

The Web (HTML + CSS)

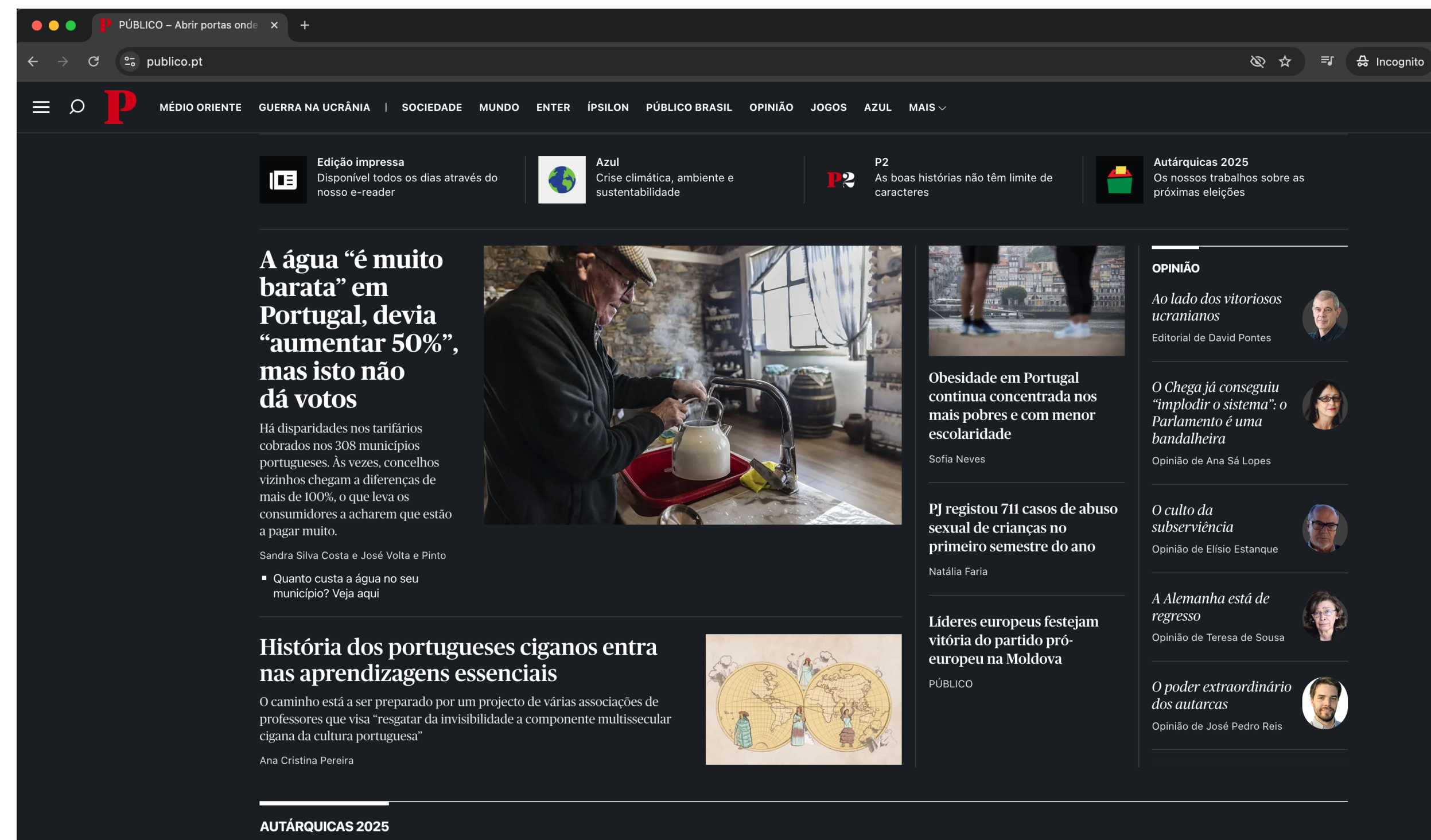
Interface Pessoa-Máquina - 25/26 - LEI / UM

Hugo Pacheco
hpacheco@di.uminho.pt

The Web: uma revolução?

