255, 1);
  --toolbar-icon-hover-bg-color: rgba(255, 255,
```

Layout Berechnet Änderungen Kompatibilität

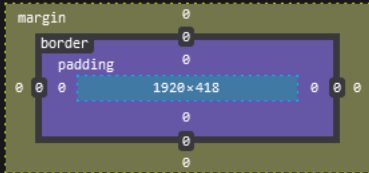
Flexbox

Flex-Behälter oder -Element auswählen, um fortzufahren.

Raster

Es wird kein CSS-Raster auf dieser Seite verwendet.

Box-Modell



margin 0

border 0

padding 0

1920x418

0 0 0 0

0 0 0 0

0 0 0 0

0 0 0 0

1920x418 static

Box-Modell-Eigenschaften

box-sizing	content-box
display	block
float	none
line-height	normal
position	static
z-index	auto

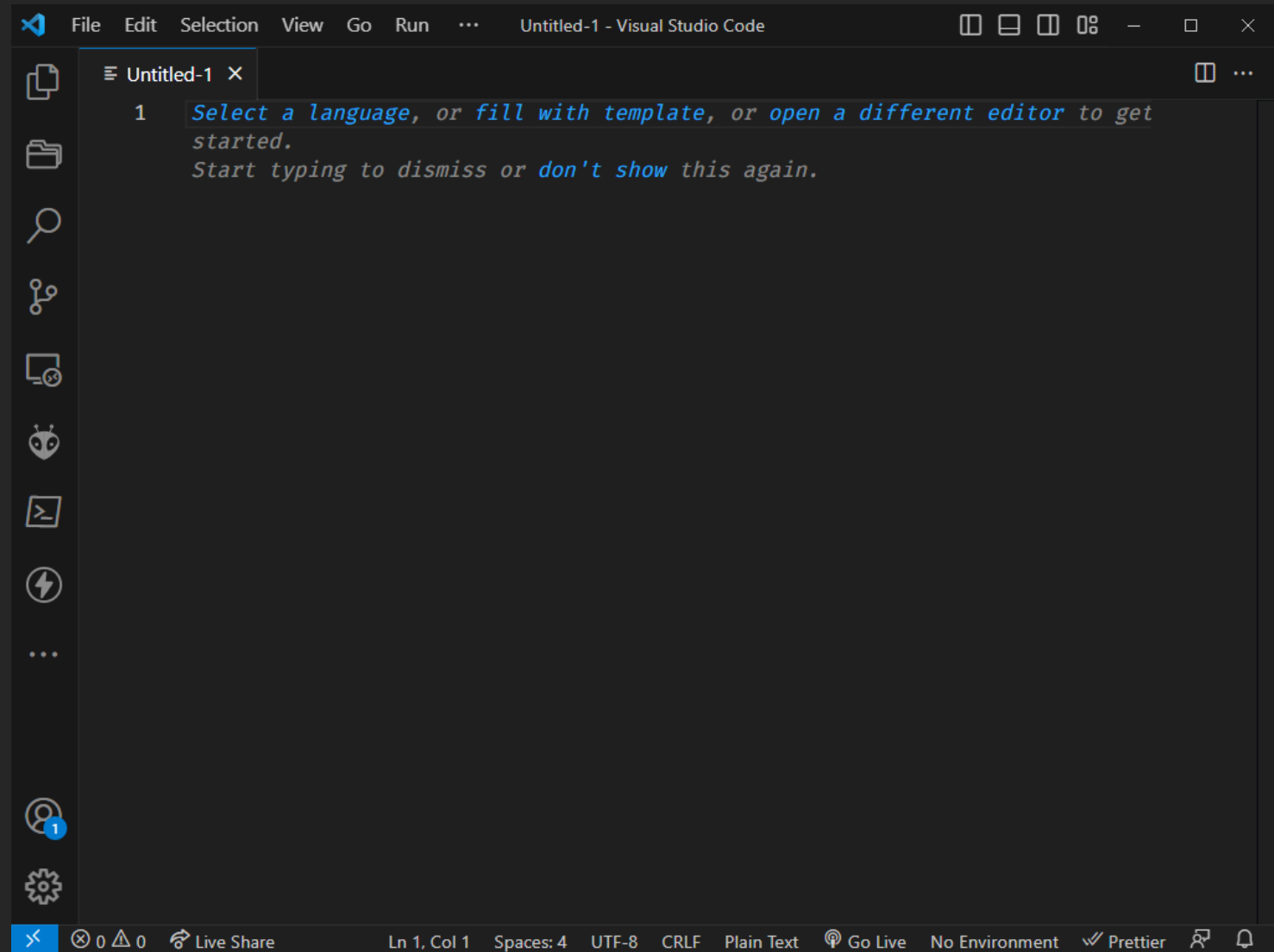
html > body

Development tools

VS Code + plugins

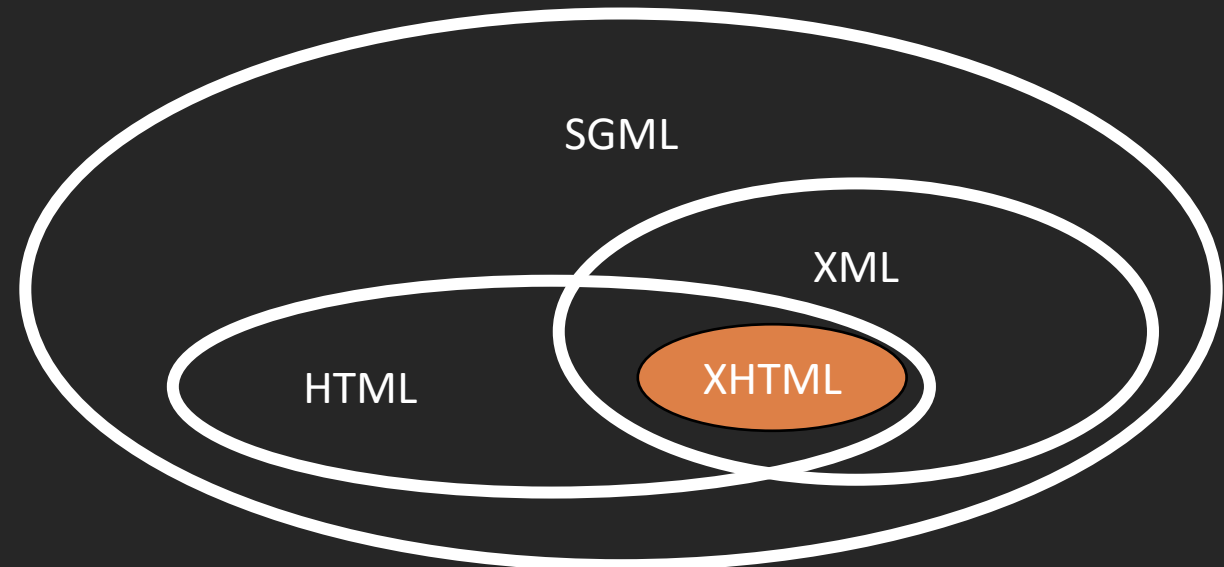
Useful plugins:

- Live Server
- HTMLHint
- Prettier
- Live Share
- Project Manager

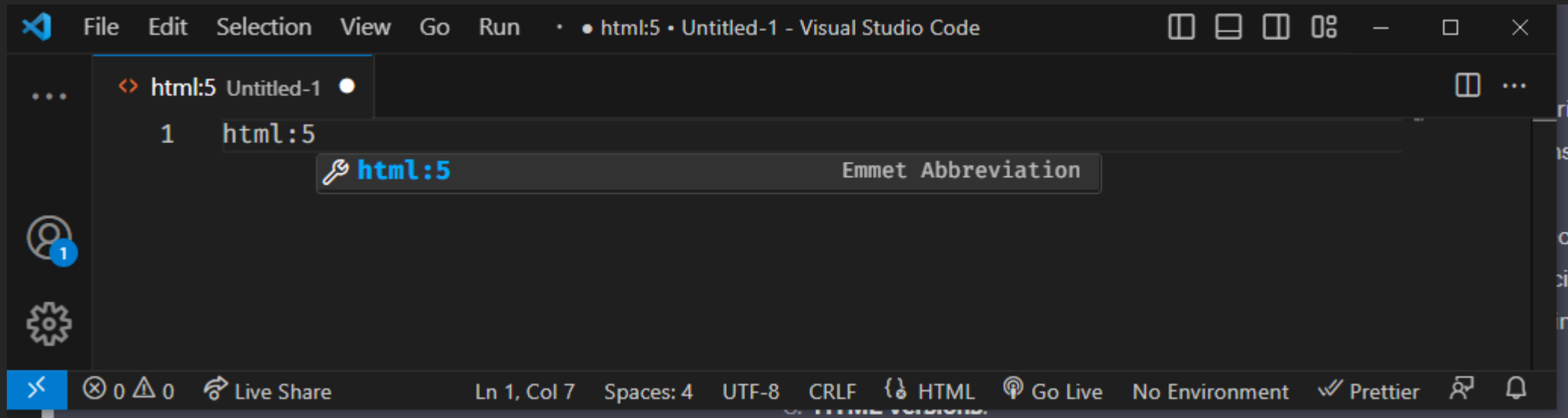


HTML

