

# Webudvikling Frontend

CSS, fonts, responsiveness & Design Principles

Webudvikling

**kea**  
KØBENHAVNS ERHVERVSAKADEMI

# Dagens Formål

- Opsamling - hands-on med moderne semantisk HTML
- Viden om performance og Web Dev Tools,
- Intro og hands-on med CSS, fonts og responsivt design
- Erfaring med værktøjer
- Og hvad med designprincipper?

# Agenda

- Opsamling
- HTML & HTML Semantics
- Performance & Dev Tool
- CSS
- Fonts
- Responsive Web Design

# Git, GitHub & GitHub Desktop

Version Control, Collaboration & how to manage  
your code projects

# Git



Git is installed and maintained on your local system (rather than in the cloud)



First developed in 2005

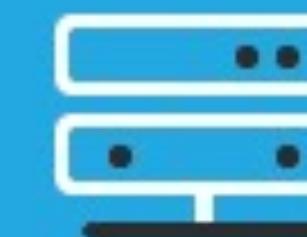


One thing that really sets Git apart is its branching model

Git is a high quality version control system

vs.

# GitHub



GitHub is designed as a Git repository hosting service

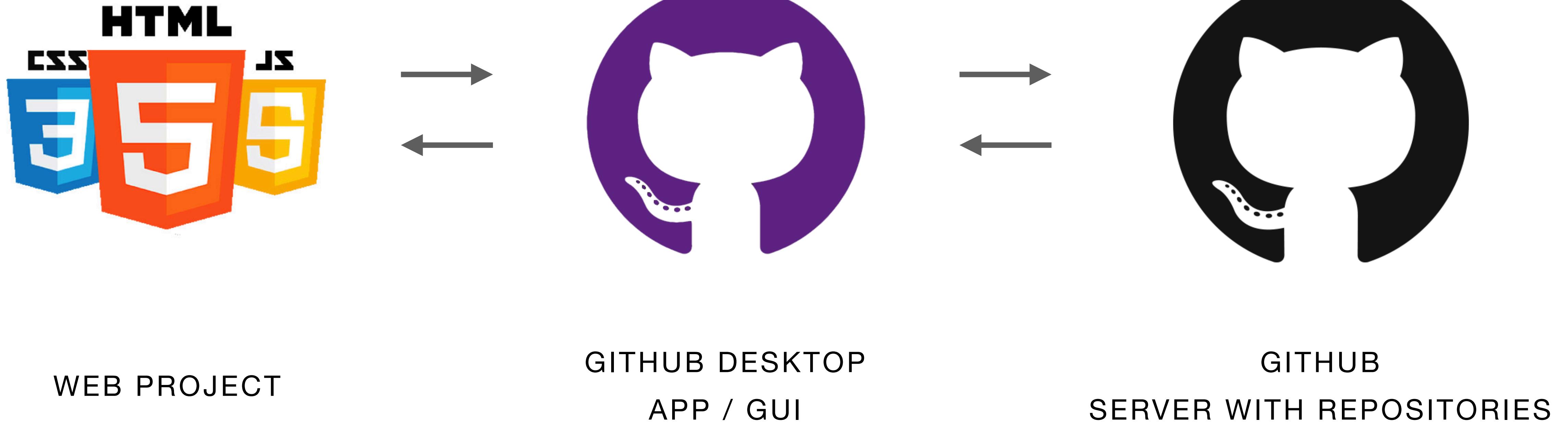


You can share your code with others, giving them the power to make revisions or edits



GitHub is a cloud-based hosting service

# Git



# Frontend stack

**HTML**



**JS**



**CSS**



STRUCTURE  
CONTENT

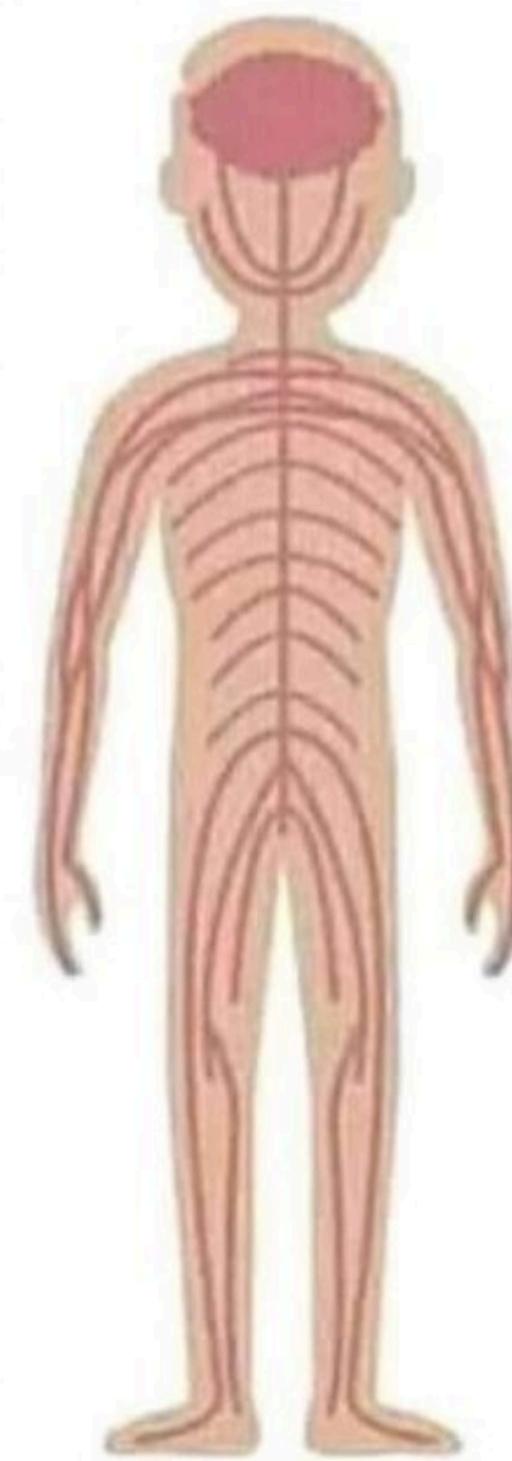
FUNCTIONALITY  
BEHAVIOR

LAYOUT/STYLING  
PRESENTATION

HTML



JS

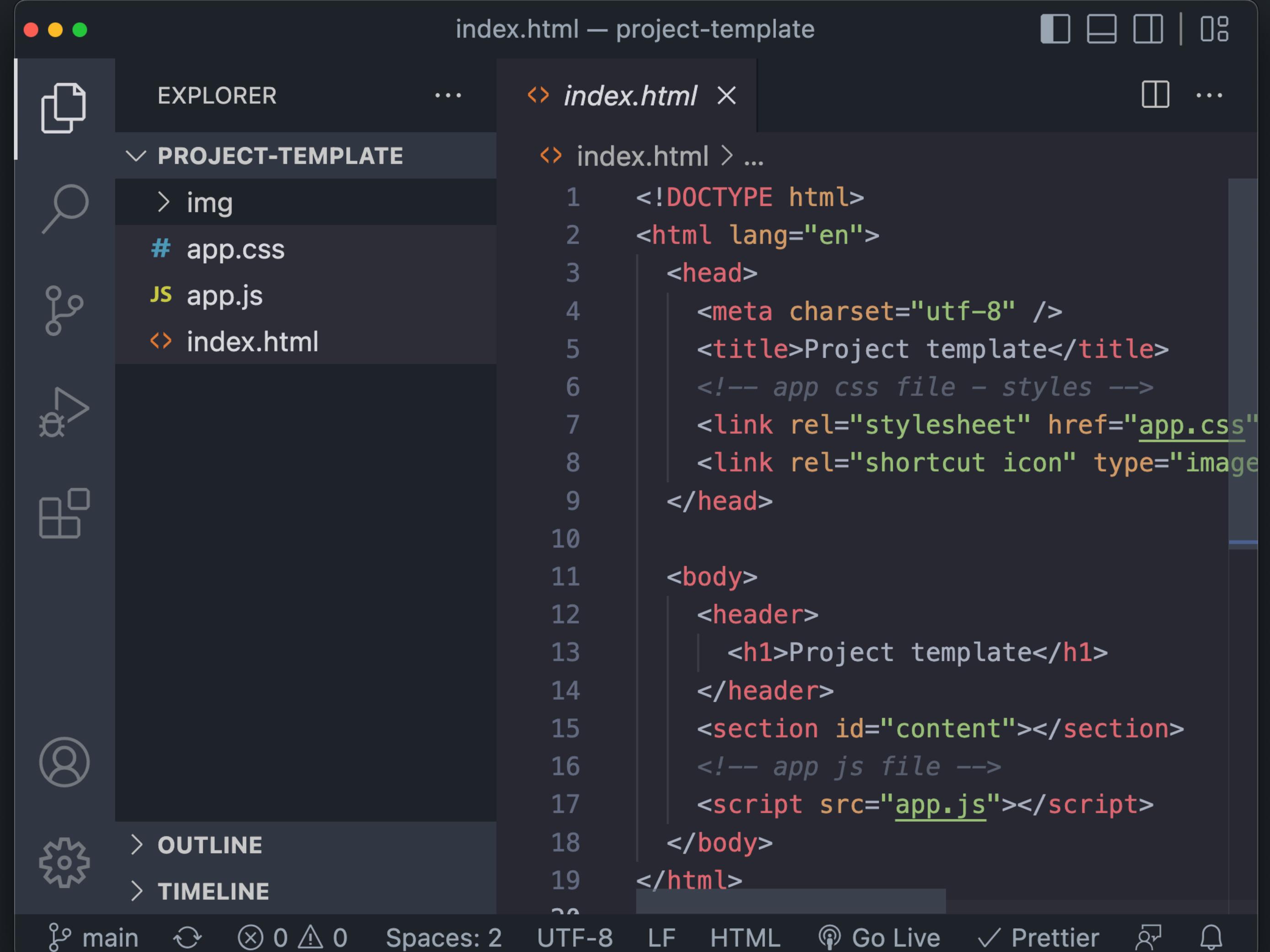


CSS



# Frontend Project Structure

- Project structure
  - HTML
  - CSS
  - JavaScript
- Keep a good structure & strive for consistency
- Separation of concerns



The screenshot shows a code editor interface with a dark theme. On the left, the Explorer sidebar displays a project structure under 'PROJECT-TEMPLATE' containing 'img', '# app.css', 'JS app.js', and 'index.html'. The 'index.html' file is selected and shown in the main editor area. The editor shows the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Project template</title>
    <!-- app css file - styles -->
    <link rel="stylesheet" href="app.css" type="text/css" />
    <link rel="shortcut icon" type="image/x-icon" href="img/icon.png" />
</head>
<body>
    <header>
        <h1>Project template</h1>
    </header>
    <section id="content"></section>
    <!-- app js file -->
    <script src="app.js"></script>
</body>
</html>
```

The status bar at the bottom indicates the file is 'index.html – project-template', has 08 tabs open, and shows icons for main, refresh, search, 0 errors, 0 warnings, spaces: 2, LF, HTML, Go Live, Prettier, and a few others.

# HTML, CSS & JavaScript

You can literally build anything with it!

# What is HTML?

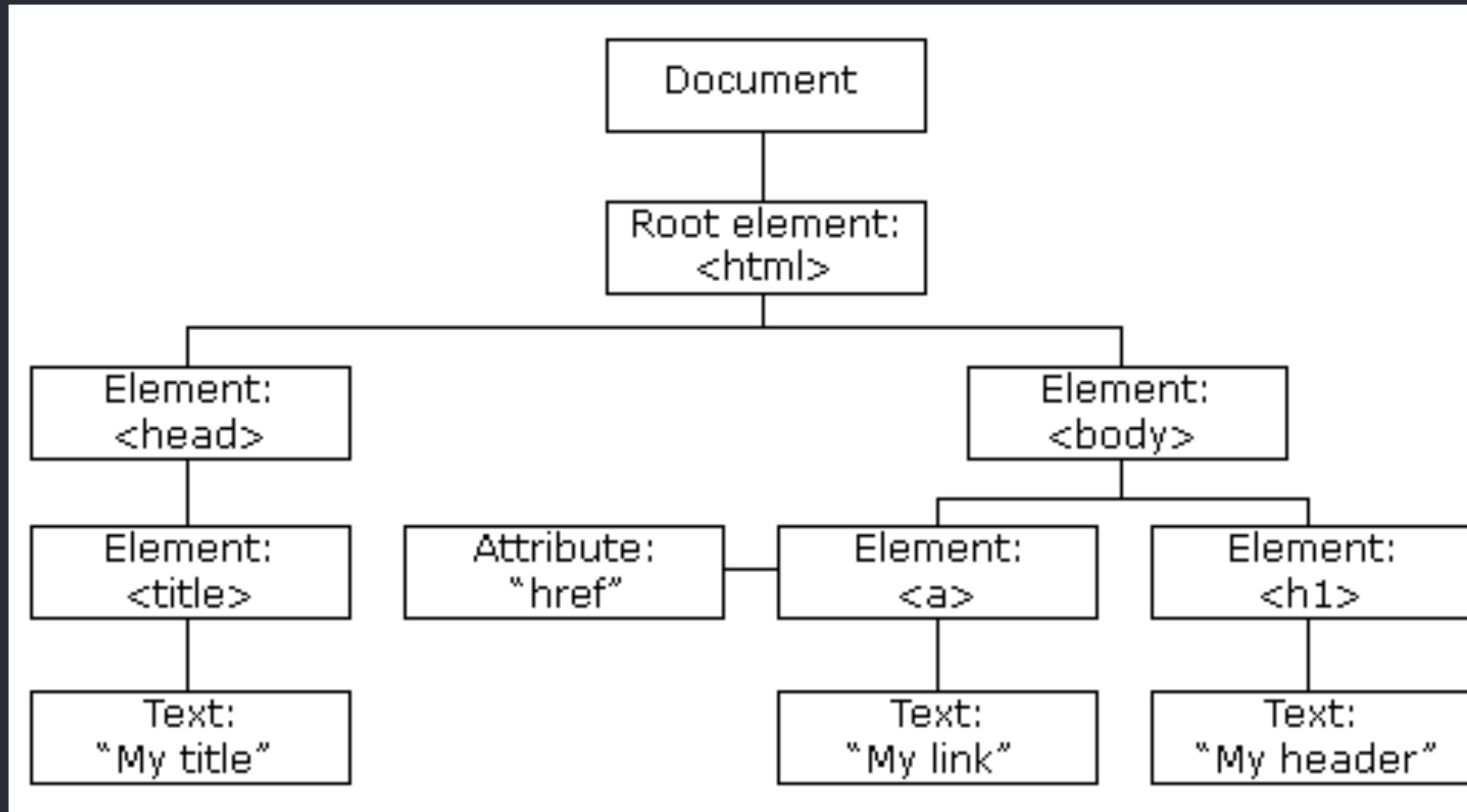
- Hyper Text Markup Language
- Standard markup language for creating Web pages
- Describes the structure of a Web page
- Consists of a series of elements (tags)
- HTML elements tell the browser how to display the content