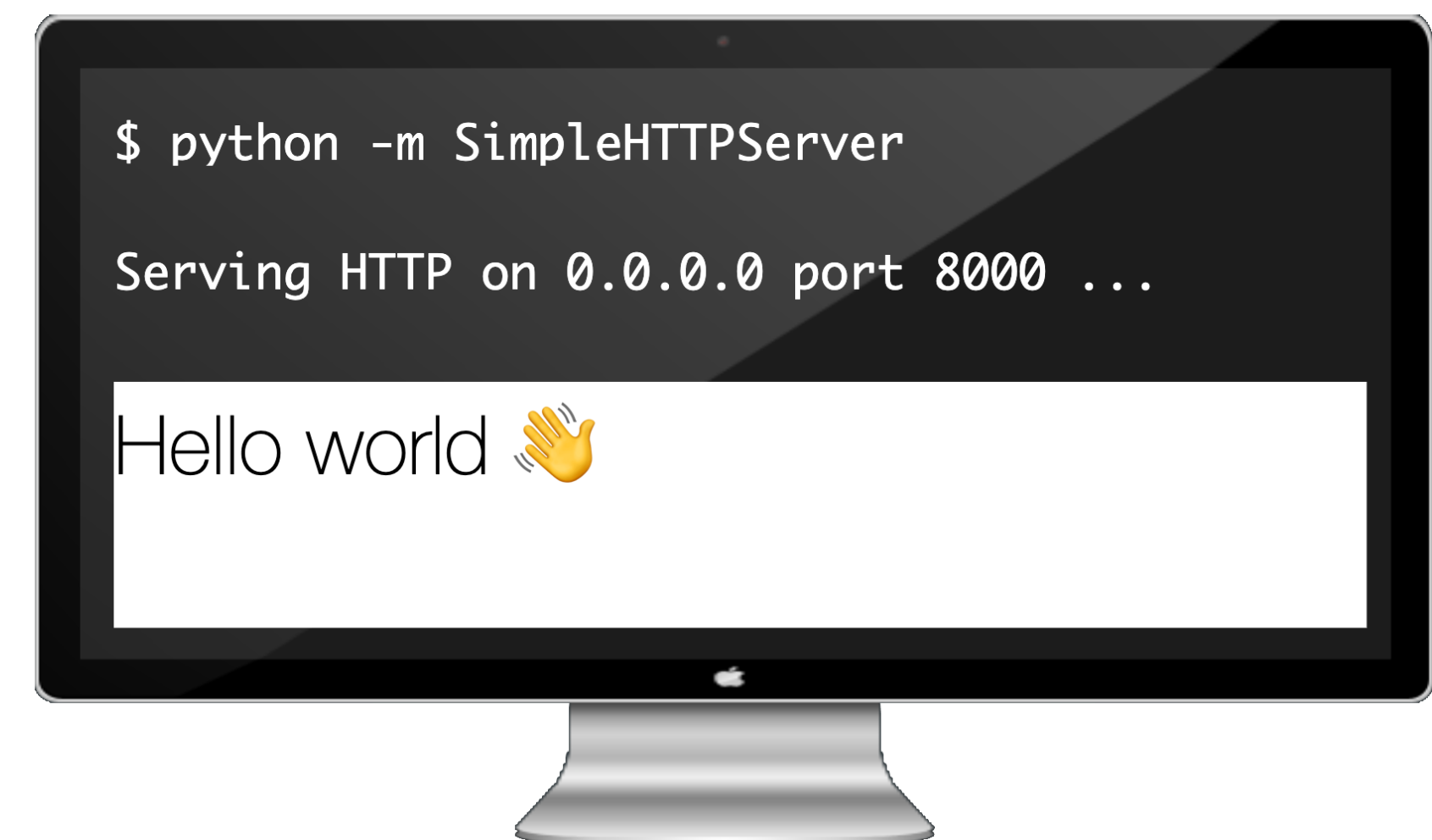
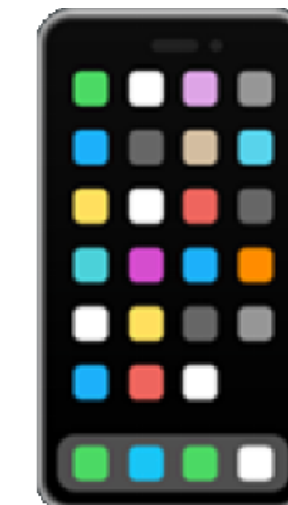
- Internet, antes da Web
 - Computadores interligados
 1. Conectar a servidor FTP
 2. Pesquisar ficheiros
 3. Download de recursos
 4. Abrir localmente com o programa certo
- Internet, depois da Web
 - Abrir hiperligação no browser

```
phoenixnap@test-system:~$ ftp 192.168.100.9
Connected to 192.168.100.9.
220 (vsFTPd 3.0.3)
Name (192.168.100.9:phoenixnap): phoenixnap
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

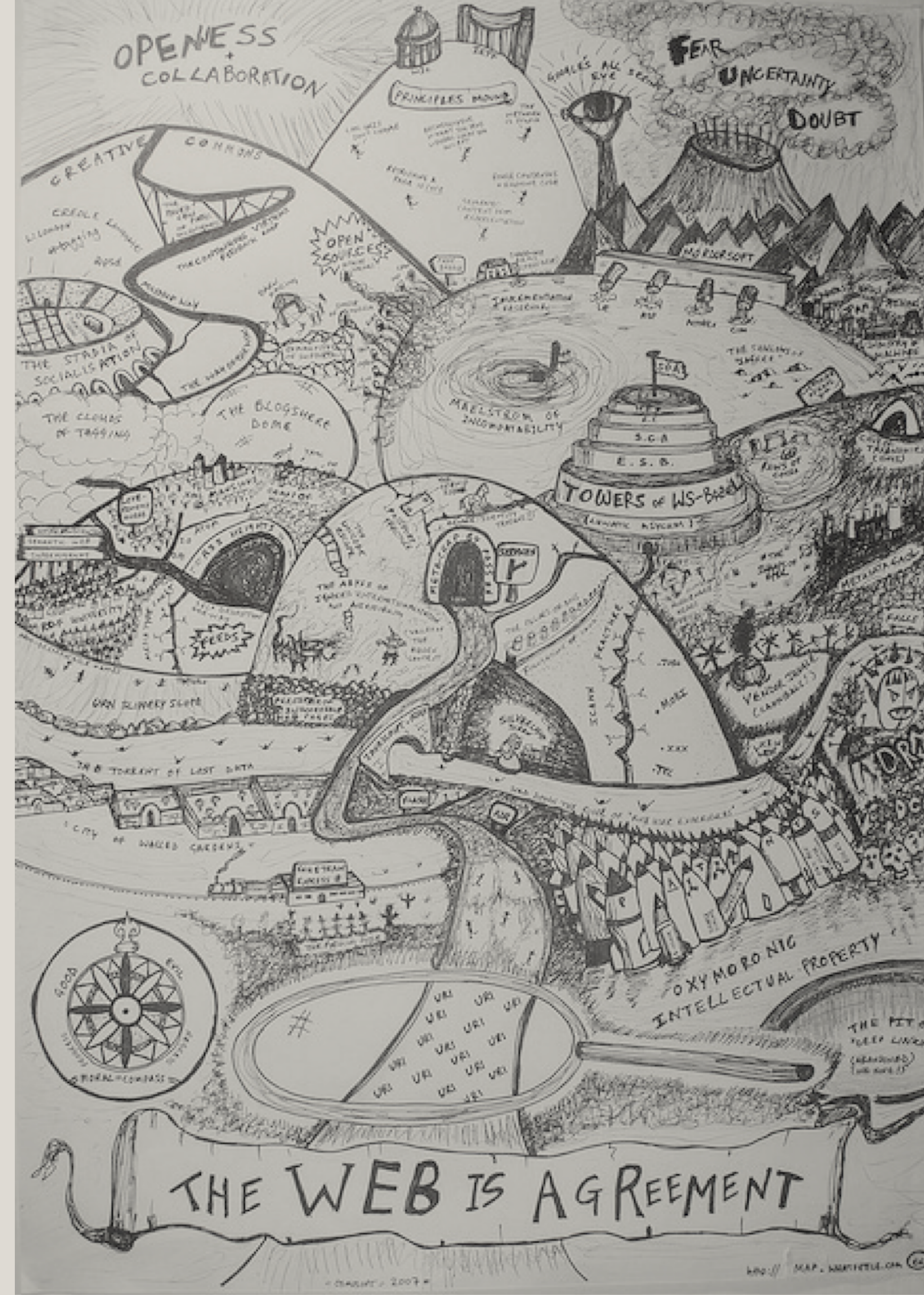


The Web: uma revolução de usabilidade

- Computadores ligados são
 - servidores: fornecedores de conteúdos (hosting de páginas)
 - clientes: utilizadores (utilizando diversos dispositivos)
- Utilizadores só precisam de um tipo de software instalado
 - um browser web
- (O mesmo computador pode ser ambos)



The Web



The Web: usabilidade é sobrevivência

“If a website is difficult to use, people **leave**.

If the homepage fails to clearly state what a company offers and what users can do on the site, people **leave**.

If users get lost on a website, they **leave**.

If a website's information is hard to read or doesn't answer users' key questions, they **leave**.

Note a pattern here? There's no such thing as a **user reading a website manual** or otherwise spending much time **trying to figure out an interface**.

There are plenty of other websites available; **leaving is the first line of defense** when users encounter a difficulty.”

(Nielsen Norman Group)

The Web

- O que acontece quando abrimos um URL no browser?
 1. Cliente envia um HTTP request
 2. Servidor recebe o request
 3. Servidor responde com um HTTP response
 4. O browser interpreta o HTML
 5. O browser interpreta e mostra o HTML



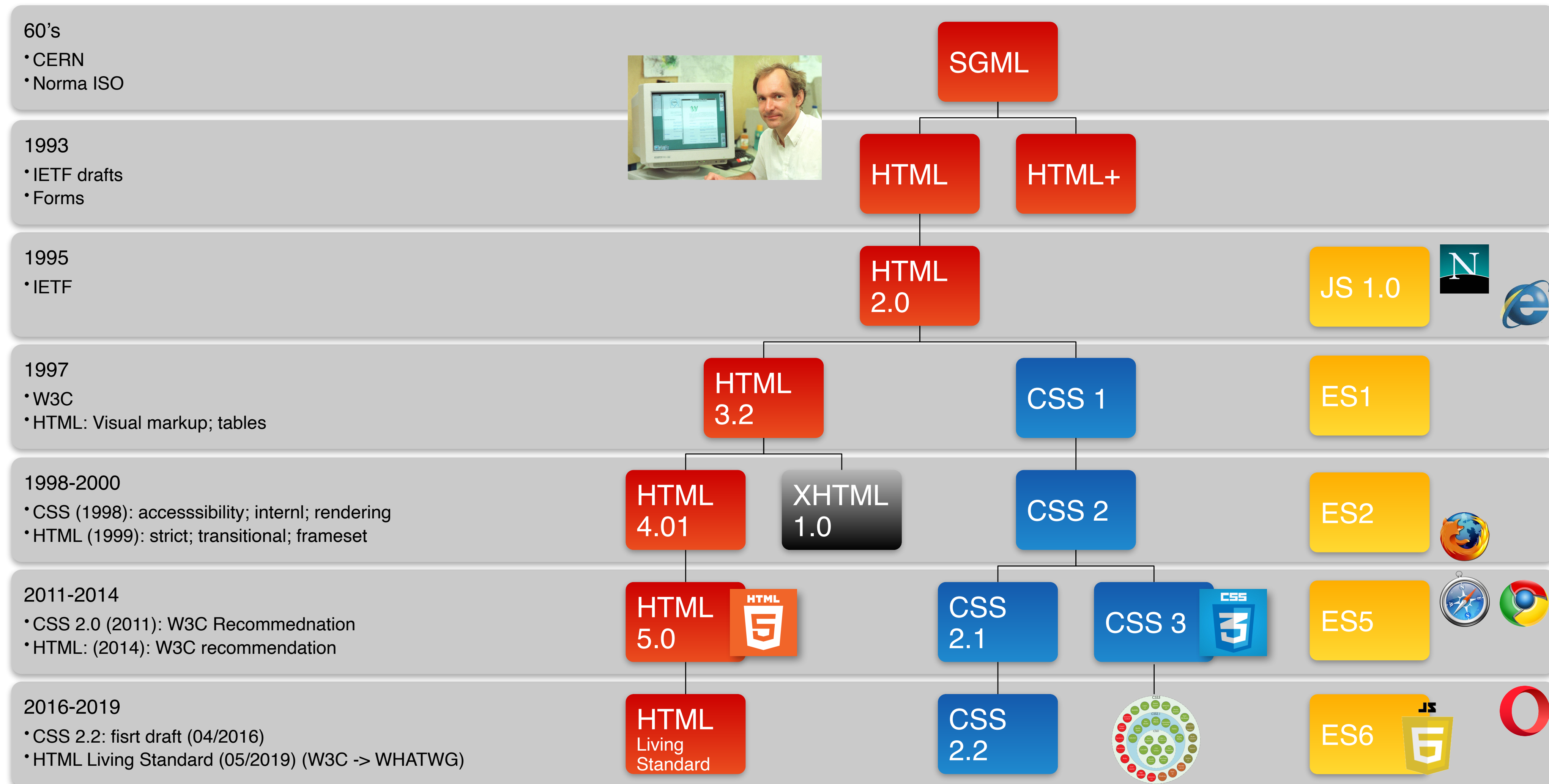
```
GET /hello.html HTTP/1.1
User-Agent: Mozilla/5.0 Firefox/71.0
Host: haystack.csail.mit.edu
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

```
HTTP/1.1 200 OK
Date: Sun, 29 Dec 2019 17:46:48 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Wed, 15 Nov 2017 21:16:50 GMT
Content-Length: 6471
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE html>
<title>Hello world</title>
<p>Hello <em>world</em> 🙌
```