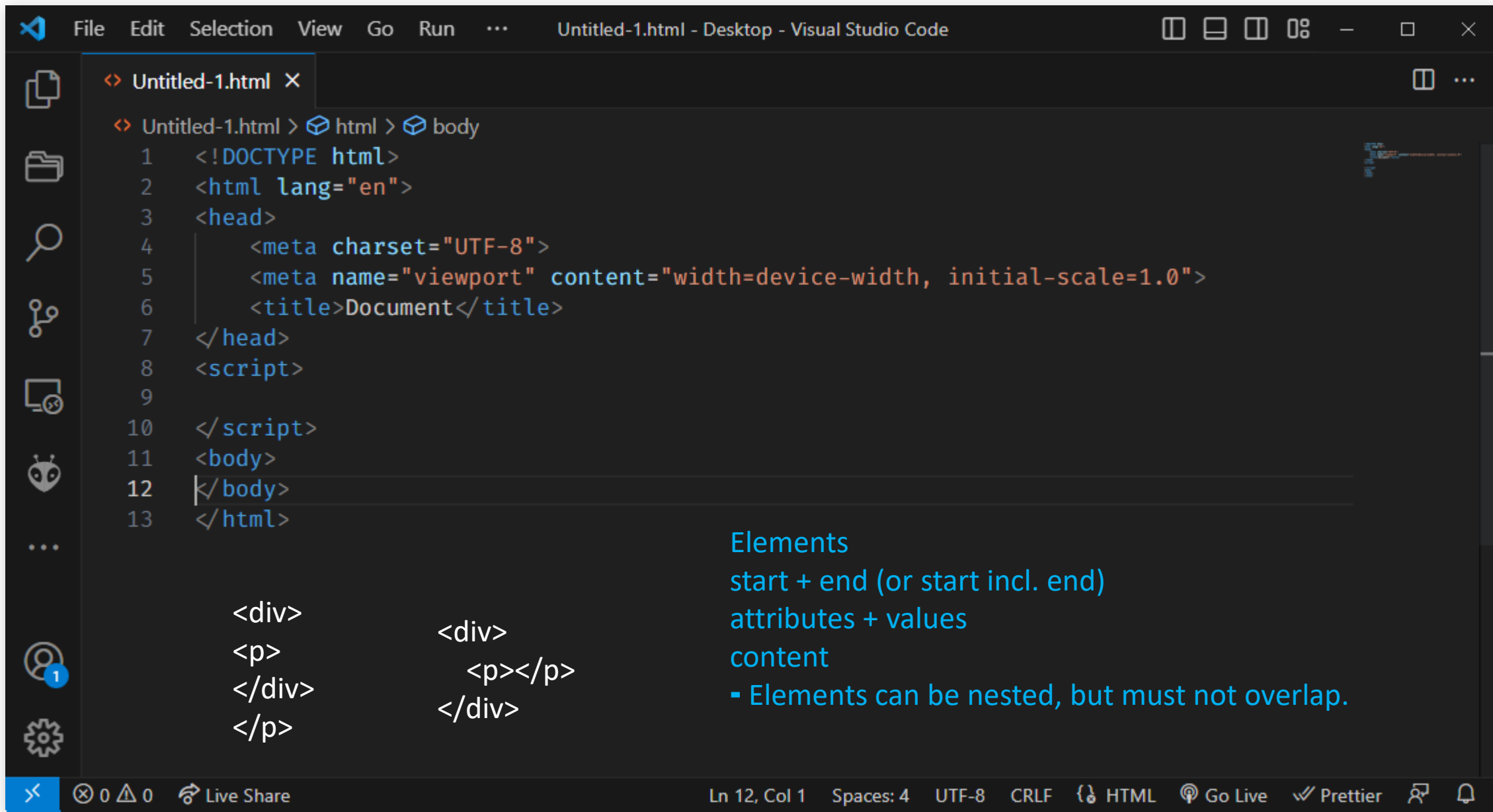
- HTML - Hypertext Markup Language
- Text markup
- Historically used to specify both form and structure, now only structure.
- Evolved from Standard Generalized Markup Language (SGML; ISO 8879:1986)
- SGML -> Meta Markup Language
 - XML -> subset of SGML
 - XHTML -> XML application
 - XHTML -> stricter HTML



HTML Structure



Hello world