```
index.html

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Page Title</title>
5   </head>
6   <body>
7     <h1>This is a Heading</h1>
8     <p>This is a paragraph.</p>
9   </body>
10 </html>
11
```

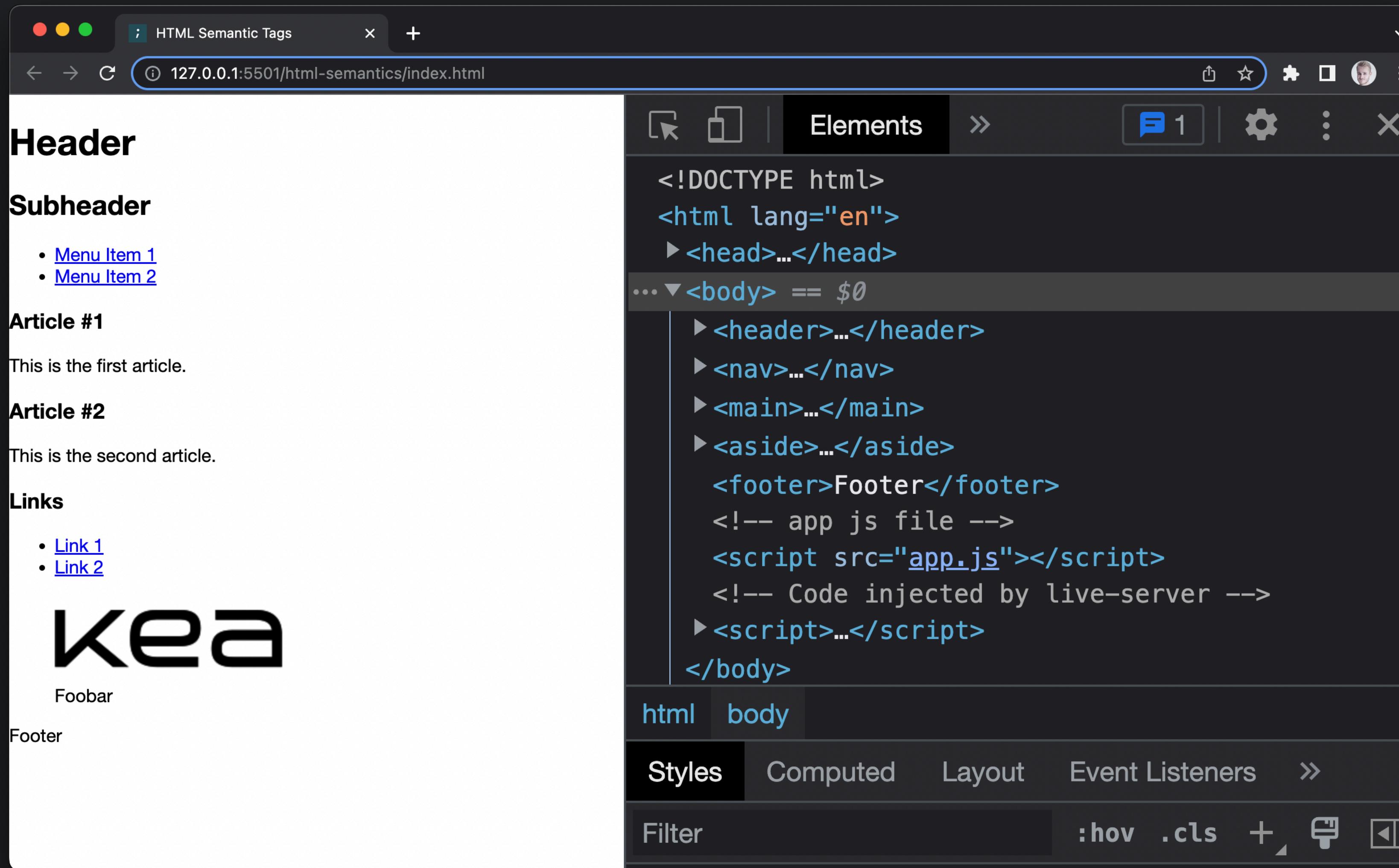
# The HTML DOM (Document Object Model)

A model as a tree of Objects



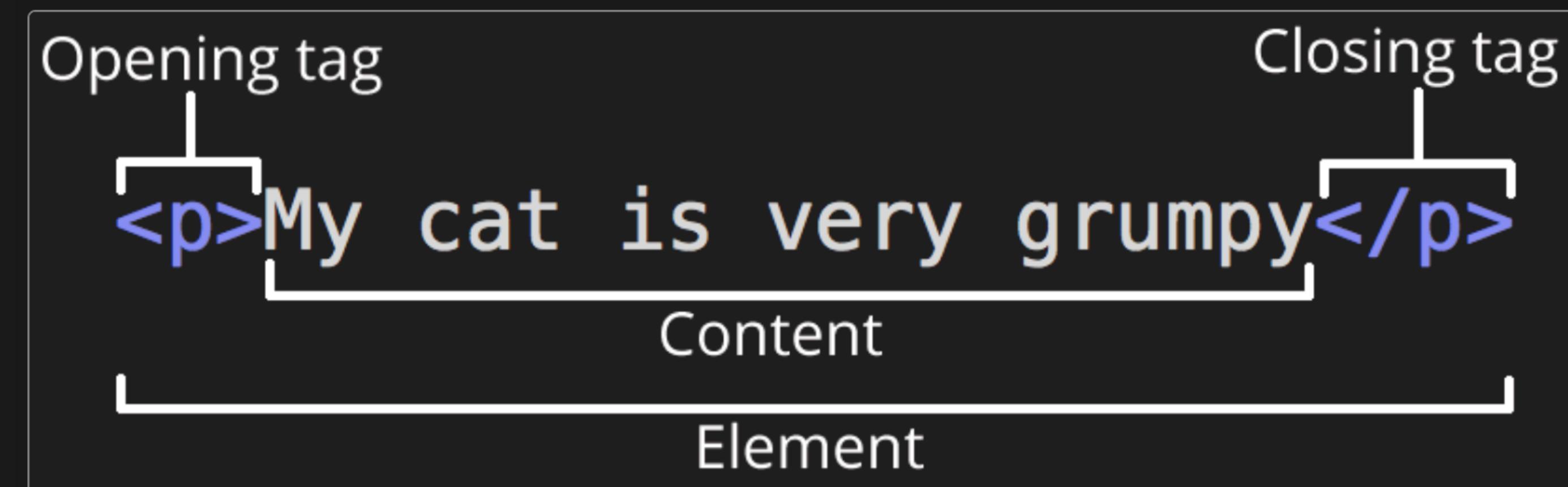
- Object Model for HTML:
  - HTML elements as objects
  - Properties for all HTML elements
  - Methods for all HTML elements
  - Events for all HTML elements

# Dev Tool: Elements



# Anatomy of an HTML element

Element: opening and closing tag + content



# Void elements

Element: single tag, usually to insert or embed something in a HTML

```

```



[https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction\\_to\\_HTML/Getting\\_started#void\\_elements](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started#void_elements)

# Attributes

- Provides additional information and all element can have them.
- Are specified in the start tag
- name/value pairs: `name="value"`

[Attributes - mdn](#)

[HTML Attributes - W3Schools](#)

The screenshot shows a dark-themed web browser window. The title bar reads "Getting started with HTML - Le X". The address bar shows the URL "developer.mozilla.org/en-US/docs/Learn/HTML/Introduction\_to\_HTML/HTML结构性标记语言". The main content area has a header "Attributes" and a sub-section "Elements can also have attributes. Attributes look like this:" followed by a code example. A callout box highlights the word "Attribute" above the code. The code example is `<p class="editor-note">My cat is very grumpy</p>`. Below the code, a text block explains that attributes contain extra information about the element and provides an example of the `class` attribute. At the bottom, there's a section titled "An attribute should have:" with a bulleted list of rules.

## Attributes

Elements can also have attributes. Attributes look like this:

Attribute

```
<p class="editor-note">My cat is very grumpy</p>
```

Attributes contain extra information about the element that won't appear in the content. In this example, the `class` attribute is an identifying name used to target the element with style information.

An attribute should have:

- A space between it and the element name. (For an element with more than one attribute, the attributes should be separated by spaces too.)
- The attribute name, followed by an equal sign.
- An attribute value, wrapped with opening and closing quote marks.

The screenshot shows a web browser window with the title "HTML Attributes". The URL in the address bar is "w3schools.com/html/html\_attributes.asp". The browser interface includes standard controls like back, forward, and search, along with a user profile icon.

The main content area has a dark background with white text. It features a navigation bar at the top with tabs for "HTML", "CSS", "JAVASCRIPT", "SQL", "PYTHON", "JAVA", and "PHP".

## The href Attribute

The `<a>` tag defines a hyperlink. The `href` attribute specifies the URL of the page the link goes to:

**Example**

```
<a href="https://www.w3schools.com">Visit W3Schools</a>
```

**Try it Yourself »**

You will learn more about links in our [HTML Links chapter](#).

## The src Attribute

The `<img>` tag is used to embed an image in an HTML page. The `src` attribute specifies the path to the image to be displayed:

**Example**

```

```

**Try it Yourself »**

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Page Title</title>
5      </head>
6      <body>
7          <h1>This is a Heading</h1>
8          <p>This is a paragraph.</p>
9      </body>
10 </html>
11
```

# This is a Heading

This is a paragraph.

# Anatomy of an HTML document

```
<> index.html ×  
1  <!DOCTYPE html>  
2  <html>  
3    <head>  
4      <title>Page Title</title>  
5    </head>  
6    <body>  
7      <h1>This is a Heading</h1>  
8      <p>This is a paragraph.</p>  
9    </body>  
10   </html>  
11
```

The `<!DOCTYPE html>` declaration defines that this document is an HTML5 document

The `<html>` element is the root element of an HTML page

The `<head>` element contains meta information about the HTML page

The `<title>` element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)

The `<body>` element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.

The `<h1>` element defines a large heading

The `<p>` element defines a paragraph

```
index.html ×  
1  <!DOCTYPE html>  
2  <html>  
3  <head>  
4    <title>Page Title</title>  
5  </head>  
6  <body>  
7    <h1>This is a Heading</h1>  
8    <p>This is a paragraph.</p>  
9  </body>  
10 </html>  
11
```

```
<html>  
  
<head>  
  
  <title>Page title</title>  
  
</head>  
  
<body>  
  
  <h1>This is a heading</h1>  
  
  <p>This is a paragraph.</p>  
  
  <p>This is another paragraph.</p>  
  
</body>  
  
</html>
```

A screenshot of a web browser window displaying the MDN Web Docs page for "HTML comments". The browser has a dark mode interface. The title bar shows the page is titled "Getting started with HTML - Le...". The address bar shows the URL "developer.mozilla.org/en-US/docs/Learn/HTML/Introduction\_to\_HTML/Getting\_started#anatomy\_of\_an\_html\_document". The page content starts with a section header "HTML comments" followed by a paragraph explaining the purpose of HTML comments. Below the paragraph, there is an example code block showing how to write an HTML comment.

## HTML comments

HTML has a mechanism to write comments in the code. Browsers ignore comments, effectively making comments invisible to the user. The purpose of comments is to allow you to include notes in the code to explain your logic or coding. This is very useful if you return to a code base after being away for long enough that you don't completely remember it. Likewise, comments are invaluable as different people are making changes and updates.

To write an HTML comment, wrap it in the special markers `<!--` and `-->`. For example:

```
<p>I'm not inside a comment</p>

<!-- <p>I am!</p> -->
```

# HTML Quiz & Exercises

HTML Quiz

HTML Exercises

# HTML Semantic Elements

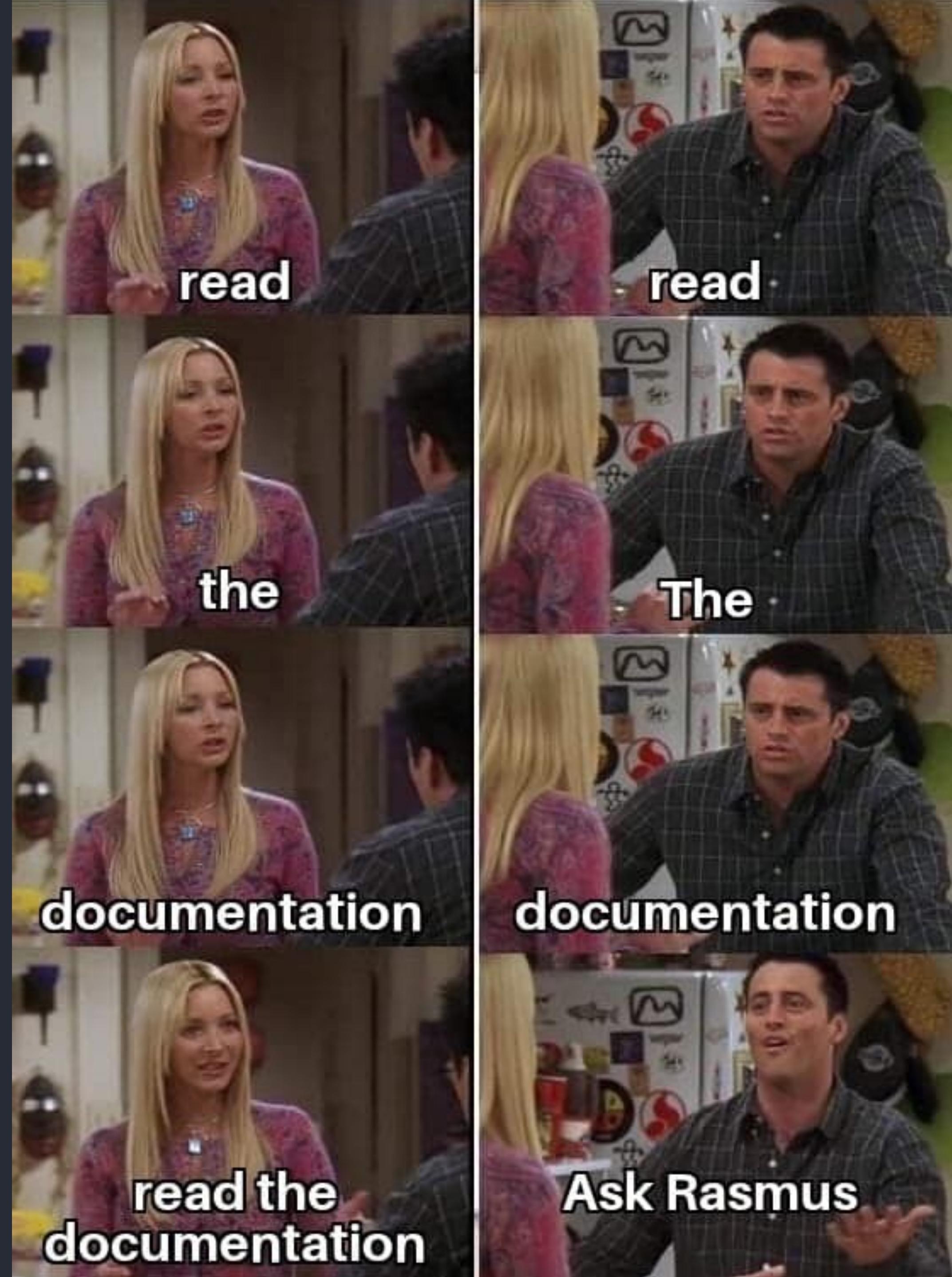
Clearly describes the meaning to both the browser and the developer.

Semantic elements = elements with a meaning

The screenshot shows a web browser window with the title "HTML Semantic Elements" and the URL "w3schools.com/html/html5\_semantic\_elements.asp". The page features a navigation bar with links for HTML, CSS, JAVASCRIPT, SQL, PYTHON, and JAVA. The main content is titled "Semantic Elements in HTML" and includes a note: "Below is a list of some of the semantic elements in HTML." A table lists 15 semantic elements with their descriptions:

Tag	Description
<code>&lt;article&gt;</code>	Defines independent, self-contained content
<code>&lt;aside&gt;</code>	Defines content aside from the page content
<code>&lt;details&gt;</code>	Defines additional details that the user can view or hide
<code>&lt;figcaption&gt;</code>	Defines a caption for a <code>&lt;figure&gt;</code> element
<code>&lt;figure&gt;</code>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<code>&lt;footer&gt;</code>	Defines a footer for a document or section
<code>&lt;header&gt;</code>	Specifies a header for a document or section
<code>&lt;main&gt;</code>	Specifies the main content of a document
<code>&lt;mark&gt;</code>	Defines marked/highlighted text
<code>&lt;nav&gt;</code>	Defines navigation links
<code>&lt;section&gt;</code>	Defines a section in a document
<code>&lt;summary&gt;</code>	Defines a visible heading for a <code>&lt;details&gt;</code> element
<code>&lt;time&gt;</code>	Defines a date/time

At the bottom, a yellow bar contains the text: "For a complete list of all available HTML tags, visit our [HTML Tag Reference](#)".



# HTML Semantic Elements

Semantic elements that can be used to define different parts of a website

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



# Why use semantic elements?

```
<body>
  <div id="header">
    <h1>PROJECT TEMPLATE</h1>
  </div>
  <div class="sections">
    <div class="article">
      <div class="figure">
        <img>
        <div class="figcaption"></div>
      </div>
    </div>
  </div>
  <div id="footer"></div>
  <!-- main js file -->
  <script src="js/main.js"></script>
</body>
```

```
<body>
  <header>
    <h1>PROJECT TEMPLATE</h1>
  </header>
  <section>
    <article>
      <figure>
        <img>
        <figcaption></figcaption>
      </figure>
    </article>
  </section>
  <footer></footer>
  <!-- main js file -->
  <script src="js/main.js"></script>
</body>
```

## NON SEMANTIC

```
<div class='header'>  
  <div class='section'>  
    <div class='article'>  
      <div class='article'>  
    </div>  
  </div>  
<div class='aside'>  
</div>  
<div class='footer'>
```

## SEMANTIC

```
<header>  
<section>  
<article>  
<article>  
<aside>  
<footer>
```



# <div> tags?

Many websites contain HTML code like:

```
<div id="nav">  
<div class="header">  
<div id="footer">
```

to indicate navigation, header, and footer.

Can we still use div tags?

Apple x + apple.com

Store Mac iPad iPhone Watch AirPods TV & Home Only on Apple Accessories Support Q □

# iPhone 14 Pro

Pro. Beyond.

[Learn more >](#) [Buy >](#)



# iPhone 14

Big and bigger.

iPhone 14 Plus available starting 10.7

[Learn more >](#) [Shop >](#)

Elements Console » F 1 ⚙️ ⋮ ×

```
><nav id="ac-globalnav" class="js no-touch no-windows no-firefox role="navigation" aria-label="Global" data-hires="false" data-analytics-region="global nav" lang="en-US" dir="ltr" data-www-domain="www.apple.com" data-store-locale="us" data-store-root-path="/us" data-store-api="/[storefront]/shop/bag/status" data-search-locale="en_US" data-search-suggestions-api="/search-services/suggestions/" data-search-defaultlinks-api="/search-services/suggestions/defaultlinks/">...</nav>
<div class="ac-gn-blur"></div>
<div id="ac-gn-curtain" class="ac-gn-curtain"></div>
<div id="ac-gn-placeholder" class="ac-gn-placeholder">
</div>
<script type="text/javascript" src="/ac/globalnav/7/en_US/scripts/ac-globalnav.built.js"></script>
...
  ><div id="ac-gn-viewport-emitter" data-viewport-emitter-dispatch data-viewport-emitter-state="{"viewport":"medium","orientation":"portrait","retina":true}">...</div> == $0
<script src="/metrics/ac-analytics/2.15.1/scripts/ac-analytics.js" type="text/javascript" charset="utf-8"></script>
  ><main class="main" role="main">...</main>
  ><footer class="js" lang="en-US" id="ac-globalfooter" data-analytics-region="global footer" role="contentinfo" aria-...
...
  -image.no-reduced-motion.no-edge.no-ie.css-mask.inline-video.deskt ...