The Web: um bocado de História



The Web: uma tríade atual

Hierarquia, navegação,
significado, conteúdo

Tipografia, cores,
layout, efeitos
visuais, animação



Interação e automação;
linguagem general-
purpose que corre no
cliente

Exemplo: HTML + CSS + JS

HTML

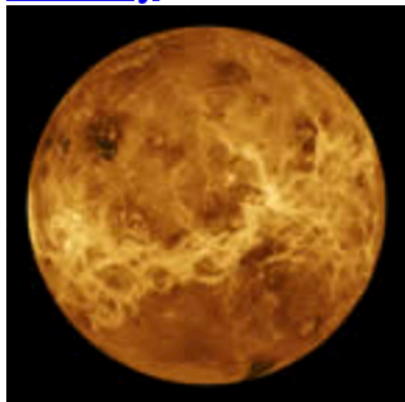
Our Solar System



[Sol](#)



[Mercury](#)

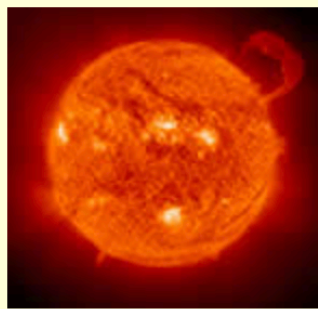


[Venus](#)

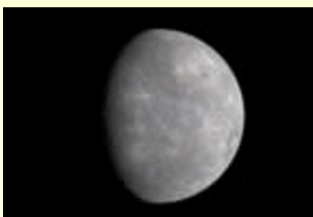


HTML + CSS

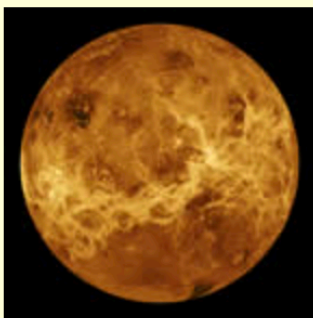
Our Solar System



[Sol](#)



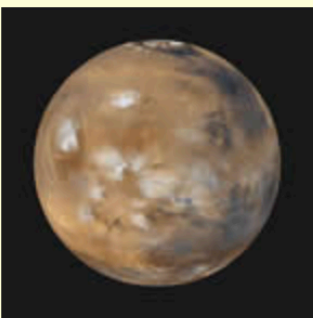
[Mercury](#)



[Venus](#)



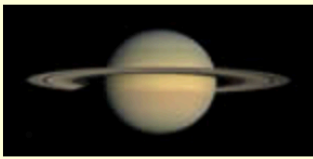
[Earth](#)



[Mars](#)



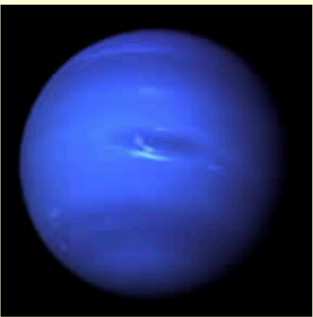
[Jupiter](#)



[Saturn](#)



[Uranus](#)



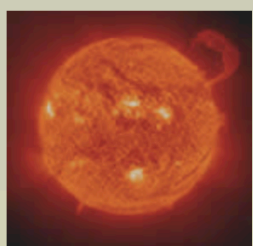
[Neptune](#)



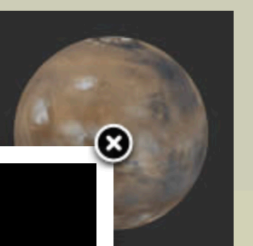
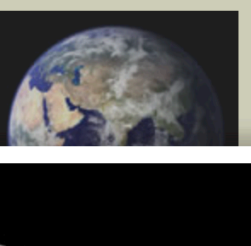
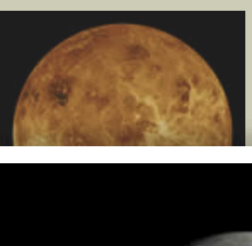
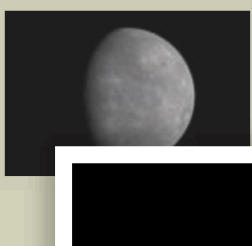
[Pluto and Charon](#)

HTML + CSS + JS

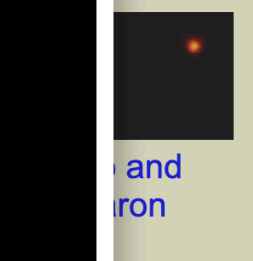
Our Solar System



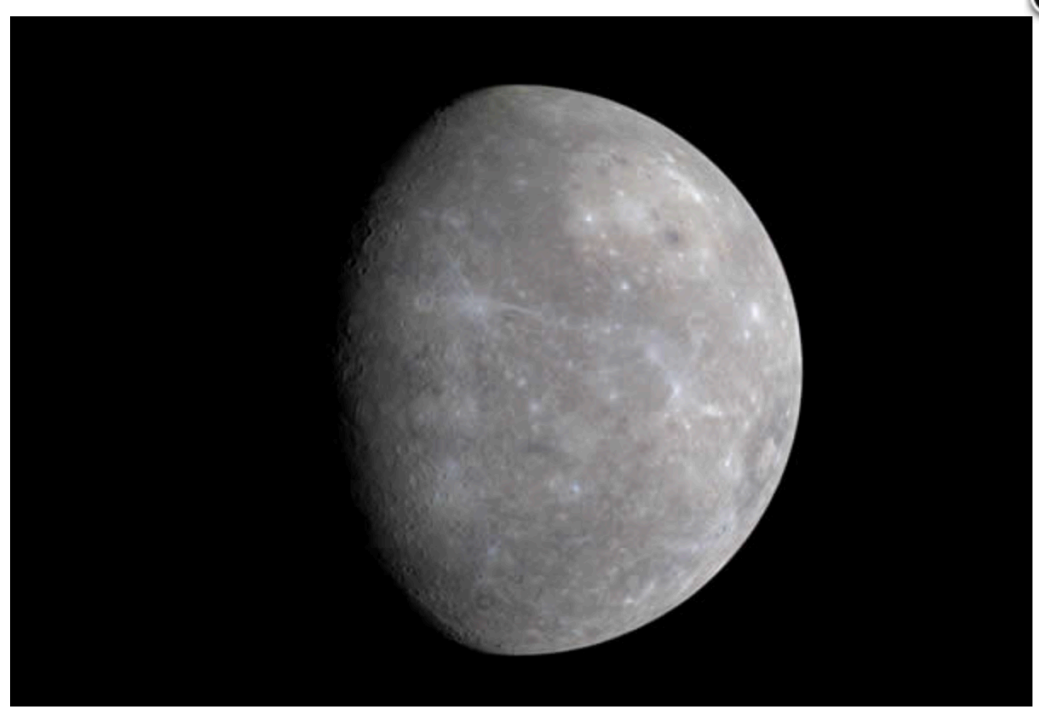
[Sol](#)



[Jupiter](#)



Images courtesy of:
Sol: ESA/NASA/SOHO; Mercury: NASA; Venus: NASA/JPL/MSSS; Earth: NASA, [The Visible Earth](#); Mars: NASA/JPL/MSSS; Jupiter: NASA; Saturn: NASA/JPL; Uranus: NASA/JPL; Neptune: NASA/JPL; Pluto: Dr. R. Albrecht, NASA/JPL.