```
File Edit Selection View Go Run ... Untitled-1.html - Desktop - Visual Studio Code
<> Untitled-1.html X
  <> Untitled-1.html > html > body
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <script>
9
10 </script>
11 <body>
12 |</body>
13 </html>
```

Elements
start + end (or start incl. end)
attributes + values
content

- Elements can be nested, but must not overlap.

```
<div>
<p>
</div>
</p>

<div>
  <p></p>
</div>
```

Ln 12, Col 1 Spaces: 4 UTF-8 CRLF HTML Go Live Prettier

Hello world

Document Object Model (DOM)

```
Inspektor  Konsole  Debugger  Netzwerkanalyse  Stilbearbeitung  Laufzeitanalyse  Speicher  Web-Speicher  Barrierefreiheit  Anwendung

HTML durchsuchen  +  ✎

<!DOCTYPE html>
<html lang="en" <event>
  <div style="border-block: initial !important; border-inline: initial !important; position: fixed !important;"></div>
  <head> </head>
  <body>
    <p>Hello world</p>
  </body>
</html>
```

```
html (root)
├─ head
│  ├─ meta
│  └─ title
└─ body
   ├─ h1
   ├─ p
   │  └─ strong
   ├─ ul
   │  ├─ li
   │  ├─ li
   │  └─ li
   ├─ ol
   │  ├─ li
   │  ├─ li
   │  └─ li
   └─ a
      └─ img
```

Anatomy of a `<tag></tag>`

- Opening tag `<tagname>`
- Closing tag `</tagname>`

`<nesting>`

`<inner></inner>`

`</nesting>`

No closing tag (e.g., `img-tag`)

` == ` (strictly: `` is more correct)

Anatomy of a `<tag></tag>`

- Attributes

`<tag attribute='value'></tag>`

or

`<tag attribute="value"></tag>`

possible but bad:

`<tag attribute=value></tag>`

Example: Link:

`google`

Anatomy of a `<tag></tag>`

- Attributes can sometimes be shortened:
 - `<script defer></script>` == `<script defer="defer"></script>`
 - `<video controls></video>` == `<video controls="controls"></video>`

HTML & Semantic

<header>
 <nav>
 <section>
 <article>
 <aside>
 <figcaption>
 <figure>
 <footer>