```

Styles Computed Layout Event Listeners DOM Breakpoints »

Filter :hov .cls + ↻

```
element.style {
}
#ac-gn-viewport-emitter { ac-globalna...built.css:1
  overflow: hidden;
  position: absolute;
  top: 0;
  left: 0;
  width: 0;
  height: 0;
  visibility: hidden;
  z-index: -1;
}
```

Google

google.com

Gmail Billeder Log ind

# Google

Google-søgning Jeg prøver lykken

Google er tilgængelig på: Føroyskt

Danmark

CO2-neutral siden 2007

Om Annoncering Erhverv Sådan fungerer Google Søgning Privatliv Vilkår Indstillinger

Elements Console ↗ 1 ↘ 1 ⚙ ⋮ X

```
<!DOCTYPE html>
<html itemscope itemtype="http://schema.org/WebPage" lang="da">
  <head>...</head>
  <body jsmodel="hspDDf" jsaction="YUC7He:.CLIENT;vPBs3b:.CLIENT;IVKTfe:.CLIENT;KsNBn:.CLIENT;sbTXNb:.CLIENT;xjhTIf:.CLIENT;02vyse:.CLIENT;Ez7VMc:.CLIENT;qqf0n:.CLIENT;me3ike:.CLIENT;IrNywb:.CLIENT;Z94jBf:.CLIENT;A8708b:.CLIENT;YcfJ:.CLIENT;A6SDQe:.CLIENT;LjVEJd:.CLIENT;VM8bg:.CLIENT;hWT9Jb:.CLIENT;wCulWe:.CLIENT;NTJodf:.CLIENT;szjOR:.CLIENT;PY1zjf:.CLIENT;wnJTPd:.CLIENT;JL9QDc:.CLIENT;kWlxhc:.CLIENT;qGMTIf:.CLIENT">
    <style data-iml="1634195875072">...</style>
    ... <div class="L3eUgb" data-hveid="1"> flex == $0
      <div class="o3j99 n1xJcf Ne6nSd">...</div> flex
      <div class="o3j99 LLD4me yr19Zb LS80J">...</div> flex
      <div class="o3j99 ikrT4e om7nvf">...</div>
      <div class="o3j99 qarstb">...</div>
      <div class="o3j99 c93Gbe">...</div>
    </div>
    <div class="Fgvgjc">
      <style data-iml="1634195875111">
        .Fgvgjc{height:0;overflow:hidden}</style>
      <div class="gTMtLb fp-nh" id="lb">...</div>
      <div jscontroller="fKZehd" style="display:none" data-u="0" jsdata="C4mkuf;_;AzQv+I" jsaction="rcuQ6b:npT2md">
      <span style="display:none">...</span>
      <script nonce="4eyH3+1nxzKy0S3HAsRNUA==">...</script>
      <div>...</div>
    </div>
  html body div.L3eUgb
  Styles Computed Layout Event Listeners DOM Breakpoints >
```

Filter :hov .cls + [ ]

```
element.style {
}
.L3eUgb {
  display: flex;
  flex-direction: column;
}
```

The screenshot shows a web browser window with the title "HTML Semantic Elements" from the website w3schools.com. The page content is as follows:

# HTML Semantic Elements

Semantic elements = elements with a meaning.

## What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: `<div>` and `<span>` - Tells nothing about its content.

Examples of **semantic** elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

## Semantic Elements in HTML

Many web sites contain HTML code like: `<div id="nav">` `<div class="header">` `<div id="footer">` to indicate navigation, header, and footer.

In HTML there are some semantic elements that can be used to define different parts of a web page:

- `<article>`
- `<aside>`
- `<details>`
- `<figcaption>`

On the right side of the browser, a developer tools sidebar is open, showing the DOM structure and CSS styles applied to the page. The DOM tree starts with `<body>` and includes nodes for the main content area, sidebar, and various semantic elements like `<header>`, `<nav>`, and `<article>`. The CSS styles panel shows rules for `html` and `body` elements, including font-family: Verdana, sans-serif; and font-size: 15px;.

# <div> tags?

Use semantic elements as much as possible (!)

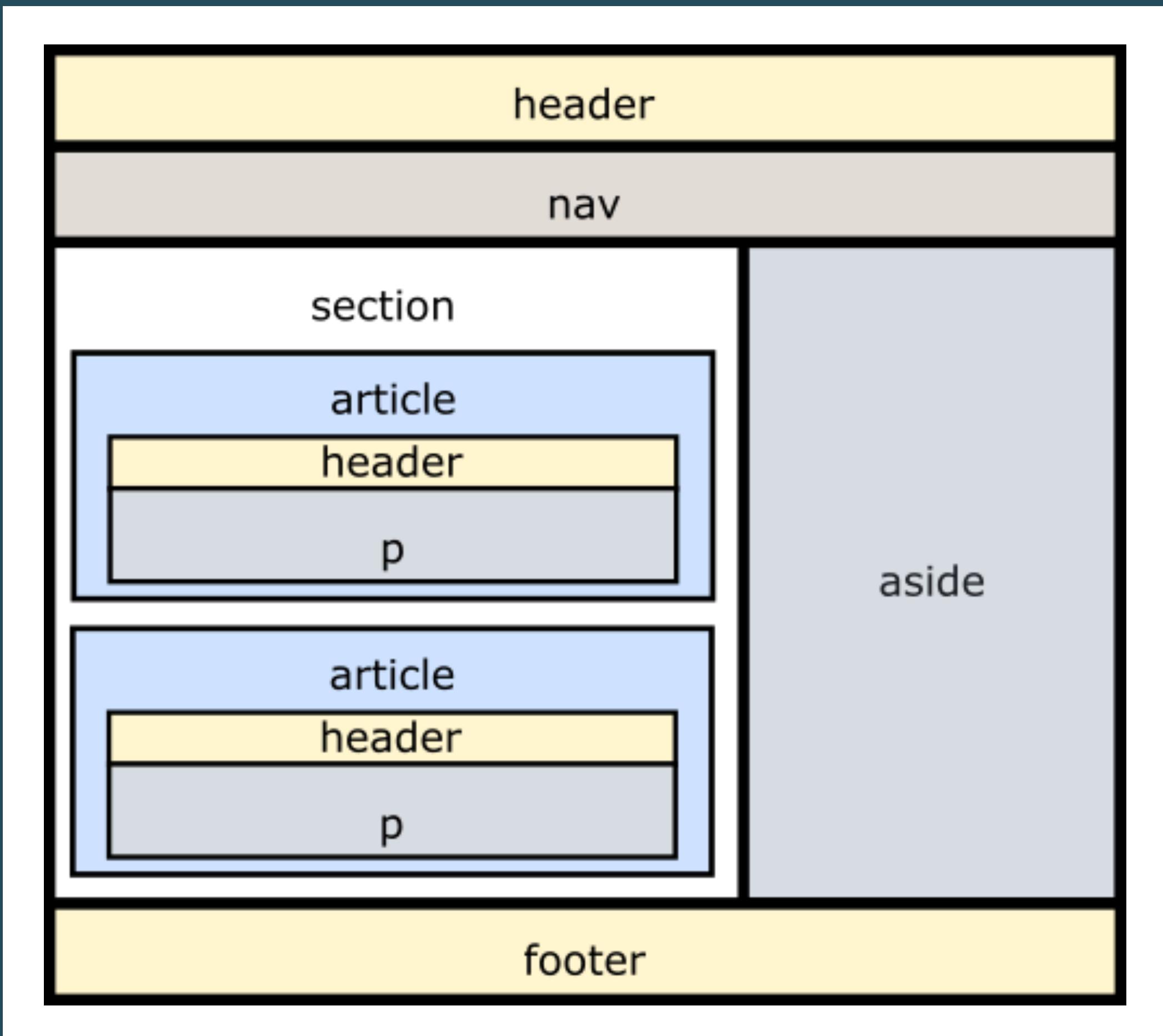
... but don't let it stop you from building

something amazing 

- RACE

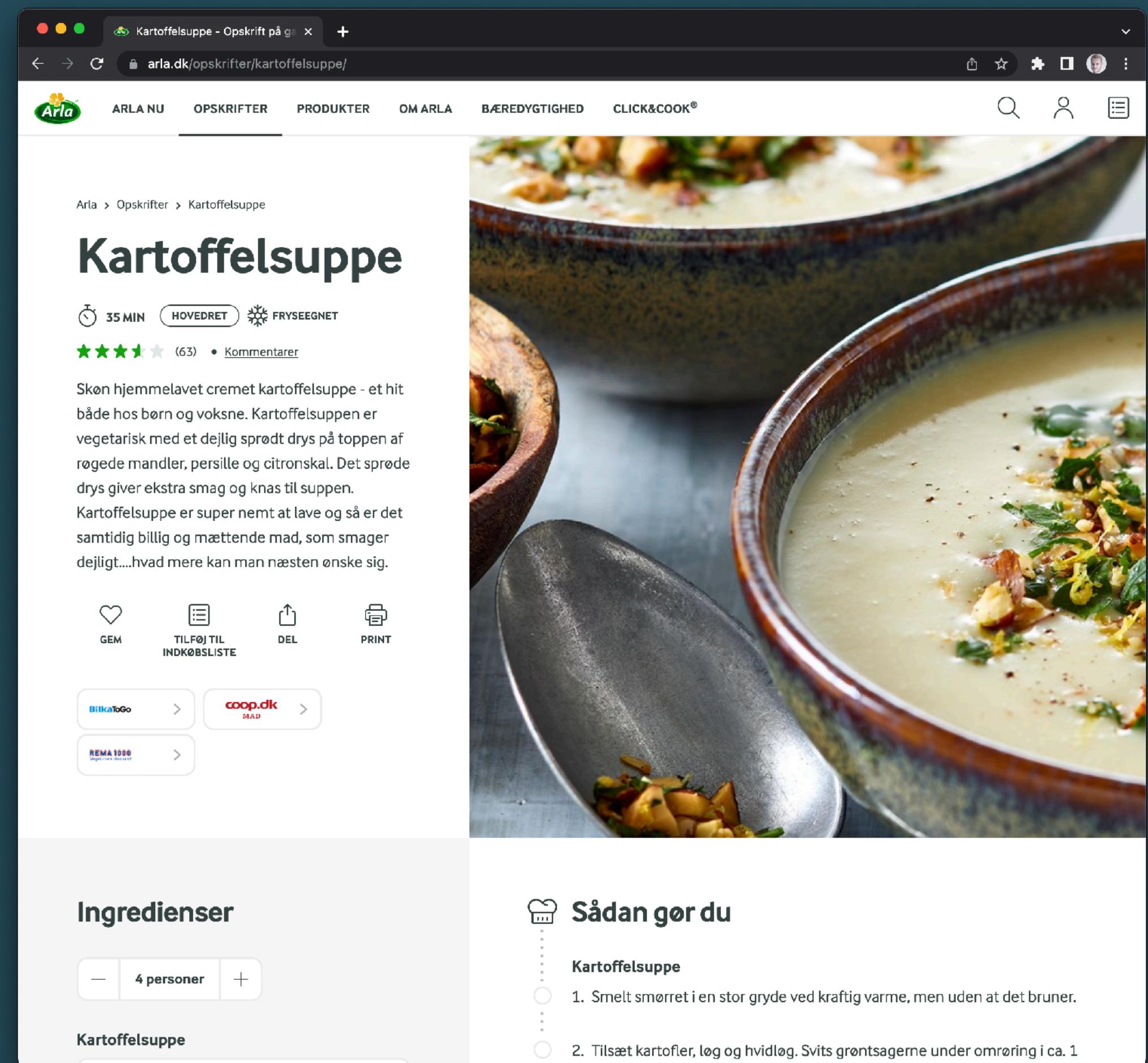
# HTML Semantics Exercise

1. Use the project-template from [GitHub](#).  
Make a copy of the project and paste the project folder into your own local dev folder.
2. Create a structure matching the image to the right. Use HTML Semantic tags.
3. Add content to the structure, like headings, links, text (paragraphs), images and contact info. Explore and use [HTML Element Reference](#) & [HTML Semantics](#).
4. Extra: Add some styling (CSS).



# HTML Recipe Exercise

- Tag udgangspunkt i dit projekt fra foregående opgave (tag eventuelt en kopi).
- Strukturér en opskriftsside med elementer som overskrift(er), beskrivelse(r), billede(r), liste af ingredienser og fremgangsmåde. Du skal bruge HTML elementer og semantiske tags.
- Fokus er struktur i HTML'en - ikke styling og layout.
- Du kan bruge elementer som h1, h2, h3, p, section, article, aside, main, ul, li og mange andre. Gå på opdagelse i "dokumentationen".
- Brug indhold fra websitet til venstre eller en anden opskrift efter eget ønske.



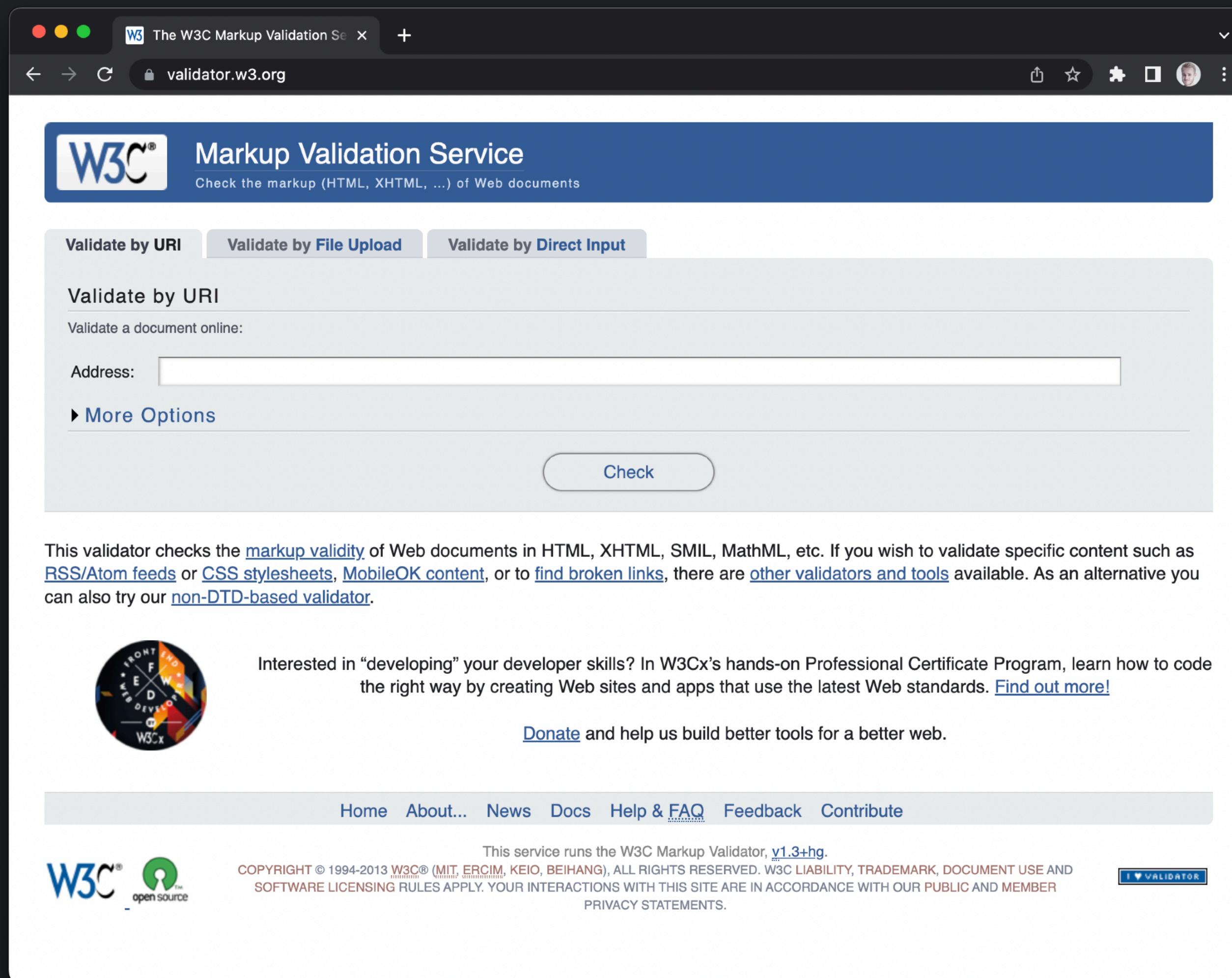
# HTML Recipe Exercise

- Undersøg Arla-opskrifter.
- Er HTML'en semantisk?  
Hvorfor, hvorfor ikke?
- Forbederingsforslag?

The screenshot displays a recipe page for "Kartoffelsuppe" (Potato Soup) from the Arla website. The page includes the following details:

- Title:** Kartoffelsuppe
- Preparation Time:** 35 MIN
- Cooking Method:** HOVEDRET (Main Course) / FRYSEEGNET (Frozen)
- Rating:** ★★★★☆ (63 reviews) • Kommentarer (Comments)
- Description:** Skøn hjemmelavet cremet kartoffelsuppe - et hit både hos børn og voksne. Kartoffelsuppen er vegetarisk med et dejlig sprødt drys på toppen af røgede mandler, persille og citronskal. Det sprøde drys giver ekstra smag og knas til suppen. Kartoffelsuppe er super nemt at lave og så er det samtidig billig og mættende mad, som smager dejligt....hvad mere kan man næsten ønske sig.
- Buttons:** GEM (Save), TILFØJ TIL INDKØBSLISTE (Add to shopping list), DEL (Delete), PRINT (Print).
- Sharing Buttons:** BlitzaToGo, coop.dk MAD, REMA 1000.
- Ingredients:** Kartoffelsuppe (4 persons).
- Instructions:** Sådan gør du (How to make it):
  1. Smelt smørret i en stor gryde ved kraftig varme, men uden at det bruner.
  2. Tilsæt kartofler, løg og hvidløg. Svits grøntsagerne under omrøring i ca. 1

# Validate your HTML



The screenshot shows a web browser window displaying the W3C Markup Validation Service. The title bar reads "The W3C Markup Validation Service". The address bar shows the URL "validator.w3.org". The main content area has a blue header bar with the W3C logo and the text "Markup Validation Service" and "Check the markup (HTML, XHTML, ...) of Web documents". Below this, there are three tabs: "Validate by URI" (which is selected), "Validate by File Upload", and "Validate by Direct Input". The "Validate by URI" section contains fields for "Address:" and a "Check" button. A "More Options" link is also present. Below this section, a descriptive text explains the validator's capabilities and links to other validation tools. At the bottom, there is a circular "W3Cx" logo with the text "FRONT END WEB DEVELOPMENT", a "DONATE" button, and a navigation menu with links to Home, About..., News, Docs, Help & FAQ, Feedback, and Contribute. The footer contains copyright information and a "VALIDATOR" button.

<https://validator.w3.org/>

# Lighthouse in Chrome

The screenshot shows a browser window displaying a cookie recipe page. The page has a header with the title "Cookies" and a sub-section titled "Amerikanske chocolate chip cookies". Below the sub-section is a paragraph of text describing the cookies. A large image of chocolate chip cookies is centered on the page.

To the right of the browser window, the Lighthouse extension interface is visible. The top bar shows the URL "http://127.0.0.1:5501/html-semantics-recipe/index.html". The main summary section displays five scores in circles: Performance (100), Accessibility (100), Best Practices (100), SEO (80), and PWA (not applicable). Below this, the "Performance" section is expanded, showing a score of 100 with a green circle. It includes a sub-section for metrics like First Contentful Paint, Speed Index, Largest Contentful Paint, Time to Interactive, Total Blocking Time, and Cumulative Layout Shift, all of which are listed as 0 ms or 0 s. At the bottom of the Lighthouse interface, there are buttons for "View Original Trace" and "View Treemap".