[Mercury](#)

Actividade: “Despir” um website

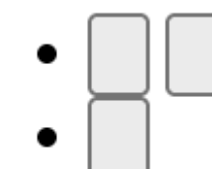
1. Ir a um site, e.g., de um jornal
2. Abrir as Developer Tools do browser
3. Remover **CSS** e **JS**
 - tags `<style>`, `<script>` e `<link>`
4. Recarregar o site

PÚBLICO

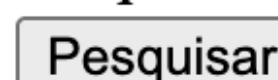
- [Médio Oriente](#)
- [Guerra na Ucrânia](#)
- [Sociedade](#)
- [Mundo](#)
- [Enter](#)
- [Ípsilon](#)
- [Público Brasil](#)
- [Opinião](#)
- [Jogos](#)
- [Azul](#)



- [Oferecer assinatura](#)



- Pesquise no Público



Segunda-feira, 29 Set 2025

[Edição impressa](#)

Temas em destaque



[Edição impressa](#)

Disponível todos os dias através do nosso e-reader



[Azul](#)

Crise climática, ambiente e sustentabilidade



[Autárquicas 2025](#)

Os nossos trabalhos sobre as próximas eleições



[Cinecartaz](#)

Estreias da semana, críticas e programação

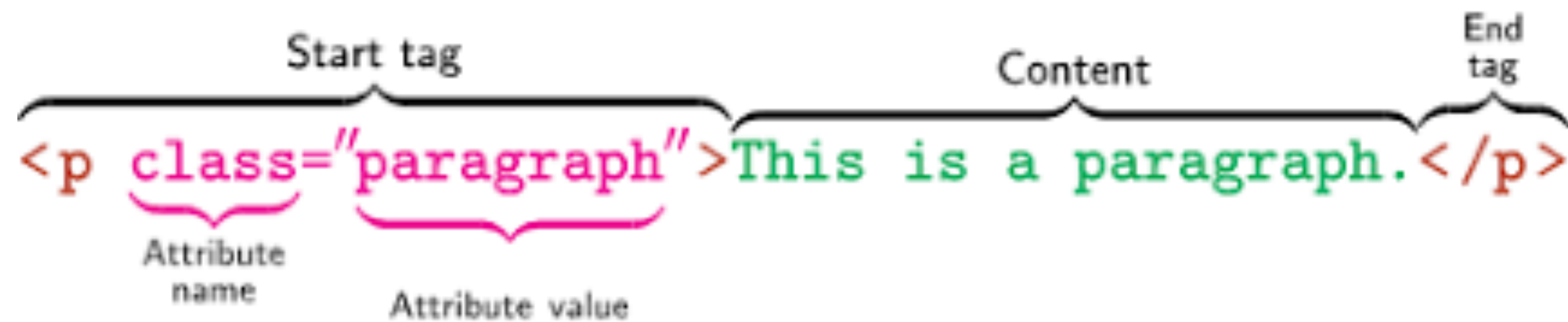
The Web: tecnologias

- Idealmente, independência completa de conceitos
 - **HTML** (estrutura) + **CSS** (apresentação) + **JS** (comportamento)
- Na verdade, alguma fusão de conceitos / tecnologias
 - **HTML 3.2** com tags de formatação (`<em>`, `<b>`, etc)
 - **CSS** influencia frequentemente a estrutura do **HTML**
 - Comum introduzir `<div>` apenas para efeitos de formatação
 - E.g., `grid-template` impõe utilização de tags (e.g. <https://grid.layoutit.com/>)
 - Em **JS** vale tudo (o que o browser deixa)... alterar o **HTML** e o **CSS** dinamicamente
 - ...

HTML

Elementos HTML

- **HyperText Markup Language** (HTML) é uma markup language (um conjunto de tags)
- Anatomia de uma tag / elemento HTML (dois modos)



- **Programação declarativa**
 - O quê? HTML descreve a estrutura e conteúdo de uma página
 - Como? Browser trata da renderização
- HTML Living Standard (25/09/2025)

Exemplo: uma página HTML

1. Visualizar em <https://codepen.io/pen>

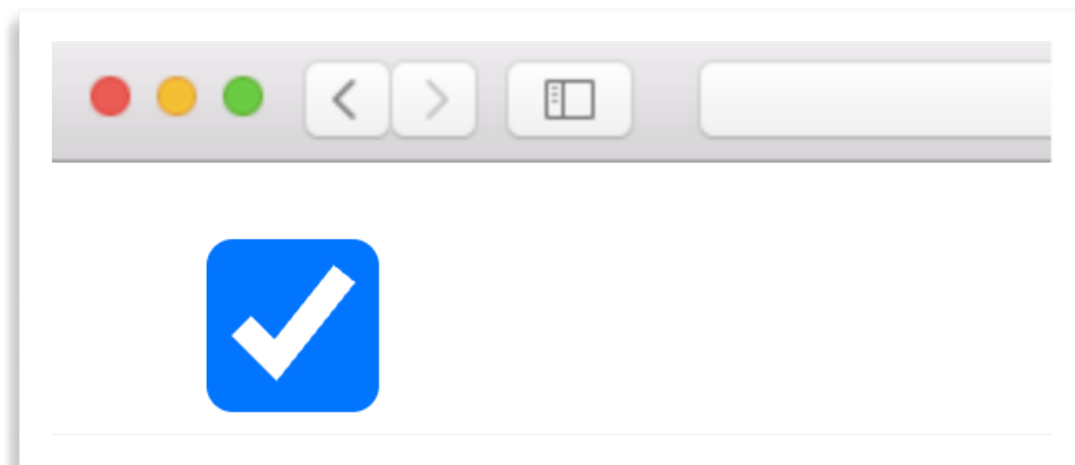
2. Abrir Developer
Tools do browser

3. O mesmo código?

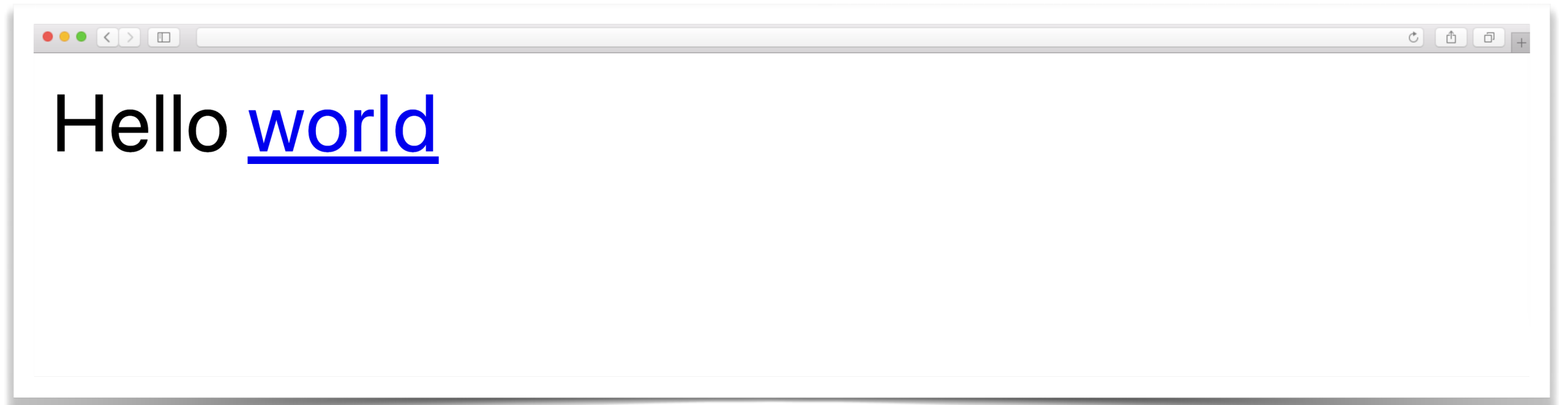
```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Hello world</title>
</head>
<body>
  <p>Hello <em>world</em> 🙌</p>
</body>
</html>
```