```

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

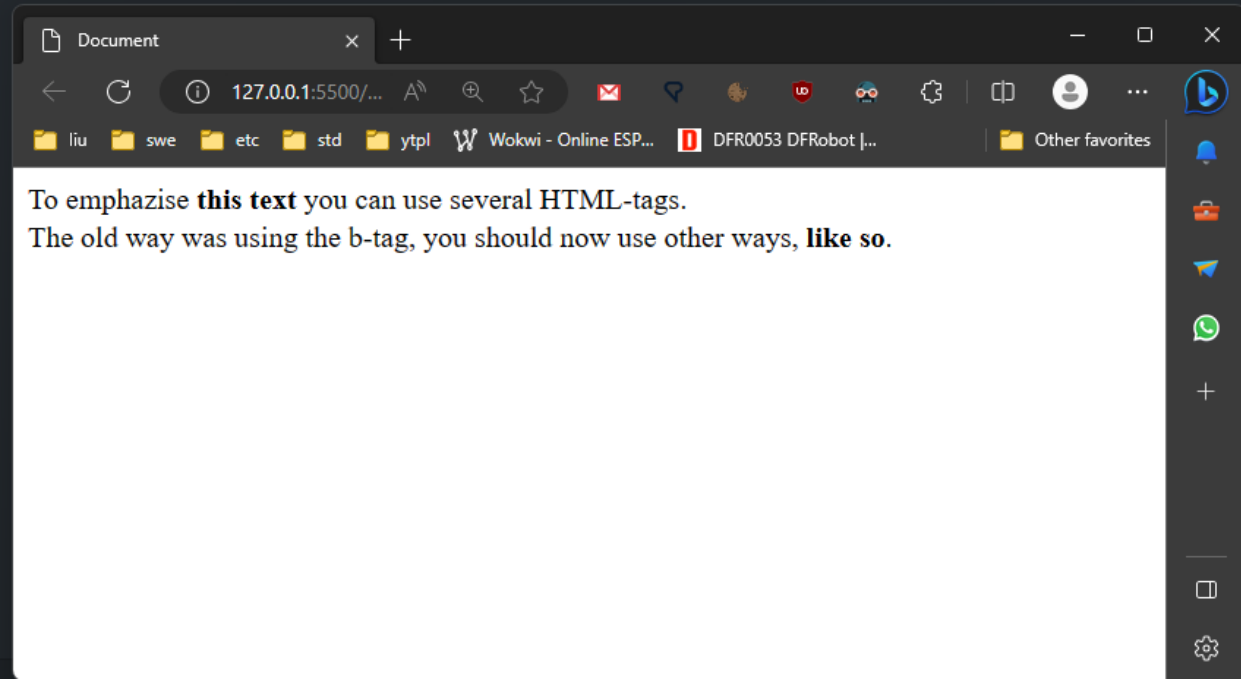
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <!--<link rel="stylesheet" href="css/reset.css">-->
  <link rel="stylesheet" href="css/layout.css">
  <link rel="stylesheet" href="css/style.css">
</head>
<script src="js/main.js" defer>
</script>

<body>

  To emphasize <b>this text</b> you can use several HTML-tags. <br>
  The old way was using the b-tag, you should now use other ways, <strong>like so</strong>.