# Semantic HTML & SEO(indexability)

Why tell the browser what the HTML elements represents?

... to improve

# Search Engine Optimization

Tell what your content is about and  
by hierarchy, tell what's important

# Why Semantic HTML?

“Because semantic HTML uses elements for their given purpose, it’s easier for both people and machines to read and understand it.”

“Semantic HTML means using correct HTML elements for their correct purpose as much as possible. Semantic elements are elements with a meaning; if you need a button, use the `<button>` element (and not a `<div>` element).”

### Semantic

```
<button>Report an Error</button>
```

### Non-semantic

```
| <div>Report an Error</div>
```

**Examples of non-semantic elements: <div> and <span>**

- Tells nothing about its content.

**Examples of semantic elements: <form>, <table>, and <article>**

- Clearly defines its content.

1. Headings <h1> to <h6>
2. Document Structure
3. Textual Meaning (Bold, Italics, Highlight)
4. Media Type
5. Correlation Tags

5 Ways to Write Semantic HTML and Improve Webpage SEO and Accessibility

[Read more here](#)

# What's Accessibility?

“[...] the practice of making your websites usable by as many people as possible.”

- and machines, programs and devices.

What is accessibility?

“Web accessibility means that people with disabilities can use the Web.”

Why Web Accessibility Is Important and How You Can Accomplish It

- Semantic HTML
- Headings are Important!
- Alternative Text
- HTML Language Type (en, da, etc.)
- Clear written language
- Show the user this is a link or a clickable element
- ARIA - a set of roles and attributes to make the content more accessible

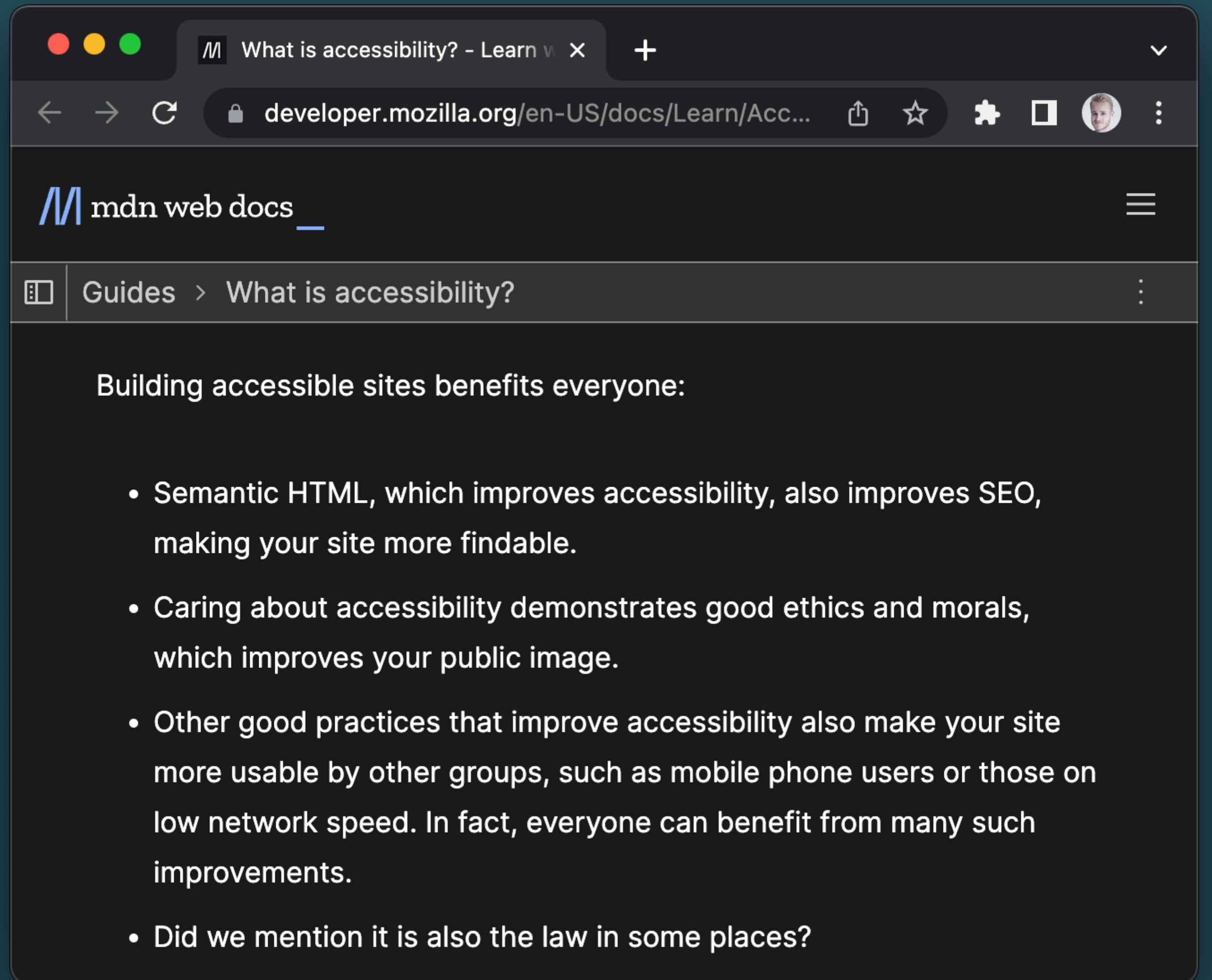
[HTML Accessibility - W3Schools](#)

[HTML: A good basis for accessibility](#)

[Accessibility and HTML](#)

[ARIA](#)

# The benefits of accessibility?

A screenshot of a web browser window displaying the MDN Web Docs page titled "What is accessibility?". The browser has a dark mode interface. The page content starts with a heading "Building accessible sites benefits everyone:" followed by a bulleted list of six items. The list discusses the benefits of accessibility, including improved SEO, good ethics, user usability, and legal requirements.

Building accessible sites benefits everyone:

- Semantic HTML, which improves accessibility, also improves SEO, making your site more findable.
- Caring about accessibility demonstrates good ethics and morals, which improves your public image.
- Other good practices that improve accessibility also make your site more usable by other groups, such as mobile phone users or those on low network speed. In fact, everyone can benefit from many such improvements.
- Did we mention it is also the law in some places?

What is accessibility?

# Lighthouse in Chrome

The image shows a screenshot of a web browser displaying the KEA - Københavns Erhvervsakademি website. The main heading on the page is "VIL DU STARTE PÅ EN UDDANNELSE TIL JANUAR?". Below it is a large button labeled "→ SØG IND NU". On the left side of the page, there are three buttons: "FÅ HJÆLP TIL AT VÆLGE UDDANNELSE", "SÅDAN ANSØGER DU", and "FIND SVAR I VORES FAQ". On the right side, there is a section titled "FÅ 10.000 TIL DIN EFTERUDDANNELSE" with a button "→ LÆS HVORDAN PÅ KEA.DK/EFTERUDDANNELSER" and a small image of a woman sitting cross-legged.

To the right of the browser window is the Lighthouse performance audit interface. The top navigation bar includes tabs for "Elements", "Console", "Sources", "Network", "Lighthouse", and "PWA". The audit score is shown as 79. Below the score, five circular icons represent different categories: Performance (79), Accessibility (89), Best Practices (67), SEO (82), and PWA (PWA). The "Performance" section is expanded, showing detailed metrics:

Metric	Value
First Contentful Paint	0.7 s
Time to Interactive	1.8 s
Total Blocking Time	0 ms
Cumulative Layout Shift	0.055

At the bottom of the Lighthouse interface, there are buttons for "View Original Trace" and "View Treemap".

# Performance is about...

- Load time of your website
- Is your site usable while loading resources? (Lazy loading)
- Smoothness and interactivity
- Does your site seem fast to the user? (Perceived performance)
- Measuring performance

Reduce and minimise size (ex images),  
loading and response time

# Lighthouse in Chrome

The image shows a screenshot of a web browser displaying the KEA - Københavns Erhvervsakademি website. The main heading on the page is "VIL DU STARTE PÅ EN UDDANNELSE TIL JANUAR?". Below it is a large button labeled "→ SØG IND NU". The page also features several other buttons: "FÅ HJÆLP TIL AT VÆLGE UDDANNELSE", "SÅDAN ANSØGER DU", and "FIND SVAR I VORES FAQ". To the right, there is a section titled "FÅ 10.000 TIL DIN EFTERUDDANNELSE" with a button "→ LÆS HVORDAN PÅ KEA.DK/EFTERUDDANNELSER" and a small image of a woman sitting cross-legged.

On the right side of the image, the Lighthouse performance audit results for the URL <https://kea.dk/> are displayed. The overall score is 79. The audit results are categorized into five sections: Performance (79), Accessibility (89), Best Practices (67), SEO (82), and PWA (PWA). The Performance section provides a detailed breakdown of metrics:

Metric	Value
First Contentful Paint	0.7 s
Speed Index	1.8 s
Largest Contentful Paint	3.0 s
Time to Interactive	1.8 s
Total Blocking Time	0 ms
Cumulative Layout Shift	0.055

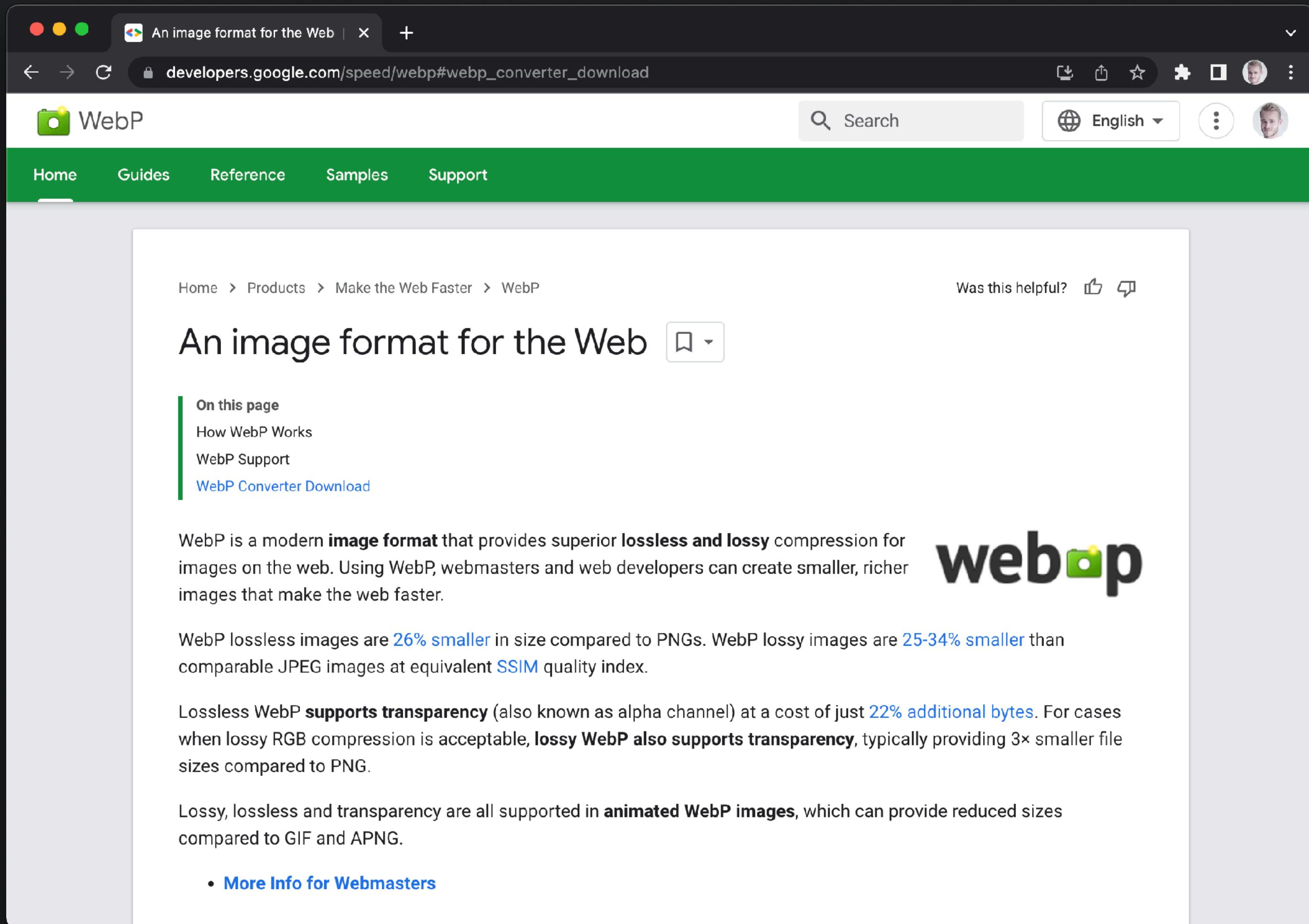
Below the metrics, there is a "View Original Trace" button and a "View Treemap" button. At the bottom, there is a horizontal timeline showing the progression of the page load.

# Not bragging



The image shows a split-screen view of a web browser. On the left is the homepage of [cederdorff.com](https://cederdorff.com), featuring a large portrait of a man wearing a black fedora and a dark turtleneck. Below the photo, the text "I'M A |" is followed by "WEB APP DEVELOPER | SENIOR LECTURER" and social media icons for Instagram, LinkedIn, and Facebook. On the right is the Lighthouse performance audit results page for the same URL. The audit summary shows scores of 95 for Performance, 94 for Accessibility, 100 for Best Practices, 100 for SEO, and a PWA badge. The "Performance" section details a score of 95 with metrics: First Contentful Paint (0.6s), Speed Index (0.8s), Largest Contentful Paint (1.5s), Time to Interactive (0.6s), Total Blocking Time (0ms), and Cumulative Layout Shift (0.003). A small thumbnail of the homepage is shown next to the performance details.

# Images, sizes & formats



The screenshot shows a web browser window displaying the "An image format for the Web" page from the Google Developers website. The URL in the address bar is [https://developers.google.com/speed/webp#webp\\_converter\\_download](https://developers.google.com/speed/webp#webp_converter_download). The page has a green header with the "WebP" logo and navigation links for Home, Guides, Reference, Samples, and Support. The main content area includes a breadcrumb trail (Home > Products > Make the Web Faster > WebP), a "Was this helpful?" feedback section, and a large heading "An image format for the Web". Below the heading, there's a sidebar titled "On this page" with links to "How WebP Works", "WebP Support", and "WebP Converter Download". The main text explains that WebP is a modern image format providing superior lossless and lossy compression. It highlights that lossless images are 26% smaller than PNGs and lossy images are 25-34% smaller than comparable JPEGs. It also notes that lossless WebP supports transparency at a cost of 22% additional bytes, while lossy WebP also supports transparency. The text concludes by mentioning animated WebP images. At the bottom, there's a link to "More Info for Webmasters". To the right of the text, the "webp" logo is displayed.

<https://developers.google.com/speed/webp>

# .webp

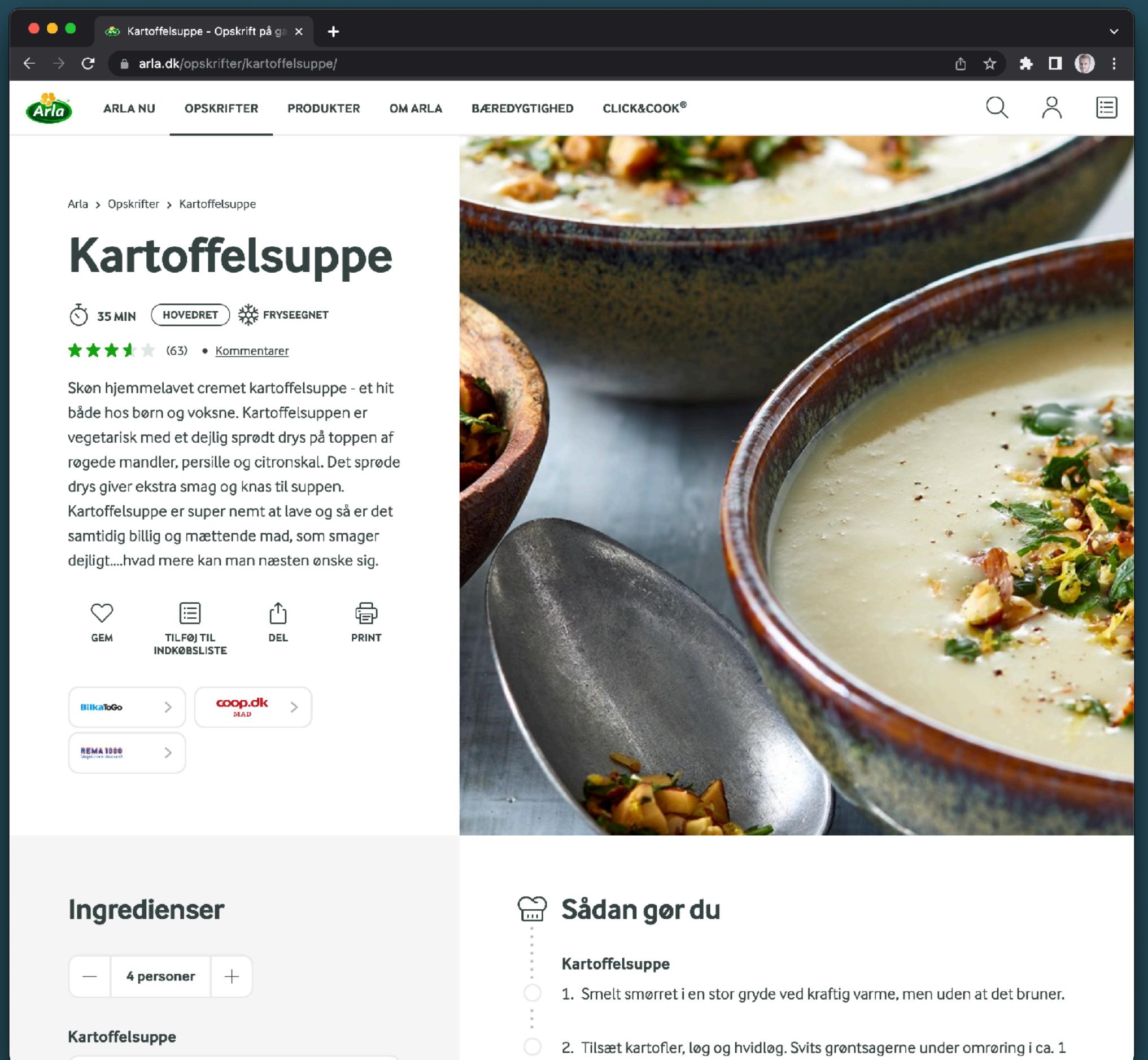
The screenshot shows a web browser window with the title "Rasmus Cederdorff | Web App" and the URL "cederdorff.com". The main content area displays a section titled "CLIENTS" with a sub-section for "HOUSE OF VINCENT" featuring a "WEBSITE + WEBSHOP". Below this, there is a paragraph about the website's implementation and a "GO TO WEBSITE AND WEBSHOP >" button. To the right of the main content is the "Network" tab of the developer tools, which shows a timeline and a table of network requests. The table lists 11 requests, all of which are successful (Status 200) and have a type of "webp". The initiator for all requests is "react-dom...". The "Waterfall" column shows a vertical stack of colored bars corresponding to each request.