Atributos HTML

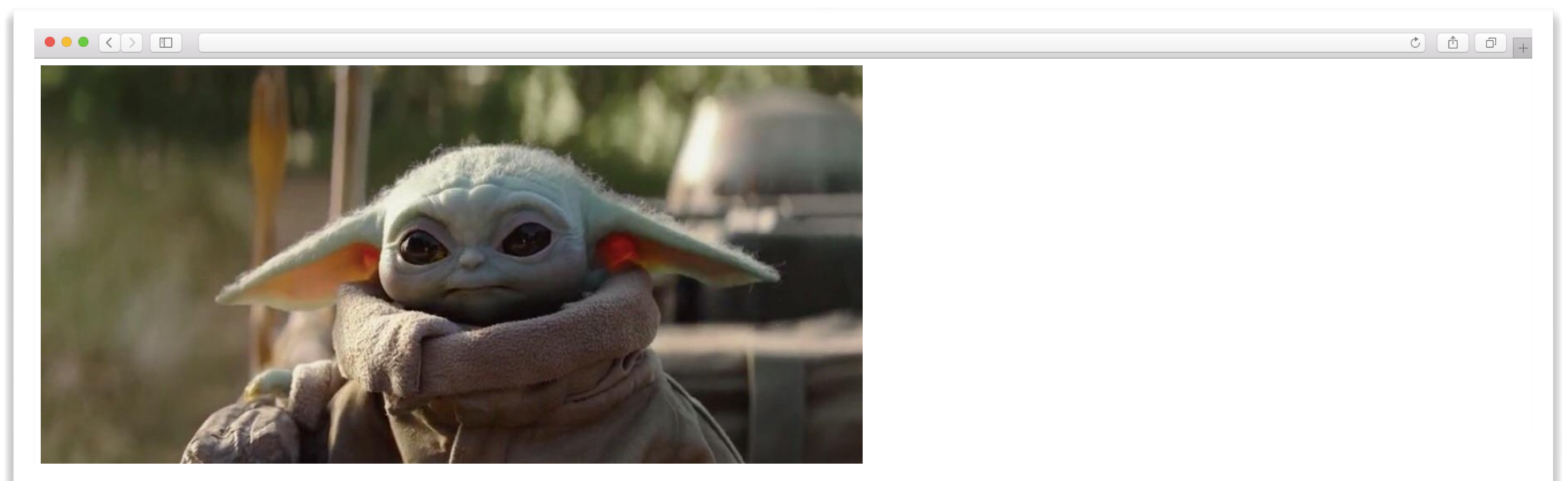
- Atributos fornecem metadados ou funcionalidade adicional
- Frequentemente utilizados como “âncoras” para CSS ou JS (e.g., `class`, `id`)
- No entanto, não determinam renderização!



```
<input type="checkbox" checked>
```



```
<a href="https://google.com">world</a>
```



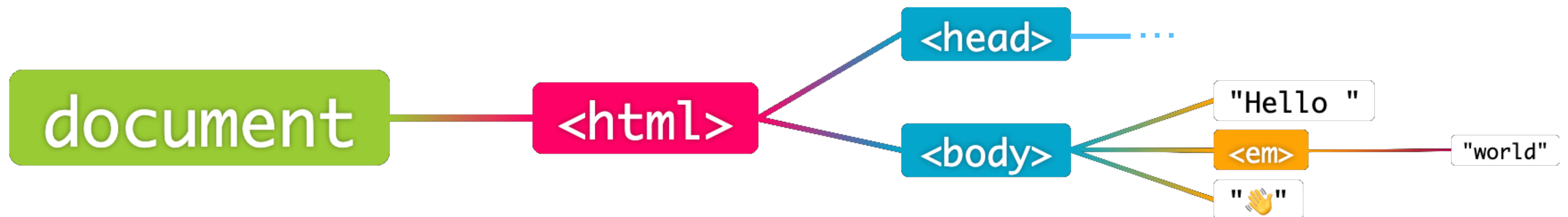
```

```

Document Object Model (DOM)

- Documento HTML = árvore de elementos HTML
- DOM = representação que o browser faz do documento HTML em memória

```
<!DOCTYPE html>  
<html>  
  <head>...</head>  
  <body>Hello <em>world</em> 🙌</body>  
</html>
```



HTML → DOM

- DOM representa uma hierarquia / árvore de elementos HTML
 - Relações pai-filho entre elementos
- *HTML parsing* = converter documento HTML numa árvore DOM
 - Um processo crucial para interoperabilidade HTML+CSS+JS+browsers
- ! Demorou cerca de 25 anos a estabilizar!
- Hierarquia HTML $\neq$ Hierarquia DOM?

```
<table>
```

```
  <tr><td>One</td></tr>
```

```
</table>
```



```
table
```

```
tbody
```

```
tr
```

```
td
```

```
One
```

HTML Parsing

- 💡 Uma das vantagens de uma linguagem declarativa é precisamente que é muito mais fácil ser-se **liberal em relação a erros**
 - Browser **nunca mostra erros** (excepto em modo de debugging)
 - Página carrega sempre, no limite mostrada como blank page
- + Parser HTML tenta **corrigir erros**
 - Tags que abrem mas não fecham, tags esperadas, tags inexistentes, etc
- + Uma forma de suportar **backward compatibility**
 - Self-closing tags (`<img />` = `<img>`), etc

Exercício: erros na Web

💡 Diferentes mecanismos de tratamento de erros

- **HTML** + **CSS** são **linguagens declarativas**
 - **HTML** tenta corrigir erros
 - **CSS** ignora erros
- **JS** é uma **linguagem imperativa**
 - **JS** pára (o script) e produz um erro (na consola)

1. Abrir página no browser. Render? Developer Tools?
2. **HTML**: apagar ou renomear tags?
3. **CSS**: criar propriedades inexistentes? apagar um ;?
4. **JS**: apagar “? Não fechar (? Dois scripts?

```
<style>
  h1 {
    color: slategray;
    color: yolo;
    font-style: red;
    font-weight: normal;
  }
</style>
<script>
  console.logg("Hello");
</script>
<h1 yolo="1">
  Error handling in <strong>HTML,
    <em>CSS</strong>, JS</em>
</h1>
```

Tags HTML5

- Root element**

- html

- Document metadata**

- head
- title
- base
- link
- meta

- Style**

- style

- Sections**

- body
- article
- section
- nav
- aside
- h1, h2, h3, h4, h5, h6
- hgroup
- header
- footer
- address

- Grouping content**

- p
- hr
- pre
- blockquote
- ol
- ul
- li
- dl

- dt
- dd
- figure
- figcaption
- main
- div

- Text-level semantics**

- a
- em
- strong
- small
- s
- cite
- q
- dfn
- abbr
- ruby
- rt
- rp
- data
- time
- code
- var
- samp
- kbd
- sub
- sup
- i
- b
- u
- mark

- bdi
- bdo
- span
- br
- wbr

- Edits**

- ins
- del

- Embedded content**

- picture
- source
- img
- iframe
- embed
- object
- param
- video
- audio
- source
- track
- map
- area

- Tabular data**

- table
- caption
- colgroup
- col
- tbody
- thead
- tfoot

- tr
- td
- th

- Forms**

- form
- label
- input
- button
- select
- datalist
- optgroup
- option
- textarea
- keygen
- output
- progress
- meter
- fieldset
- legend

- Interactive elements**

- details
- summary
- menu
- menuitem
- dialog

- Scripting**

- script
- noscript
- template
- canvas

HTML: Listas

- Existem três tipos de listas em HTML (cf. LaTeX):
 - Não ordenadas (`<ul>`): mostram os itens com marcadores ou outros símbolos
 - Ordenadas (`<ol>`): mostram os itens com números ou letras
 - Descrições (`<dl>`): mostram os itens com termos e definições
- Cada item de uma lista é marcado com
 - uma tag `<li>` (para listas ordenadas ou não ordenadas)
 - ou pares `<dt>` (título) e `<dd>` (descrição)

HTML: Listas

```
<ul>
  <li>All games</li>
  <li>My games</li>
  <li>Add games</li>
</ul>

<ol>
  <li>All games</li>
  <li>My games</li>
  <li>Add games</li>
</ol>

<dl>
  <dt>HTML</dt>
  <dd>HiperText Markup Language</dd>
  <dt>CSS</dt>
  <dd>Cascade Style Sheets</dd>
  <dt>Javascript</dt>
  <dd>Uma linguagem de scripting para desenvolvimento web</dd>
</dl>
```

- All games
- My games
- Add games

1. All games
2. My games
3. Add games

HTML

HiperText Markup Language

CSS

Cascade Style Sheets

Javascript

Uma linguagem de scripting para desenvolvimento web

HTML: Tabelas

- Tabela HTML (<table>)
 - Elemento <tr> para cada linha da tabela
 - Elementos <th> (cabeçalho) ou <td> (dados) para cada célula
- Atributos colspan e rowspan para fazer as células ocuparem várias colunas ou linhas
- Elemento <caption> para legendas
- Elementos <thead>, <tfoot> para cabeçalhos e rodapés
- Elemento <tbody> para o corpo é obrigatório (ou inferido)