</body>
</html>
  
```

to separate style from semantic



HTML legacy tags



what are some html-tags that should not be used, show alternatives, present in a table, be concise



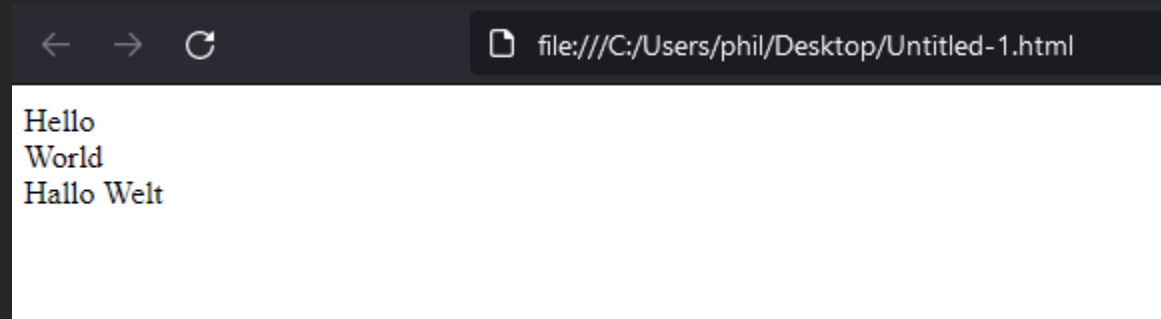
Deprecatd/Obsolete Tag	Alternative(s)
<code><center></code>	CSS <code>text-align: center;</code>
<code></code>	CSS <code>font-family</code> , <code>font-size</code>
<code><strike></code>	CSS <code>text-decoration: line-through;</code>
<code><s></code>	CSS <code>text-decoration: line-through;</code>
<code><u></code>	CSS <code>text-decoration: underline;</code>
<code></code>	CSS <code>font-weight: bold;</code>
<code><i></code>	CSS <code>font-style: italic;</code>
<code><frame></code>	Use <code><iframe></code> for embedding content
<code><frameset></code>	Use modern layout techniques (e.g., Flexbox, CSS Grid)
<code><applet></code>	Use modern web technologies (e.g., JavaScript)
<code><marquee></code>	CSS animations or JavaScript for scrolling text



Remember, it's best practice to use modern HTML and CSS techniques for creating and styling web content. Always keep accessibility and compatibility in mind when using HTML tags and styling properties.

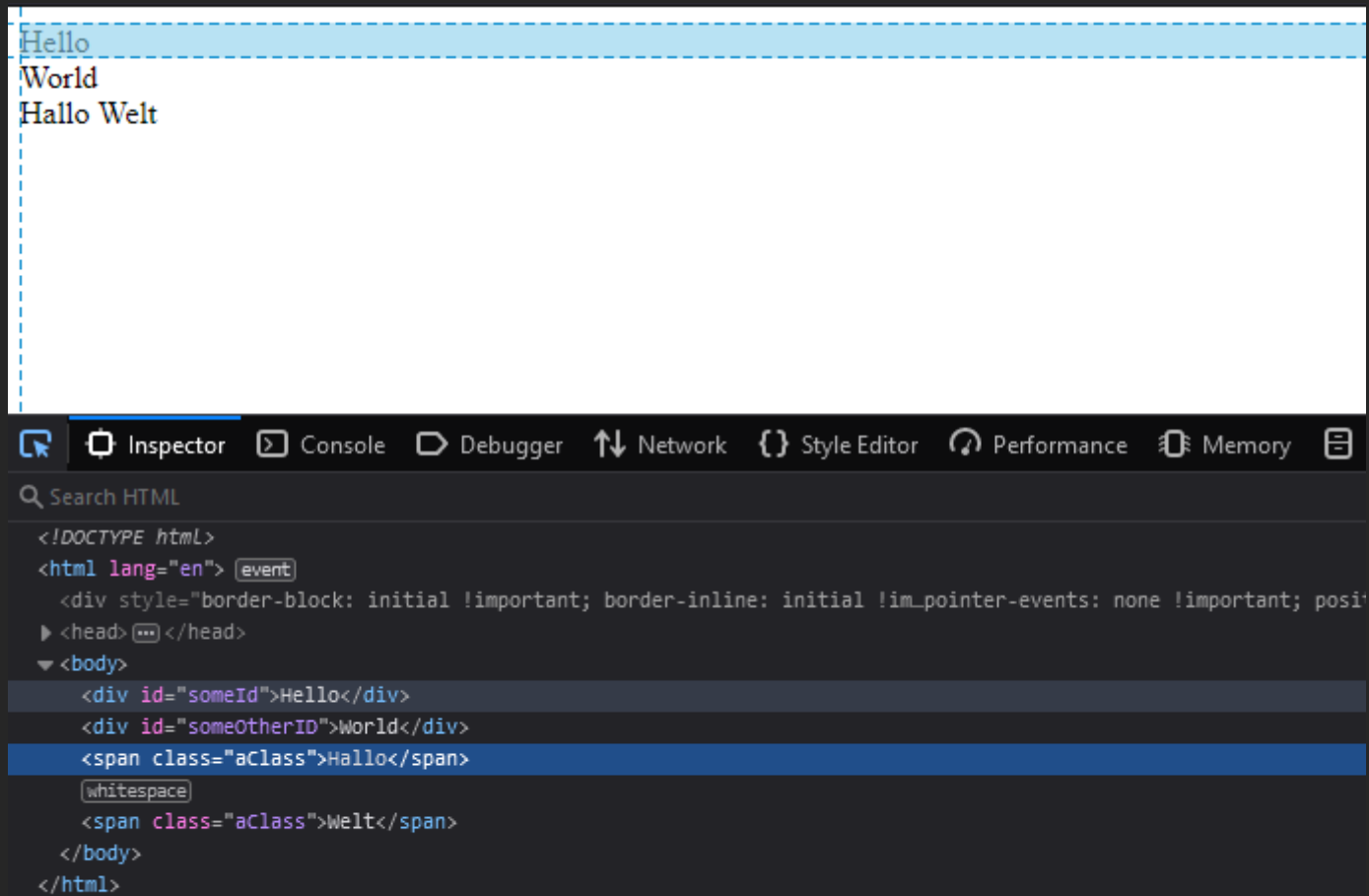
DIV & SPAN

- The `<div>` element is the most common html element
- Similar to a div is the `` element
- Both have different implicit visual attributes.