Name	Status	Type	Initiator	Size	Time	Waterfall
banner_web.webp	200	webp	react-dom...	(mem...)	0 ms	
rasmus_new.webp	200	webp	react-dom...	(mem...)	0 ms	
logo_inverted.webp	200	webp	react-dom...	(mem...)	0 ms	
header_mobile.6eb9647...	200	webp	main.b9823...	(mem...)	0 ms	
expertise_bg.9998200f8...	200	svg+...	main.b9823...	(mem...)	0 ms	
houseofvincent.webp	200	webp	react-dom...	(mem...)	0 ms	
boutime_web.webp	200	webp	react-dom...	(mem...)	0 ms	
sidewalk.webp	200	webp	react-dom...	(mem...)	0 ms	
thebigfridge_web.webp	200	webp	react-dom...	(mem...)	0 ms	
cphcloud_web.webp	200	webp	react-dom...	(mem...)	0 ms	
karolineshus_web.webp	200	webp	react-dom...	(mem...)	0 ms	

11 / 23 requests | 0 B / 529 kB transferred | 720 kB / 1.8 MB resources | Finish: 86 ms | DOMContentLoaded

# Webp Exercise

1. Find og download et par billeder du kan bruge på din opskriftsside.
2. Sørg for at billeder ikke er større end nødvendigt (opløsning, height og width).
3. Test med Lighthouse. Tag et screenshot af resultatet.
4. Konverter dine billeder til .wepb (brug fx en online converter).
5. Brug webp-filerne i stedet for dine tidligere billedfiler.
6. Test med Lighthouse igen og sammenlign resultatet med tidligere



# WHAT IS CSS?

**CSS** stands for **Cascading Style Sheets**

CSS describes how HTML elements are to be displayed on screen, paper, or in other media

CSS **saves a lot of work**. It can control the layout of multiple web pages all at once

External stylesheets are stored in **CSS files**

[https://www.w3schools.com/css/css\\_intro.asp](https://www.w3schools.com/css/css_intro.asp)

```
styles.css x
1 body {
2   font-family: Helvetica, Arial, sans-serif;
3   text-align: center;
4 }
5
6 h1 {
7   font-weight: lighter;
8 }
9
10 #my-paragraph {
11   color: blue;
12 }
13
14 .big {
15   font-size: 25px;
16 }
17
```

w CSS Introduction

w3schools.com/css/css\_intro.asp

HTML CSS JAVASCRIPT SQL PYTHON JAVA PHP BOOTSTRAP HOW TO W3.CSS C C++ C# REACT R

Find dit efterårslook  
SHOP →

NEW We just launched W3Schools videos

Explore now

COLOR PICKER

f i n g

Get certified by completing a CSS course today!

w3schools 3 CERTIFIED 2022

## CSS Demo - One HTML Page - Multiple Styles!

Here we will show one HTML page displayed with four different stylesheets. Click on the "Stylesheet 1", "Stylesheet 2", "Stylesheet 3", "Stylesheet 4" links below to see the different styles:

# Welcome to My Homepage

Use the menu to select different Stylesheets

**Stylesheet 1**

Stylesheet 2  
Stylesheet 3  
Stylesheet 4  
No Stylesheet

## Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the menu, or by selecting one of the following links:  
[Stylesheet1](#), [Stylesheet2](#), [Stylesheet3](#), [Stylesheet4](#).

### No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet:  
[No Stylesheet](#).

Side-Bar

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

[https://www.w3schools.com/css/css\\_intro.asp](https://www.w3schools.com/css/css_intro.asp)

# Anatomy of a CSS ruleset



- **Selector:** the HTML element(s) to be styled
- **Declaration:** single rule like `color: red` or `margin: 10px`
- **Properties:** ways you can style HTML element(s). Ex `color` or `margin`. You choose what properties you want to affect.
- **Property value:** One out of many possible appearances for a given style (property). Ex `red` or `10px`.

index.html

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Page Title</title>
5     <link rel="stylesheet" href="styles.css" />
6   </head>
7   <body>
8     <h1>This is a Heading</h1>
9     <p>This is a paragraph.</p>
10    <p id="my-paragraph">This is another paragraph.</p>
11    <p class="big">This is another paragraph.</p>
12  </body>
13 </html>
14
```

styles.css

```
1 body {
2   font-family: Helvetica, Arial, sans-serif;
3   text-align: center;
4 }
5
6 h1 {
7   font-weight: lighter;
8 }
9
10 #my-paragraph {
11   color: blue;
12 }
13
14 .big {
15   font-size: 25px;
16 }
```

Browser Tests

https://9nypo.csb.app/

This is a Heading

This is a paragraph.

This is another paragraph.

This is another paragraph.

index.html x

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Page Title</title>
5          <link rel="stylesheet" href="styles.css" />
6      </head>
7      <body>
8          <h1>This is a Heading</h1>
9          <p>This is a paragraph.</p>
10         <p id="my-paragraph">This is another paragraph.</p>
11         <p class="big">This is another paragraph.</p>
12         
13     </body>
14 </html>
```

styles.css x

```
9
10 #my-paragraph {
11     color: blue;
12 }
13
14 .big {
15     font-size: 25px;
16 }
17
18 img {
19     width: 100%;
20 }
```

...

Browser

Tests



https://9nypo.csb.app/



# CSS SELECTORS

CSS selectors are used to "find" (or select) the HTML elements you want to style.

Simple selectors (name, id, class)

Combinator selectors (a specific relationship between them)

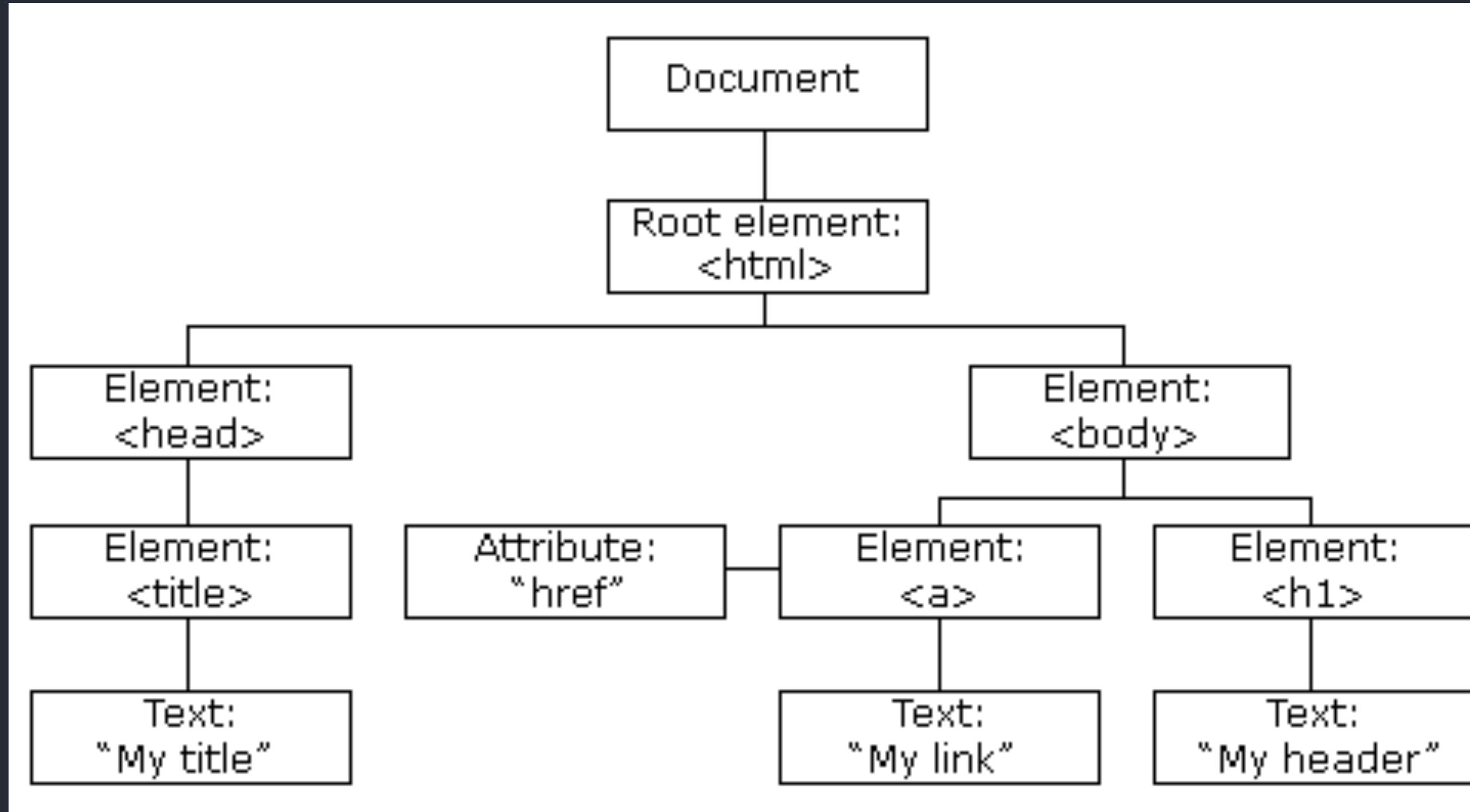
Pseudo-class selectors (a certain state)

Pseudo-elements selectors (a part of an element)

Attribute selectors (attribute or attribute value)

# The HTML DOM (Document Object Model)

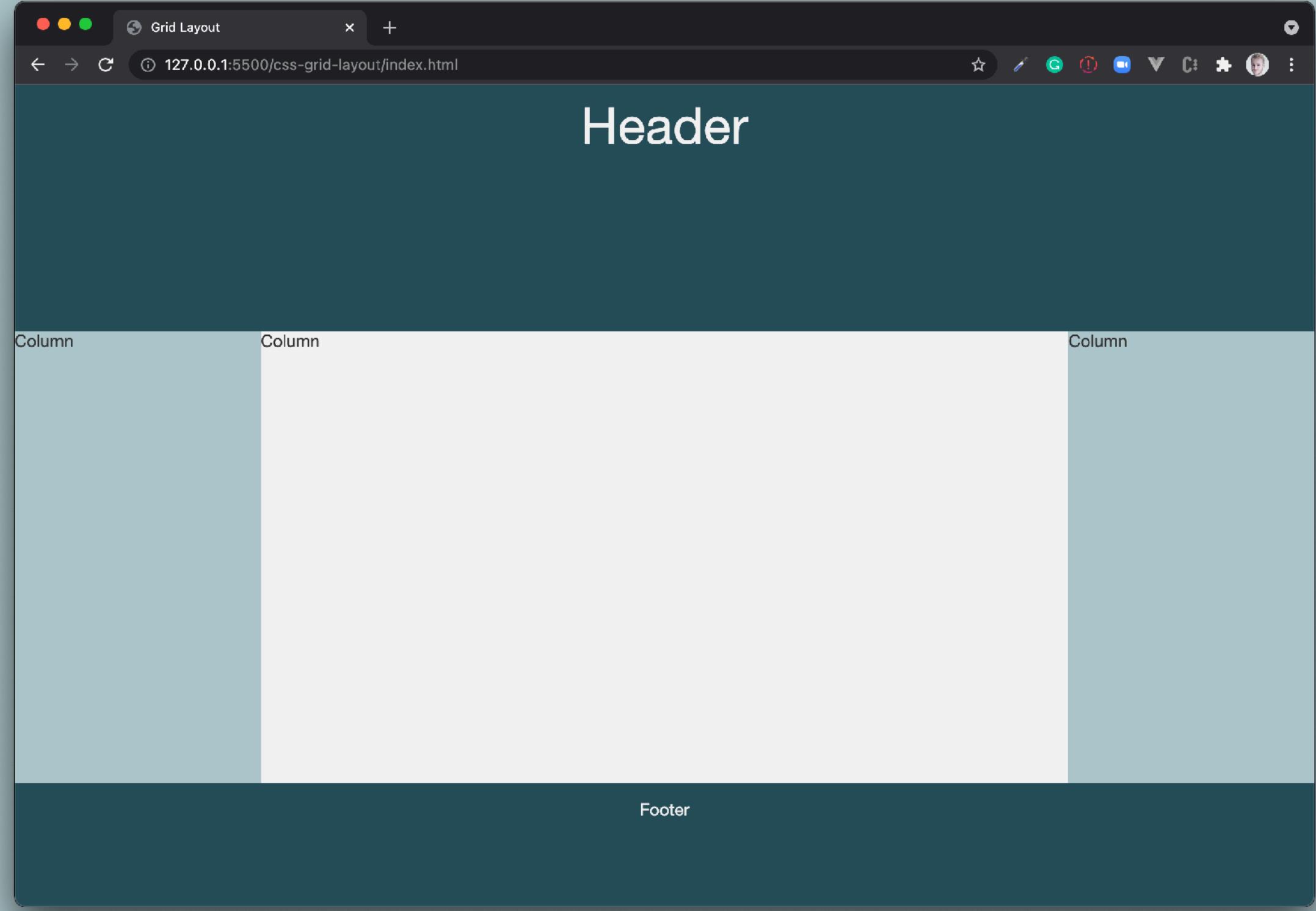
A model as a tree of Objects



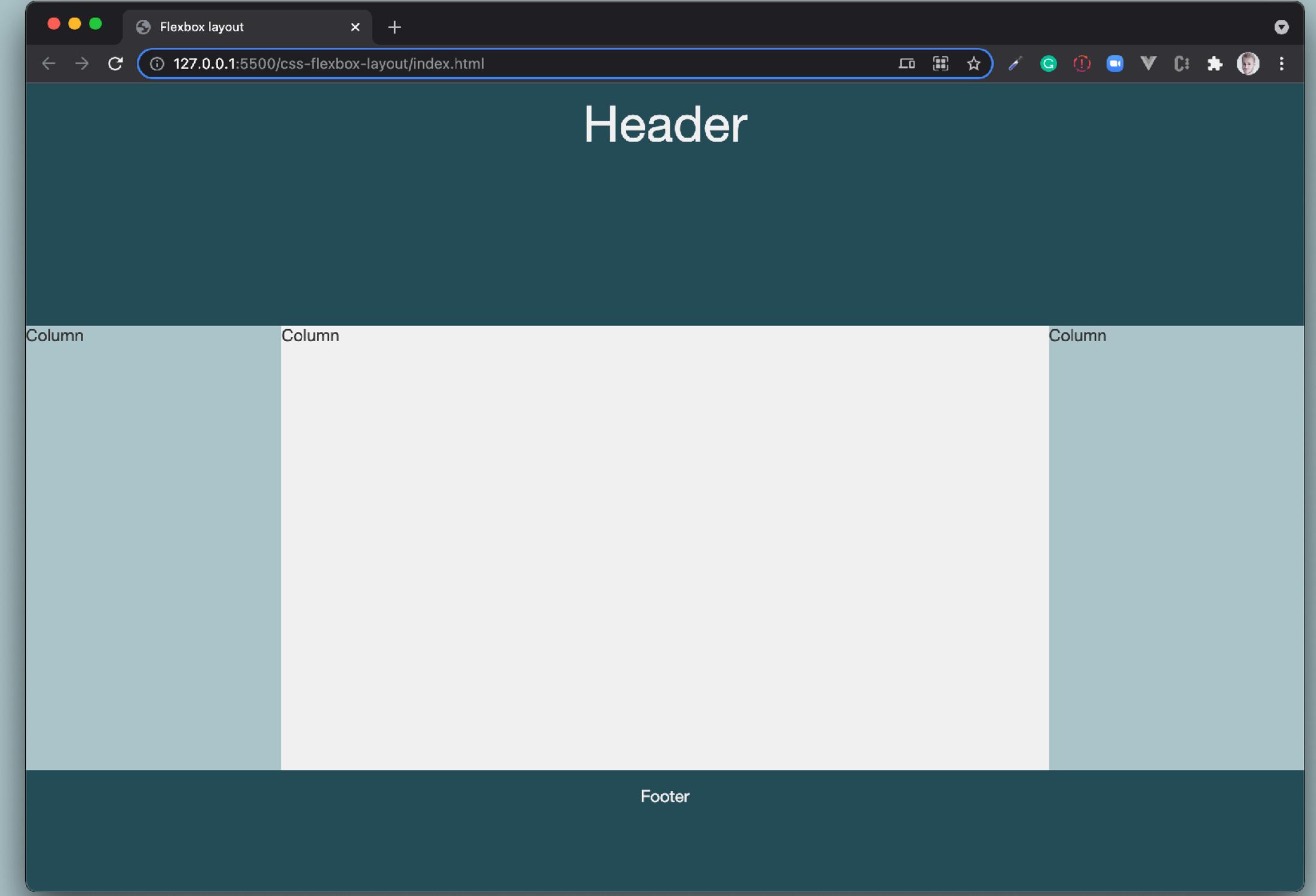
- Finding/ select HTML Elements:
  - id
  - tag name
  - class name
  - CSS selectors