HTML: Tabelas

```
<table>
  <caption>Frutas e cores</caption>
  <thead>
    <tr>
      <th>Fruta</th> <th>Cor</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Morango</td><td>Vermelha</td>
    </tr>
    <tr>
      <td>Banana</td><td>Amarela</td>
    </tr>
    <tr>
      <td>Laranja</td><td>Laranja</td>
    </tr>
  </tbody>
</table>
```

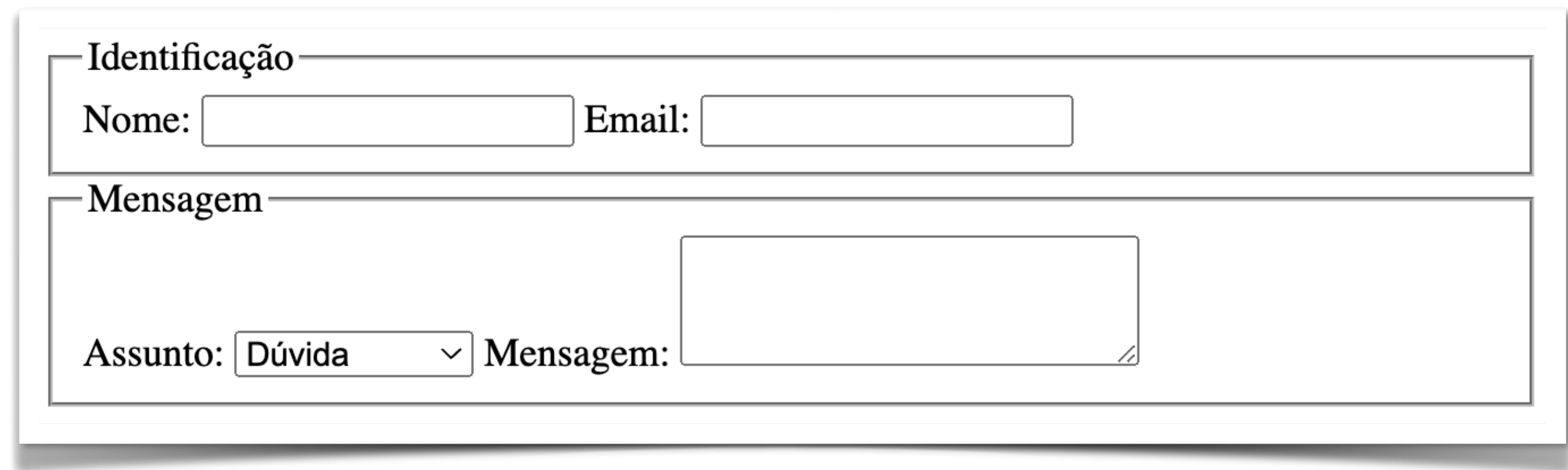
Fruta	Cor
Morango	Vermelha
Banana	Amarela
Laranja	Laranja

HTML: Formulários

- Formulário HTML (`<form>`)
 - Elementos de tipos diferentes
 - `<input>`, `<label>`, `<select>`, `<textarea>`, ...
 - Cada elemento de `input` tem atributos
 - `name` (nome usado para referência)
 - `value` (valor por defeito)
 - `type` (define o seu comportamento e aparência)
- Atributos `action` e `method` indicam ações (para onde dados são enviados) e como, respetivamente

HTML: Formulários

```
<form action="/enviar.jsp" method="POST">
  <fieldset>
    <legend>Identificação</legend>
    <label for="nome">Nome:</label>
    <input type="text" id="nome" name="nome" required>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
  </fieldset>
  <fieldset>
    <legend>Mensagem</legend>
    <label for="assunto">Assunto:</label>
    <select id="assunto" name="assunto">
      <option value="duvida">Dúvida</option>
      <option value="sugestao">Sugestão</option>
      <option value="reclamacao">Reclamação</option>
    </select>
    <label for="mensagem">Mensagem:</label>
    <textarea id="mensagem" name="mensagem" rows="3" cols="20"></textarea>
  </fieldset>
</form>
```



Identificação

Nome: Email:

Mensagem

Assunto: Mensagem:

```
<form method="get" action="https://www.google.com/search"
target="_blank">
  <input name="q" required />
  <button>Search Google</button>
</form>
```



HTML: Formulários

- Diferentes tipos de inputs

`<input type="button">`

`<input type="checkbox">`

`<input type="color">`

`<input type="date">`

`<input type="datetime-local">`

`<input type="email">`

`<input type="file">`

`<input type="hidden">`

`<input type="image">`

`<input type="month">`

`<input type="number">`

`<input type="password">`

`<input type="radio">`

`<input type="range">`

`<input type="reset">`

`<input type="search">`

`<input type="submit">`

`<input type="tel">`

`<input type="text">`

`<input type="time">`

`<input type="url">`

`<input type="week">`

The image displays a variety of HTML form input types arranged in a grid-like fashion. At the top left, there are three checkboxes: an unchecked one, a checked one, and a disabled one. Next to them is a color picker showing a black square. Below these are several date and time pickers: a date picker with the format 'dd/mm/yyyy', a datetime-local picker with the format 'dd/mm/yyyy, --:--', a month picker with dashes, and a week picker with the format 'Week --, ----'. There is also a file upload control with a 'Choose file' button and the text 'No file chosen'. A range slider is shown with a blue track and a blue knob. Several buttons are present, including a 'Submit' button with a green icon and a 'Reset' button. There are also several empty text input fields of different sizes.

HTML: Formulários

- Declarativo (HTML)

```
<form>
  <label for="user">Username:</label>
  <input type="text" name="user" />
  <input type="submit" value="Submit" />
</form>
```

A screenshot of a web form rendered from the provided HTML code. It features a light gray background. On the left, the text "Username:" is followed by a white text input field with a thin gray border. To the right of the input field is a rounded rectangular button with a light gray background and the word "Submit" in a dark gray font.

- 💡 Imperativo inclui layout
- 💡 Podíamos ter repetido o exercício em JS (ainda mais complexo)

- Imperativo (Java)

```
JFrame frame = new JFrame("Form Example");
JPanel panel = new JPanel();
JLabel label = new JLabel("Username:");
JTextField textField = new JTextField(10);
JButton button = new JButton("Submit");

button.addActionListener(e -> {
    String value = textField.getText();
    leNome(value);
});

panel.add(label);
panel.add(textField);
panel.add(button);

frame.add(panel);
frame.pack();
frame.setVisible(true);
```

A screenshot of a Java Swing window titled "Form Example". The window has a white background. It contains a label "Username:" in a blue font, followed by a white text input field with a thin gray border. To the right of the input field is a rounded rectangular button with a light gray background and the word "Submit" in a dark gray font.

Markup semântico

? **Puzzle:** cabeçalho vs texto normal

```
<h1>This is a top level heading</h1>  
<span style="font-size: 2em; margin: 0.67em  
0; font-weight: bold;">  
  Is this a top level heading?  
</span>
```

This is a top level heading
Is this a top level heading?

💡 Mesma representação visual, significado diferente

? Porque é que a semântica é relevante?

- Nem todos os “web surfers” vêem bem $\Rightarrow$ screen readers lêem páginas para invisuais
- Nem todos os “web surfers” usam rato $\Rightarrow$ navegação com o teclado depende da estrutura
- Compatibilidade $\Rightarrow$ rendering pode mudar com browsers e versões

Navegação com o teclado

- HTML tem uma noção de elemento atualmente focado
 - Clickar em elementos com o rato altera foco
 - Tab/Space navegam pela página, alterando foco
- Nem todos os elementos são navegáveis
 - <https://allyjs.io/data-tables/focusable.html>
- Exemplos:
 - `<input type="text" autofocus>`
 - https://www.google.com/advanced_search