```
</style>
<body>
  <div>
    Hello
  </div>
  <div>
    World
  </div>
  <span>Hallo</span>
  <span>Welt</span>
</body>
</html>
```


DIV & SPAN



Inspector Console Debugger Network Style Editor Performance Memory

Search HTML

```
<!DOCTYPE html>
<html lang="en">
  <div style="border-block: initial !important; border-inline: initial !important; position: absolute; top: 0px; left: 0px; width: 100%; height: 100%; background-color: #e0f0ff;">
    <head>
    </head>
    <body>
      <div id="someId">Hello</div>
      <div id="someOtherID">World</div>
      <span class="aClass">Hallo</span>
      <span class="aClass">Welt</span>
    </body>
  </html>
```



span 35.55 x 17.6

Hello Workd

Elemente Konsole Quellen

```
<!DOCTYPE html>
... <html lang="en">
  <head>
  </head>
  <body>
    <div>Hello</div>
    <div>World</div>
    <span>Hello</span>
    <span>Workd</span>
  </body>
</html>
```

Find an element in the DOM

body > span:nth-child(3) (CSS Selector)

/html/body/span[1] (XPath)

Inspector Console Debugger Network Style Editor Performance Memory Storage

Search HTML

```
<!DOCTYPE html>
<html lang="en" >
  <div style="border-block: initial !important; border-inline: initial !important; position:
  <head>
  </head>
  <body>
    <div>Hello</div>
    <div>World</div>
    <span>Hallo</span>
    <span>Welt</span>
  </body>
</html>
```

body > span:nth-child(3) (CSS Selector)

/html/body/span[1] (XPath)

html > body > span

Ids & classes

<> Untitled-1.html ×

<> Untitled-1.html > html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <style>
9
10 </style>
11 <body>
12     <div id="someId">
13         Hello
14     </div>
15     <div id="someOtherID">
16         World
17     </div>
18     <span class="aClass">Hallo</span>
19     <span class="aClass">Welt</span>
20 </body>
21 </html>
```

it is a good strategy to be as general as possible
With applications of styles/interactions.

Further reading

There is not only `<div>`

<https://codingbeautydev.com/blog/rare-html-tags/>