CSS LAYOUT.IO

LAYOUTS AND PATTERNS MADE WITH CSS



css-grid-layout

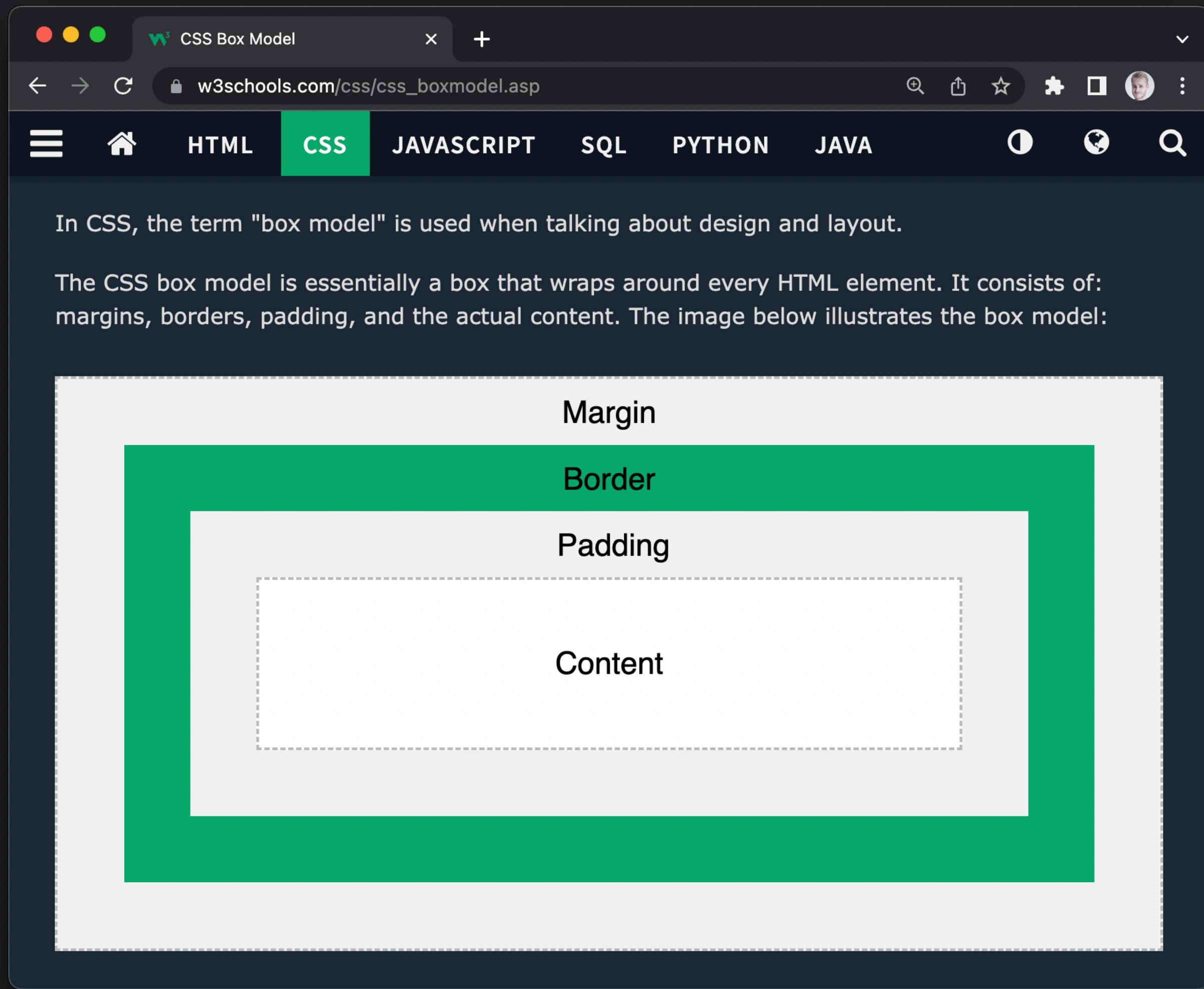


css-flexbox-layout

# CSS templates

[https://www.w3schools.com/css/css\\_templates.asp](https://www.w3schools.com/css/css_templates.asp)

# The CSS Box Model



[https://www.w3schools.com/css/css\\_boxmodel.asp](https://www.w3schools.com/css/css_boxmodel.asp)

# CSS Position

The screenshot shows a web browser window with a dark theme. The title bar reads "CSS Layout - The position Prop x". The address bar shows the URL "w3schools.com/css/css\_positioning.asp". The navigation menu at the top includes links for "HTML", "CSS" (which is highlighted in green), "JAVASCRIPT", and "SQL". Below the menu, a main content area contains the following text:

The `position` property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

## The position Property

The `position` property specifies the type of positioning method used for an element.

There are five different position values:

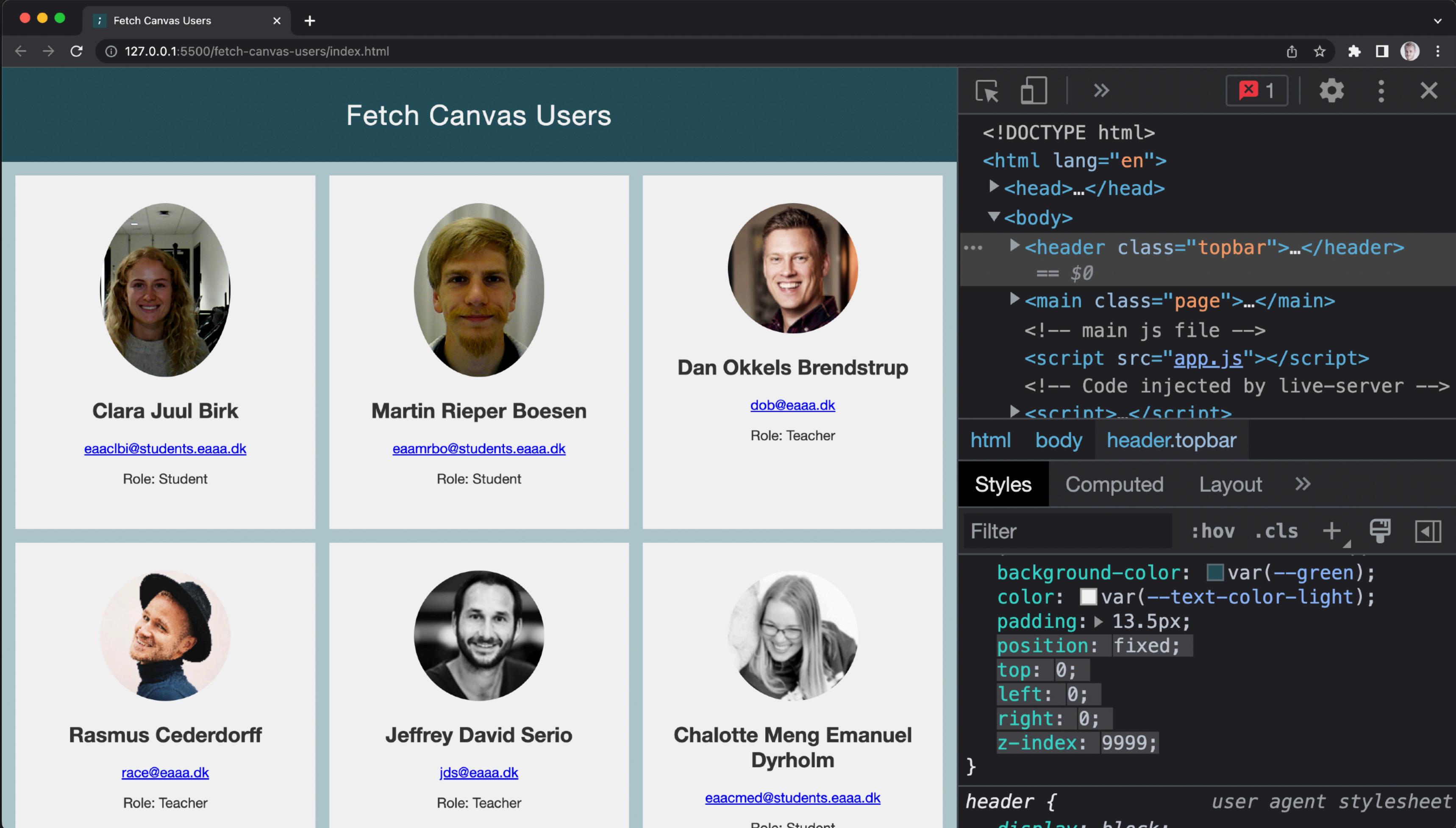
- `static`
- `relative`
- `fixed`
- `absolute`
- `sticky`

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the `position` property is set first. They also work differently depending on the position value.

This <div> element has `position: fixed;`

[https://www.w3schools.com/css/css\\_positioning.asp](https://www.w3schools.com/css/css_positioning.asp)

# Fixed Header



The screenshot shows a web application titled "Fetch Canvas Users" running in a browser. The page displays a grid of six user profiles, each consisting of a circular profile picture and the user's name and contact information. The users are arranged in two rows of three. The first row contains Clara Juul Birk, Martin Rieper Boesen, and Dan Okkels Brendstrup. The second row contains Rasmus Cederdorff, Jeffrey David Serio, and Charlotte Meng Emanuel Dyrholm. The browser's developer tools are open, specifically the element inspector, which highlights the `header` element with a `position: fixed;` style. The inspector also shows the full HTML structure of the page.

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <header class="topbar">...</header>
    == $0
    <main class="page">...</main>
    <!-- main js file -->
    <script src="app.js"></script>
    <!-- Code injected by live-server -->
    <script>...</script>
  </body>
</html>
```

Header styles:

```
background-color: var(--green);
color: var(--text-color-light);
padding: 13.5px;
position: fixed;
top: 0;
left: 0;
right: 0;
z-index: 9999;
```

Header element:

```
header { display: block; }
```

[https://www.w3schools.com/css/css\\_positioning.asp](https://www.w3schools.com/css/css_positioning.asp)

# CSS - Float & Clear

The CSS `float` property specifies how an element should float.

The CSS `clear` property specifies what elements can float beside the cleared element and on which side.

[Float Left](#)      [Float Right](#)

## The float Property

The `float` property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The `float` property can have one of the following values:

- `left` - The element floats to the left of its container
- `right` - The element floats to the right of its container
- `none` - The element does not float (will be displayed just where it occurs in the text). This is default
- `inherit` - The element inherits the float value of its parent

In its simplest use, the `float` property can be used to wrap text around images.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac...



```
img {  
    float: right;  
}
```

# CSS calc

The screenshot shows a web application titled "Fetch Canvas Users" running at `127.0.0.1:5500/fetch-canvas-users/index.html`. The application displays a grid of six user profiles. Each profile card contains a circular user photo, the user's name, their email address, and their role. The profiles are:

- Clara Juul Birk** (`eaacjbi@students.eaaa.dk`, Role: Student)
- Martin Rieper Boesen** (`eaamrbo@students.eaaa.dk`, Role: Student)
- Dan Okkels Brendstrup** (`dob@eaaa.dk`, Role: Teacher)
- Rasmus Cederdorff** (`race@eaaa.dk`, Role: Teacher)
- Jeffrey David Serio** (`jds@eaaa.dk`, Role: Teacher)
- Charlotte Meng Emanuel Dyrholm** (`eaacmed@students.eaaa.dk`, Role: Student)

The developer tools on the right side of the browser window are used to inspect the CSS code. The `Elements` panel shows the DOM structure, and the `Styles` panel highlights the `min-height: calc(100vh - 110px);` rule in the `page` class definition.

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <header>...</header>
    <main>...</main> == $0
      <!-- main js file -->
      <script src="app.js"></script>
      <!-- Code injected by live-server -->
      <script>...</script>
    </main>
  </body>
</html>
```

```
.page {
  min-height: calc(100vh - 110px);
  background: var(--light-grey);
  animation: fadeIn 0.4s;
  padding: 108px 0 0;
}
```

```
main {
  display: block;
```

[https://www.w3schools.com/cssref/func\\_calc.asp](https://www.w3schools.com/cssref/func_calc.asp)

# CSS VARIABLES

`var()` - refer to a defined CSS variable.

CSS variables are declared inside  
the `:root` selector

CSS variables can be accessed through  
the entire CSS document.

Makes the code easier to read (more  
understandable)

Makes it much easier to change the color  
values

[https://www.w3schools.com/css/css3\\_variables.asp](https://www.w3schools.com/css/css3_variables.asp)

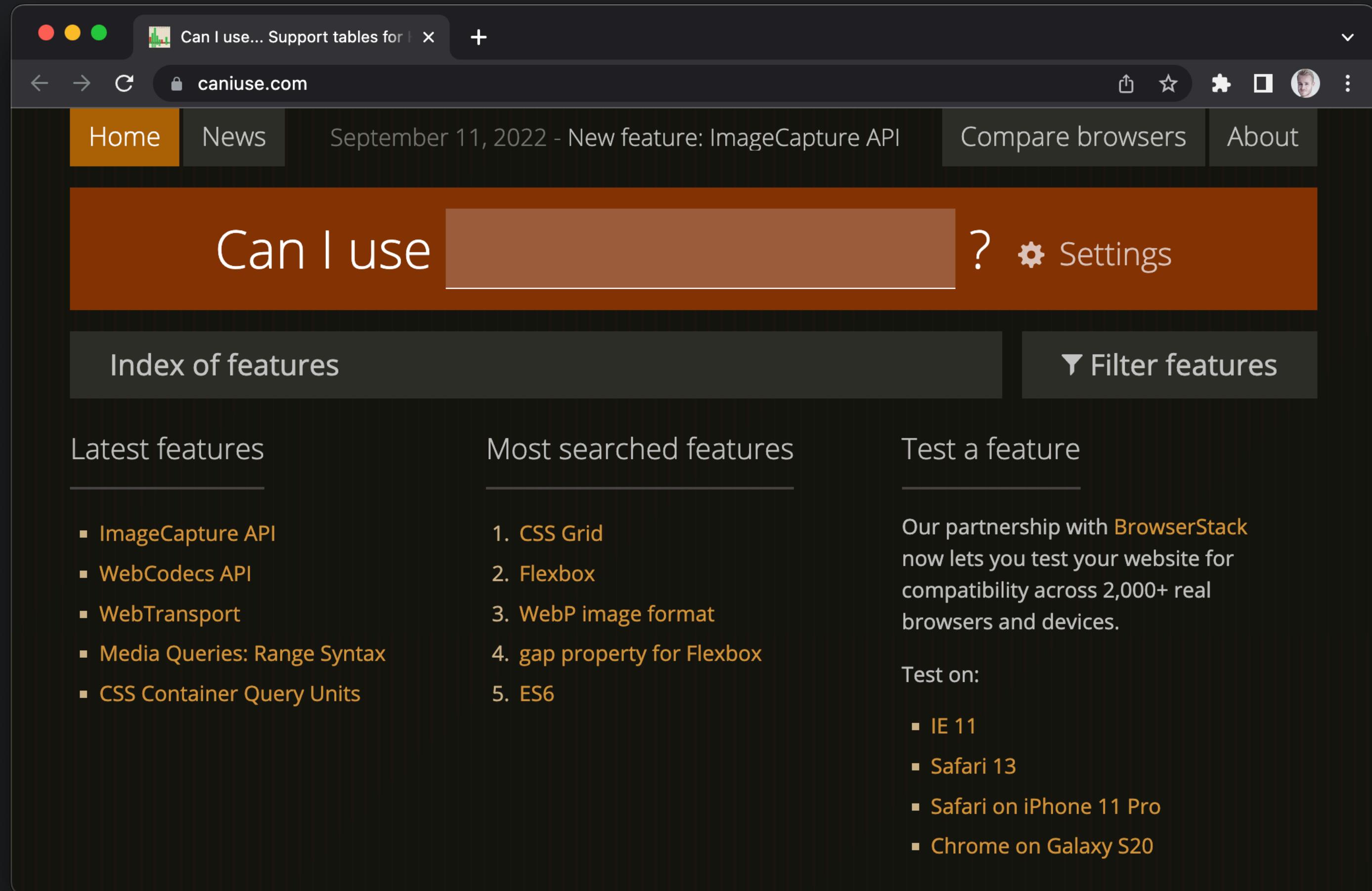
```
/* ----- root variables ----- */
:root {
    --green: #rgb(38, 76, 89);
    --green-opacity: #rgba(38, 76, 89, 0.2);
    --light-green: #rgb(172, 198, 201);
    --light-grey: #f1f1f4;
    --text-color-light: #f1f1f1;
    --text-color-dark: #333;
    --white: #fff;
    --font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
}

/* ----- styling ----- */
html,
body {
    color: var(--text-color-dark);
    font-family: var(--font-family);
    margin: 0;
    padding: 0;
    border: 0;
    outline: 0;
    background-color: var(--light-grey);
}

a {
    color: var(--green);
}

.grid > article {
    text-align: center;
    padding: 2em 1em;
    background-color: var(--light-green);
    transition: 0.3s;
}
```

# Compatibility: Can I use?



The screenshot shows the homepage of the [Can I use...](https://caniuse.com/) website. The page has a dark background with orange and white text. At the top, there's a navigation bar with links for Home, News, a date (September 11, 2022), Compare browsers, and About. Below the navigation is a large orange header with the text "Can I use" and a settings icon. Underneath the header are two buttons: "Index of features" and "Filter features". The main content area is divided into three columns: "Latest features" (listing ImageCapture API, WebCodecs API, WebTransport, Media Queries: Range Syntax, and CSS Container Query Units), "Most searched features" (listing CSS Grid, Flexbox, WebP image format, gap property for Flexbox, and ES6), and "Test a feature" (with a note about a partnership with BrowserStack). A sidebar on the right lists browser compatibility for various features.

Can I use

Home News September 11, 2022 - New feature: ImageCapture API Compare browsers About

Index of features Filter features

Latest features

- ImageCapture API
- WebCodecs API
- WebTransport
- Media Queries: Range Syntax
- CSS Container Query Units

Most searched features

1. CSS Grid
2. Flexbox
3. WebP image format
4. gap property for Flexbox
5. ES6

Test a feature

Our partnership with [BrowserStack](#) now lets you test your website for compatibility across 2,000+ real browsers and devices.