Create account

Your name

Email

Password

Password again

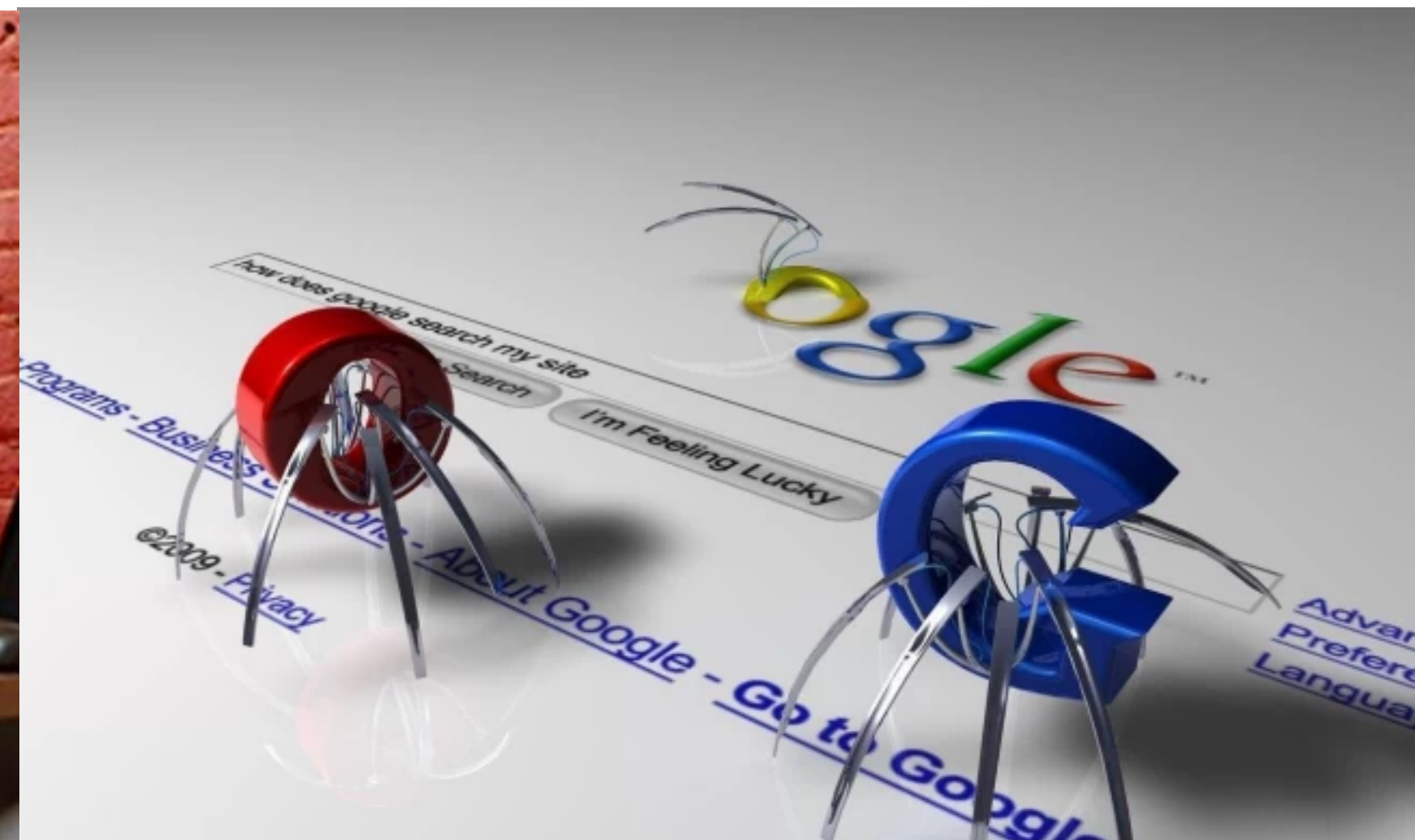
Create your Amazon account

By creating an account, you agree to Amazon's [Conditions of Use](#) and [Privacy Notice](#).

Already have an account? [Sign in](#)

Navegação alternativa

- Nem todos os “web surfers” são humanos
 - Screen readers
 - Web crawlers
 - Web scrapers
- Nem todos os browsers têm interfaces gráficas



```
Your continued donations keep Wikipedia running!  
Lynx (web browser)  
From Wikipedia, the free encyclopedia  
Jump to: navigation, search  
CAPTION: Lynx  
Wikipedia Main Page displayed in Lynx
```

Hierarquia HTML $\neq$ Hierarquia Conteúdo?

- E.g., podemos construir sites inteiros só com tabelas

💡 Algo muito comum antes de **CSS** (**flex** ou **grid** layouts) ser tão poderoso

+ Posicionamento mais previsível

- Hierarquia flat

- Estrutura afetada por visualização

```
<form method="post" action="/login">
  <table>
    <tr>
      <td>Username:</td>
      <td><input type="text"></td>
    </tr>
    <tr>
      <td>Password:</td>
      <td><input type="password"></td>
    </tr>
  </table>
</form>
```



Visual representation of the HTML form structure shown in the code block above. It consists of a light gray rounded rectangle containing two rows. The first row has the text 'Username:' followed by a white rectangular input field with a thin gray border. The second row has the text 'Password:' followed by a white rectangular input field with a thin gray border.

Markup semântico

- 💡 Tags **HTML** têm significado (excepto `<div>` e `<span>`)
- 💡 Utilização de tags para definir o seu conteúdo, não a sua visualização
- + Facilita manutenção do código
- + Facilita o acesso (e.g., utilizadores com deficiência, indexação por motores de busca), independentemente de **CSS** e **JS**
- + Facilita formatação com **CSS** e interação com **JS**
- + Facilita adaptação a diferentes media / dispositivos
- 💡 **Uma forma de usabilidade:** nem sempre trivial, mas esforço compensa

Exemplo: identificar markup?

<header>

Title

App name

User name

All gamesMy gamesAdd game

Games	Year	Platform
Game 1	1992	Megadrive
Game 2	2017	PC

Filter

year

Platform

123

<nav>

<aside>

<main>

<footer>

Exemplo: identificar markup?

amazon

Try Prime

All

CSS

ENHello, Sign inAccount & ListsReturns & OrdersTry PrimeCart

1-48 of 323 results for 4 Stars & Up : Get it by Tomorrow : "css"

Sort by: Featured

Amazon Prime

☐prime

Delivery Day

< Clear

☒Get it by Tomorrow

Eligible for Free Shipping

☐Free Shipping by Amazon

All customers get FREE Shipping on orders over \$25 shipped by Amazon

Department

Books

Avg. Customer Review

< Clear

★★★★★

& Up

★★★★☆

& Up

★★★☆☆

& Up

★★☆☆☆

& Up

★☆☆☆☆

& Up

Amazon Global Store

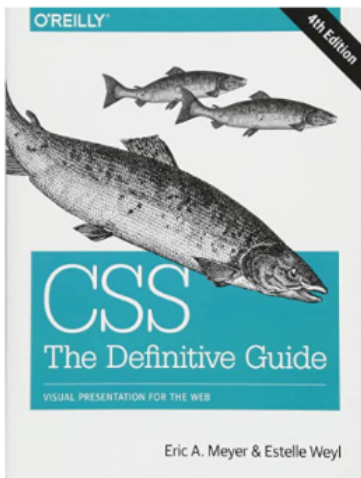
☐Amazon Global Store

International Shipping

☐International Shipping Eligible

Condition

New



CSS: The Definitive Guide: Visual Presentation for the Web

by Eric A. Meyer and Estelle Weyl

★★★★★ < 28

Paperback

\$56⁶⁶ ~~\$69.99~~

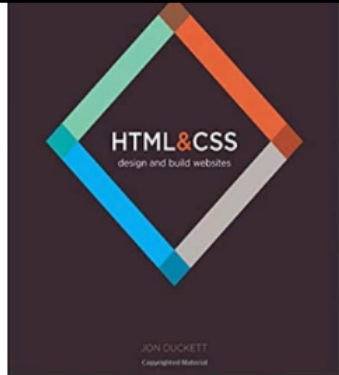
☒prime Get it as soon as Tomorrow, Feb 6

FREE Shipping by Amazon

More Buying Choices

\$43.98 (58 used & new offers)

Other format: [Kindle](#)



HTML and CSS: Design and Build Websites

by Jon Duckett

★★★★★ < 1,245

Paperback

\$17⁵⁸ ~~\$29.99~~

☒prime Get it as soon as Sat, Feb 8

FREE Shipping on orders over \$25 shipped by Amazon

More Buying Choices

\$5.24 (115 used & new offers)

Other format: [Hardcover](#)



Web Design with HTML, CSS, JavaScript and jQuery Set

by Jon Duckett

★★★★★ < 677

Paperback

\$26⁹⁹ ~~\$58.00~~

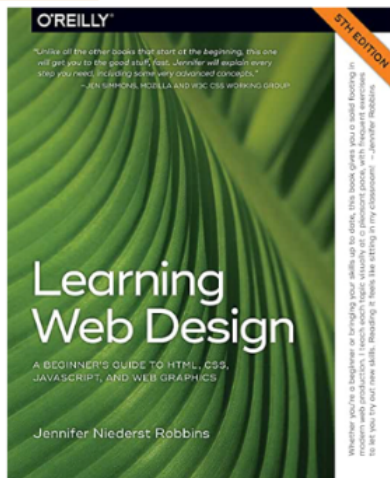
☒prime Get it as soon as Tomorrow, Feb 6