Test on:

- IE 11
- Safari 13
- Safari on iPhone 11 Pro
- Chrome on Galaxy S20

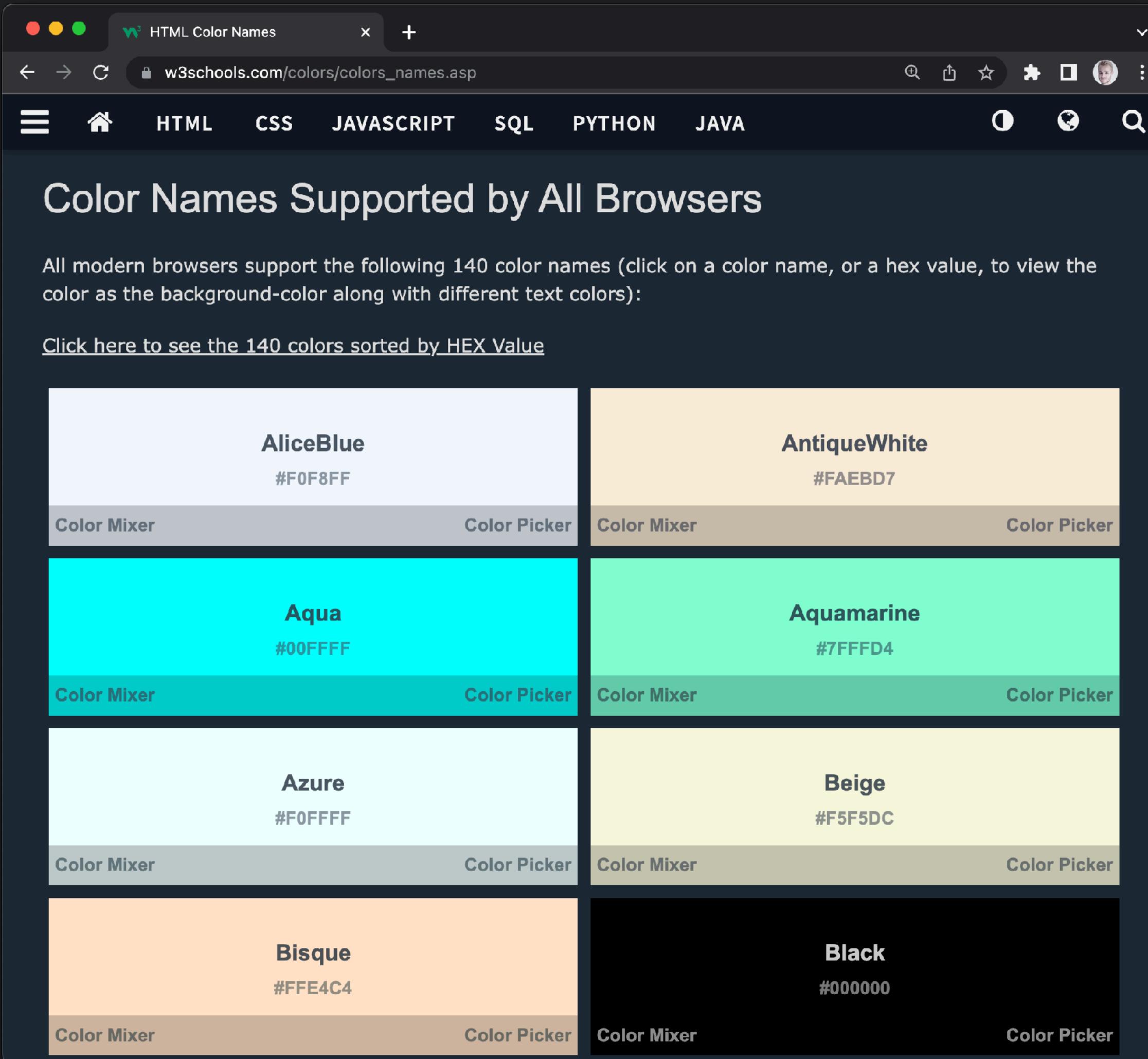
<https://caniuse.com/>

# Further CSS

Overflow: [https://www.w3schools.com/css/css\\_overflow.asp](https://www.w3schools.com/css/css_overflow.asp)

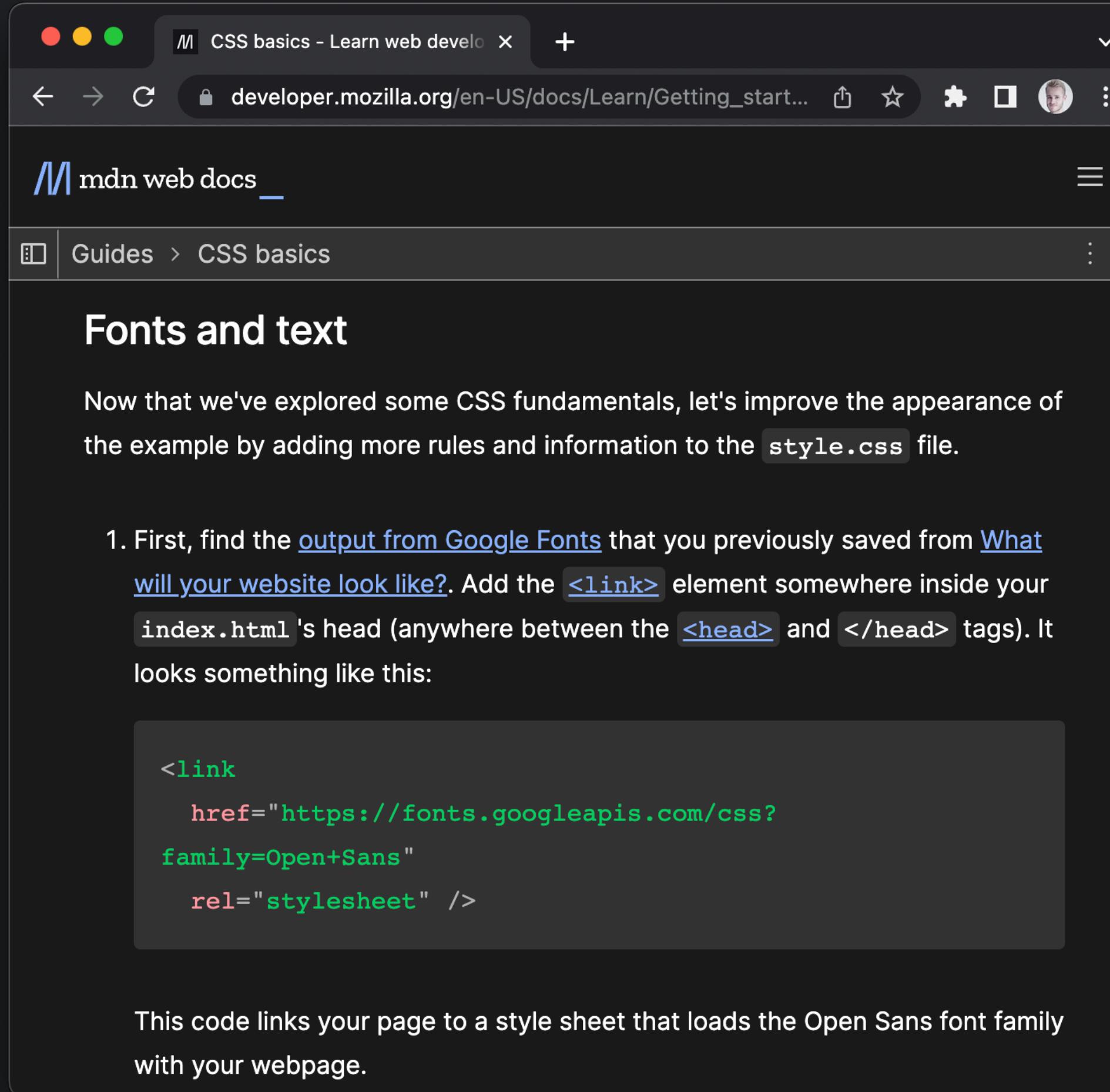
CSS Tutorial: <https://www.w3schools.com/css/default.asp>

# Web Safe Colors



[https://www.w3schools.com/colors/colors\\_names.asp](https://www.w3schools.com/colors/colors_names.asp)

# Fonts



The screenshot shows a dark-themed web browser window. The address bar displays 'developer.mozilla.org/en-US/docs/Learn/Getting\_started\_with\_the\_web/CSS\_basics/Fonts\_and\_text'. The main content area is titled 'Fonts and text'. It contains text about improving page appearance using CSS and provides an example of a `<link>` element for loading the Open Sans font from Google Fonts.

```
<link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet" />
```

This code links your page to a style sheet that loads the Open Sans font family with your webpage.

- Read: [Fonts and text](#)
- Use “Open Sans” in your solution.
- Types:
  - Web Safe: universally installed across all browsers and devices.
  - Font Files (ex. [cederdorff.com](#)).  
[@font-face rule](#)
  - External Fonts (scripts): link or import from external font resources like [Google Font](#) and [Fontshare](#)

# Fonts

The screenshot shows the Google Fonts website for the Roboto font family. The main content area displays various font styles (Light 300, Light 300 Italic, Regular 400, Regular 400 Italic, Medium 500) of the Roboto font in a large, white, sans-serif font against a dark background. Above the specimens, there's a navigation bar with links for Google Fonts, Fonts, Icons, Knowledge, FAQ, and a download button. To the right, a sidebar titled 'Selected family' lists the chosen styles: Roboto, Light 300, Light 300 Italic, and Regular 400. Below this, sections for 'Use on the web' and 'CSS rules to specify families' provide code snippets for embedding the font in HTML and CSS respectively.

- Explore [fonts.google.com](https://fonts.google.com)
- Select fonts and use in your CSS.

# Fonts

The screenshot shows the Fontshare website for the 'Satoshi' font family. At the top, there's a navigation bar with tabs for Styles, Glyphs, Layouts, Details, and License. Below the navigation, there are sections for 'Your Text', 'Cities', 'Excerpts', and 'Names'. A font preview area shows 'Regular', 'Italic', and 'Medium' styles. On the right, a modal window titled '3 Styles Selected' has a 'Review' button and a prominent 'Use' button. Below the button, instructions say 'Copy the code below to <head> of your HTML file' and provide a link to the API. It also says 'And include these CSS rules' and shows the following CSS code:

```
<link href="https://api.fontshare.com/v2/css?f[]="satoshi@500,300,400&display=swap" rel="stylesheet">  
  
font-family: 'Satoshi', sans-serif;
```

At the bottom of the modal, it says 'Use Code to access selected fonts from Fontshare servers.' and 'Download Fonts to use them locally or self-host them.' There's also a 'Download Fonts' button and a '+ Add Style' link.

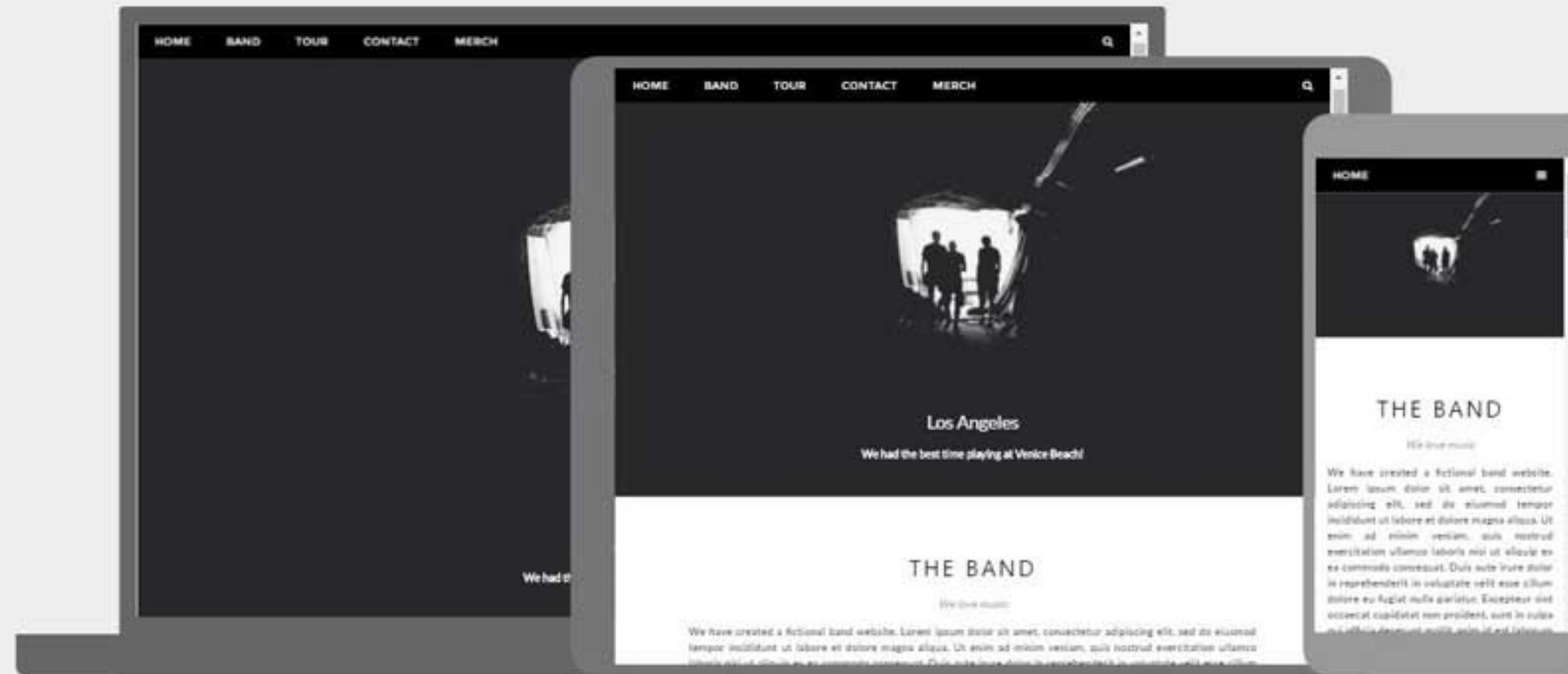
- Google Fonts alternative:  
[fontshare.com](#)
- Select fonts and use in your CSS.

# Responsive Web Design



- Create Web Design that looks good across all devices!
- Automatically adjust for different screen sizes and viewports.
- Resize, hide, shrink, or enlarge.

# Resize, hide, shrink, enlarge



[https://www.w3schools.com/w3css/tryw3css\\_templates\\_band.htm](https://www.w3schools.com/w3css/tryw3css_templates_band.htm)  
[All W3School Templates](#)

# Responsive Images



HTML Responsive Web Design

Not Secure | w3schools.com/html/html\_r...

HTML CSS JAVASCRIPT

## Responsive Images

Responsive images are images that scale nicely to fit any browser size.

### Using the width Property

If the CSS `width` property is set to 100%, the image will be responsive and scale up and down:

```

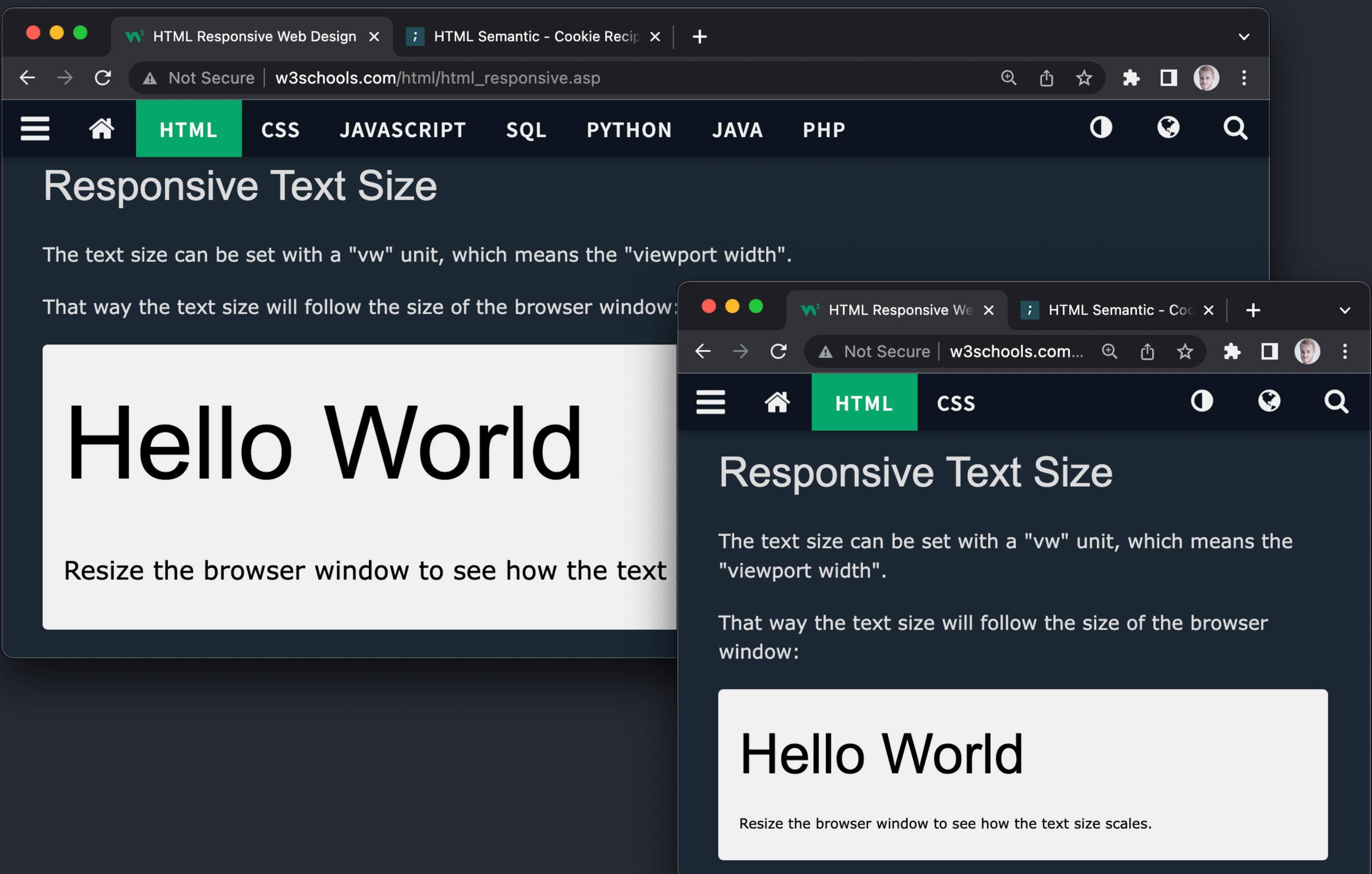
```

```
img {  
    max-width: 100%;  
    height: auto;  
}
```

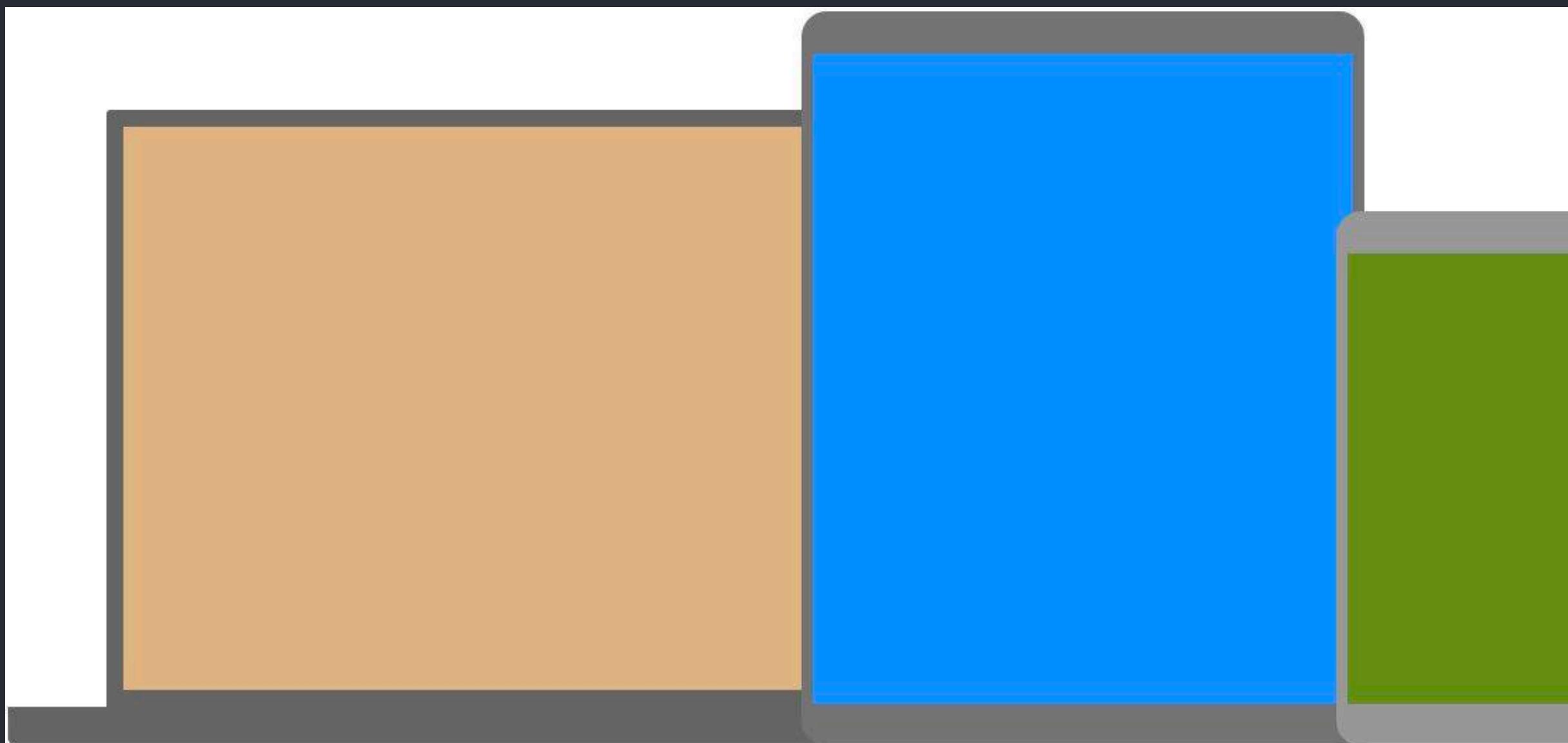
# Responsive Text Size

```
<h1>Cookies</h1>
```

```
h1 {  
    font-size: 10vw;  
}
```



# Media Queries



```
/* Set the background color of body to tan */
body {
    background-color: tan;
}

/* On screens that are 992px or less, set the background color to blue */
@media screen and (max-width: 992px) {
    body {
        background-color: blue;
    }
}

/* On screens that are 600px or less, set the background color to olive */
@media screen and (max-width: 600px) {
    body {
        background-color: olive;
    }
}
```

# Media Queries



The screenshot shows a web browser window with the title "Responsive Web Design Media". The address bar displays "w3schools.com/css/css\_rwd\_mediaqueries.asp". The navigation bar includes links for HTML, CSS (which is highlighted), JAVASCRIPT, SQL, and PYTHON. Below the navigation bar, there is a section titled "Example" containing code and a "Try it Yourself" button. Another section titled "Add a Breakpoint" contains text and a diagram illustrating media queries.

Media query is a CSS technique introduced in CSS3.

It uses the `@media` rule to include a block of CSS properties only if a certain condition is true.

**Example**

If the browser window is 600px or smaller, the background color will be lightblue:

```
@media only screen and (max-width: 600px) {  
  body {  
    background-color: lightblue;  
  }  
}
```

[Try it Yourself »](#)

**Add a Breakpoint**

Earlier in this tutorial we made a web page with rows and columns, and it was responsive, but it did not look good on a small screen.

Media queries can help with that. We can add a breakpoint where certain parts of the design will behave differently on each side of the breakpoint.



The diagram illustrates a responsive design. On the left, a "Desktop" icon shows a wide screen with a purple header, a white main content area with three blue horizontal bars, and a blue footer. On the right, a "Phone" icon shows a much narrower screen with the same purple header, white content area, and blue footer, demonstrating how the layout adapts to a smaller screen.

<https://www.w3schools.com/j/s/default.asp>

```
.header h1 {  
  font-size: 1.3em;  
  margin-top: 0;  
  margin: 10px 0 0;  
}  
  
@media (min-width: 768px) {  
  .header h1 {  
    font-size: 1.5em;  
    margin: 0;  
  }  
}  
  
@media (min-width: 992px) {  
  .header h1 {  
    font-size: 1.8em;  
  }  
}
```

# Design for Mobile First

Start with mobile, then add CSS for tablet,  
desktop and other screen sizes.

[https://www.w3schools.com/css/css\\_rwd\\_mediaqueries.asp](https://www.w3schools.com/css/css_rwd_mediaqueries.asp)

The screenshot shows a browser window with two tabs: "Responsive Web Design Media" and "CSS3 Media Queries - Example". The main content area displays a "Media Queries For Menus" section. It includes a description of how media queries are used to create a responsive navigation menu that changes design based on screen size. Below this, there are two examples: "Large screens:" showing a horizontal navigation bar with four links ("Home", "Link 1", "Link 2", "Link 3"), and "Small screens:" showing a vertical dropdown menu where the "Home" link is at the top, followed by "Link 1", "Link 2", and "Link 3". At the bottom, there is an "Example" section containing the CSS code for the menu.

```
/* The navbar container */
.topnav {
  overflow: hidden;
  background-color: #333;
}

/* Navbar links */
.topnav a {
  float: left;
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

/* On screens that are 600px wide or less, make the menu links stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {
  .topnav a {
    float: none;
    width: 100%;
  }
}
```

# Media Queries & Menus

```
/* The navbar container */
.topnav {
  overflow: hidden;
  background-color: #333;
}

/* Navbar links */
.topnav a {
  float: left;
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

/* On screens that are 600px wide or less, make the menu links stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {
  .topnav a {
    float: none;
    width: 100%;
  }
}
```

# Media Queries For Columns

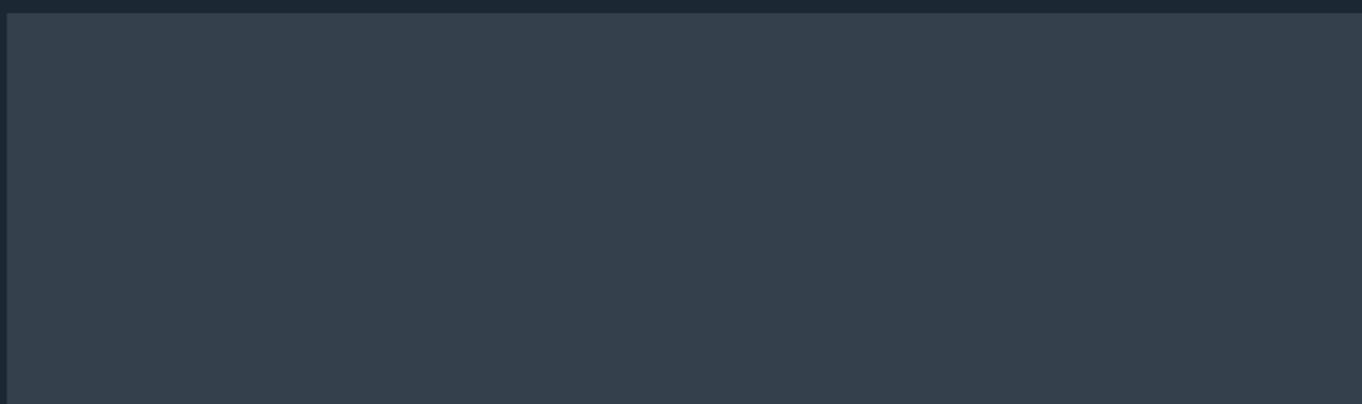
Large screens:



Medium screens:



Small screens:

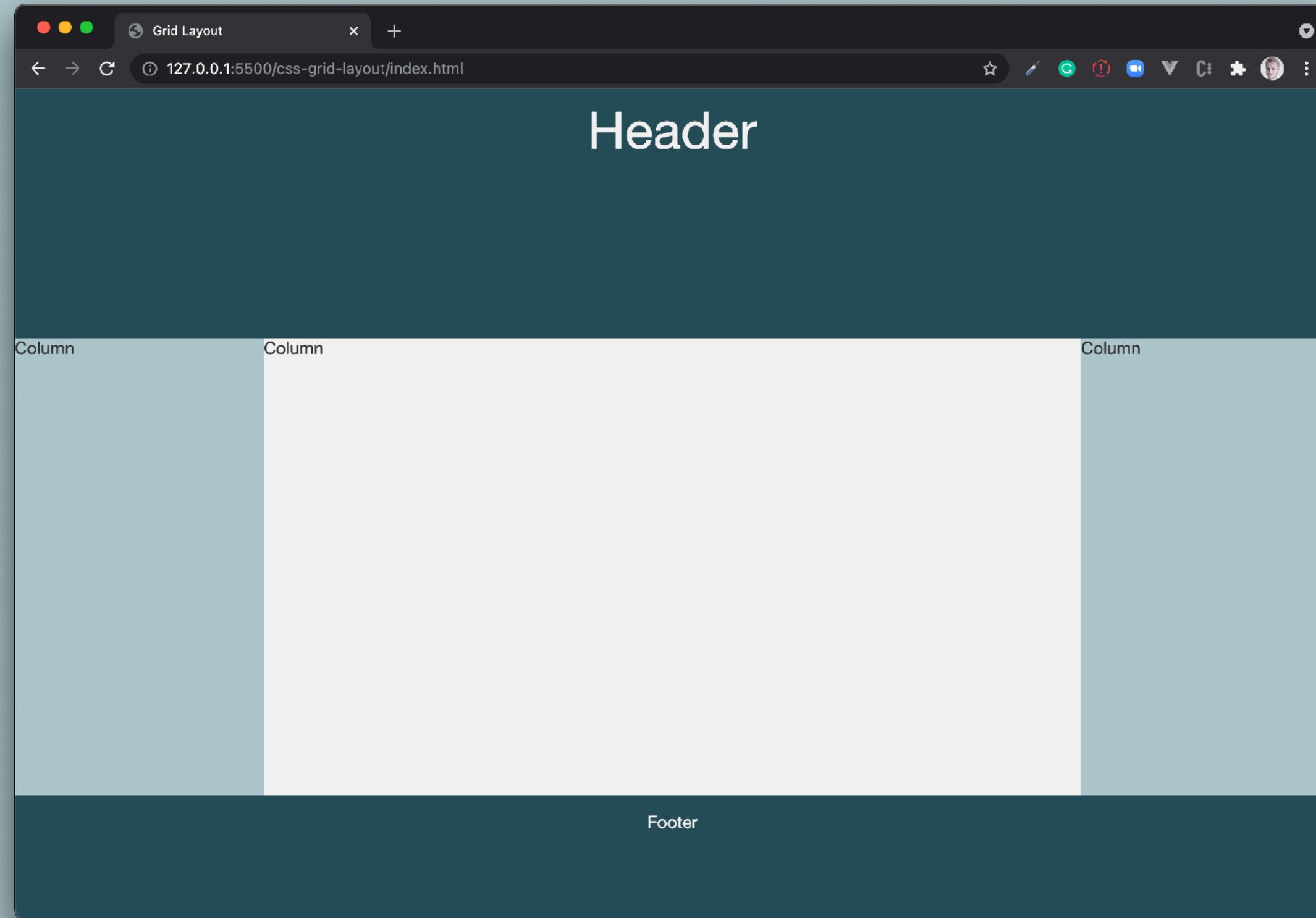


```
/* Create four equal columns that floats next to each other */
.column {
    float: left;
    width: 25%;
}

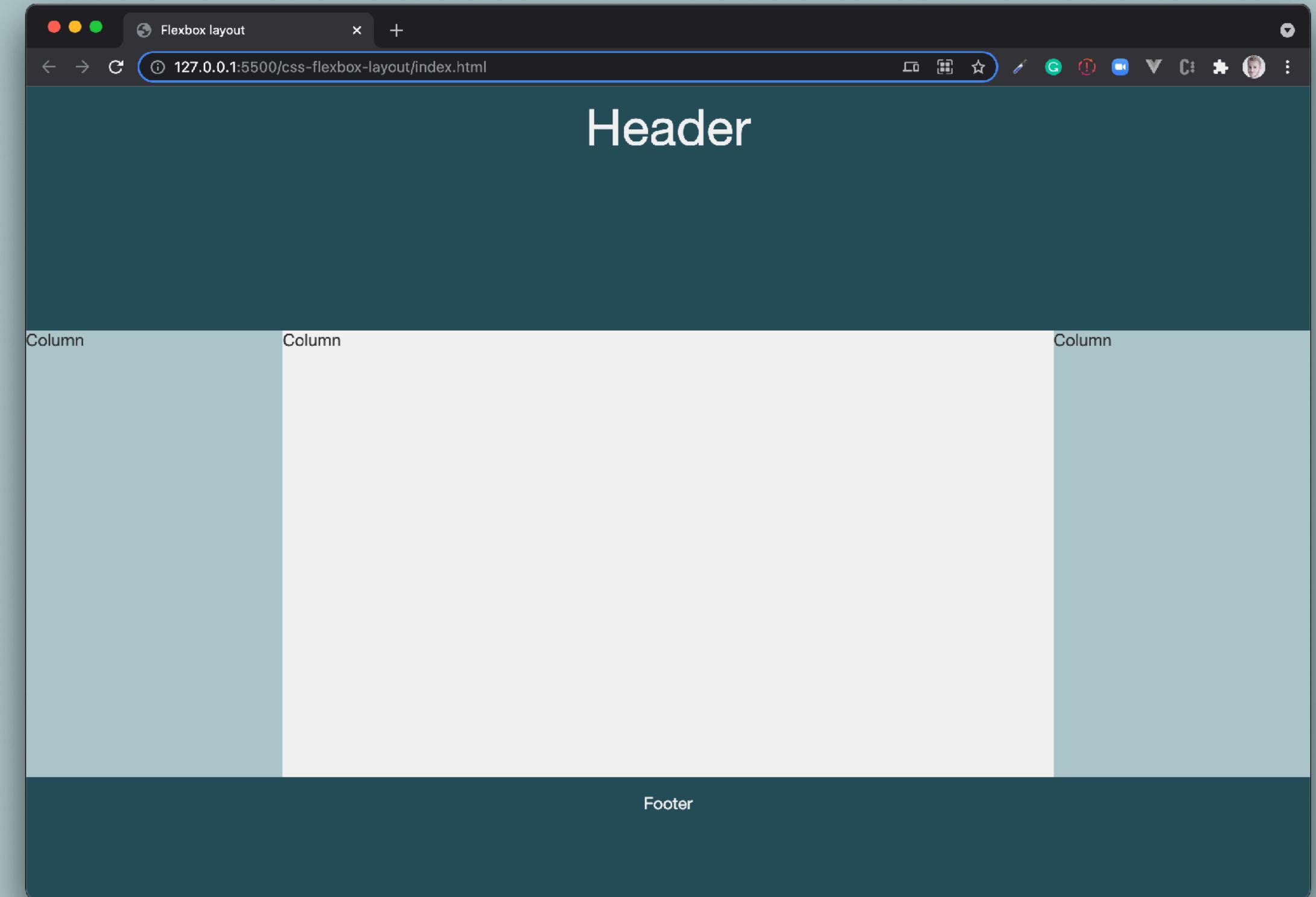
/* On screens that are 992px wide or less, go from four columns to two columns */
@media screen and (max-width: 992px) {
    .column {
        width: 50%;
    }
}

/* On screens that are 600px wide or less, make the columns stack on top of each other instead */
@media screen and (max-width: 600px) {
    .column {
        width: 100%;
    }
}
```

# Grids



css-grid-layout



css-flexbox-layout

# Responsive Web Design Templates

[https://www.w3schools.com/css/css\\_rwd\\_templates.asp](https://www.w3schools.com/css/css_rwd_templates.asp)

# Hvordan er “formen”?

Tak for i dag

Webudvikling

**kea**  
KØBENHAVNS ERHVERVSAKADEMI