FREE Shipping by Amazon

More Buying Choices

\$19.50 (29 used & new offers)

Other format: [Hardcover](#)



Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics

by Jennifer Robbins

★★★★★ < 45

Paperback

\$46⁹⁹ ~~\$59.99~~

☒prime Get it as soon as Tomorrow, Feb 6

FREE Shipping by Amazon

More Buying Choices

\$33.81 (51 used & new offers)

Other format: [Kindle](#)



CSS Secrets: Better Solutions to Everyday Web Design Problems

by Lea Verou

★★★★★ < 97

Paperback

\$27²¹ ~~\$44.99~~

☒prime Get it as soon as Tomorrow, Feb 6

FREE Shipping by Amazon

More Buying Choices

\$12.70 (70 used & new offers)

Other format: [Kindle](#)



CSS: The Missing Manual (85 Books)

by David Sawyer McFarland

★★★★★ < 55

Paperback

\$29¹⁶ ~~\$49.99~~

☒prime Get it as soon as Tomorrow, Feb 6

FREE Shipping by Amazon

Only 10 left in stock (more on the way).

More Buying Choices

\$9.99 (48 used & new offers)

Other format: [Kindle](#)



Microsoft Surface

Intel CORE i7 10TH GEN

New Surface Pro 7

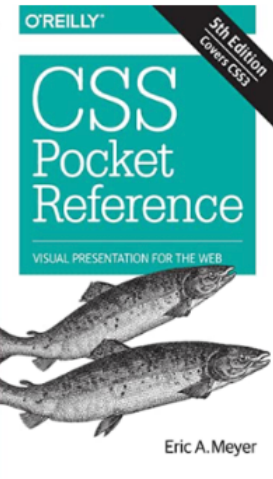
with the latest 10th Gen Intel Core processor



Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages

by Elisabeth Robson and Eric Freeman

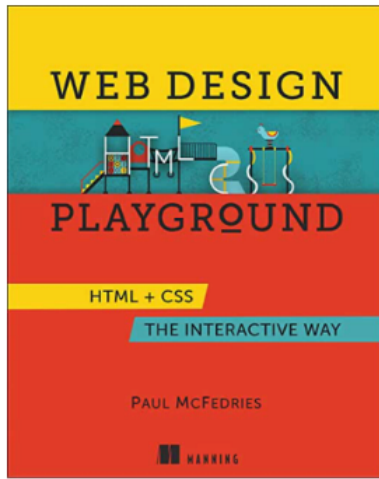
★★★★★ < 17



CSS Pocket Reference: Visual Presentation for the Web

by Eric A. Meyer

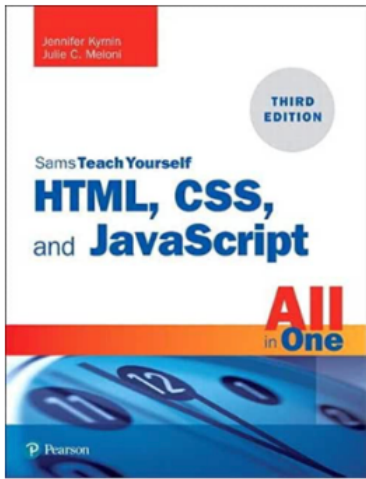
★★★★★ < 17



Web Design Playground: HTML & CSS the Interactive Way

by Paul McFedries

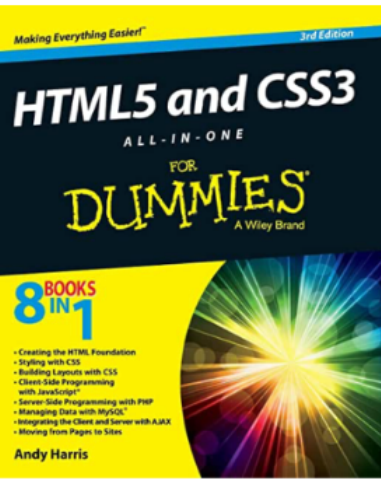
★★★★★ < 10



HTML, CSS, and JavaScript All in One, Sams Teach Yourself (3rd Edition)

by Julie C. Meloni and Jennifer Kyrnin

★★★★★ < 103



HTML5 and CSS3 All-in-One For Dummies

by Andy Harris

★★★★★ < 103

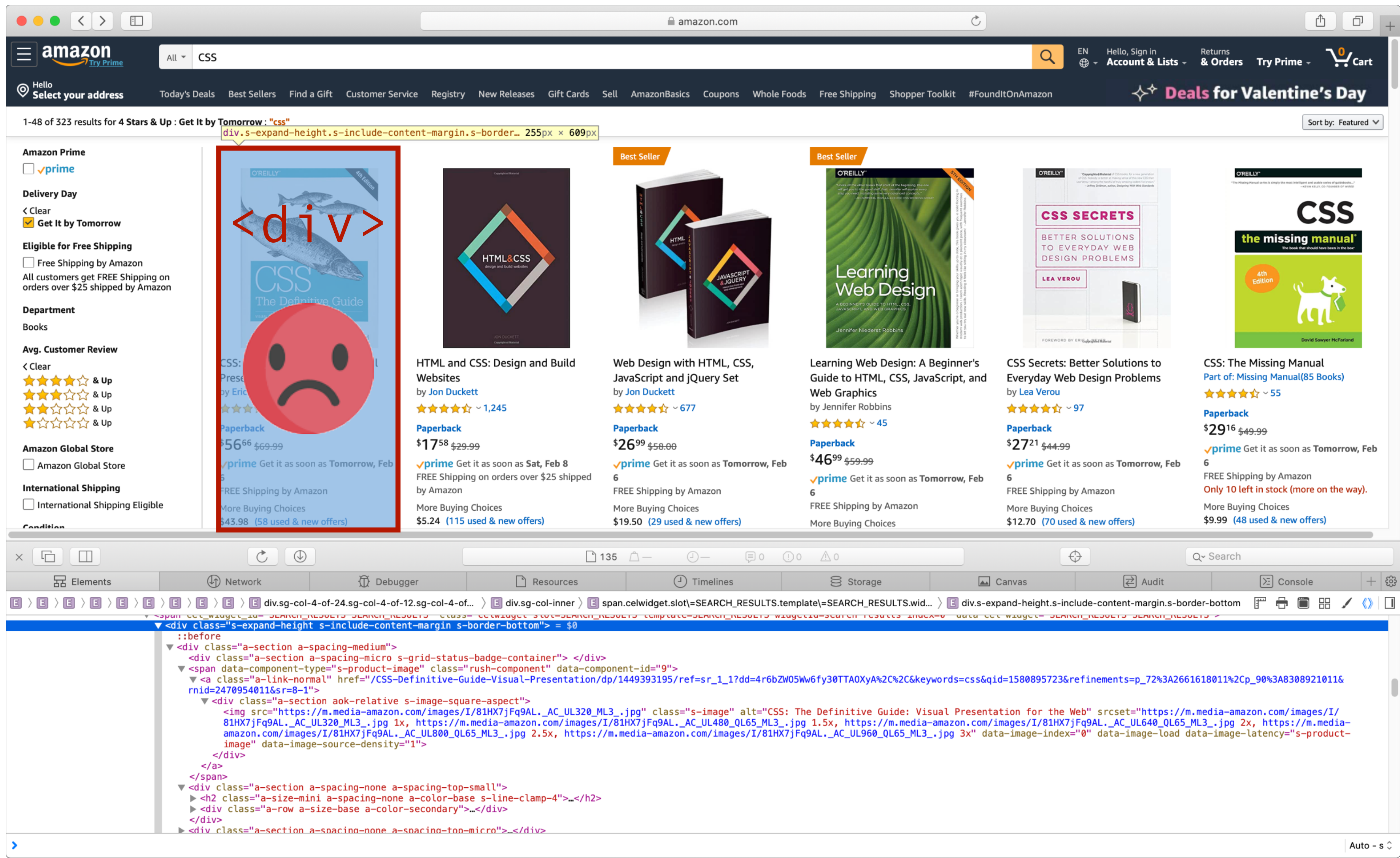


HTML and CSS: Visual QuickStart Guide (8th Edition)

by Elizabeth Castro and Bruce Hyslop

★★★★★ < 77

Exemplo: identificar markup?



Exemplo: article HTML element

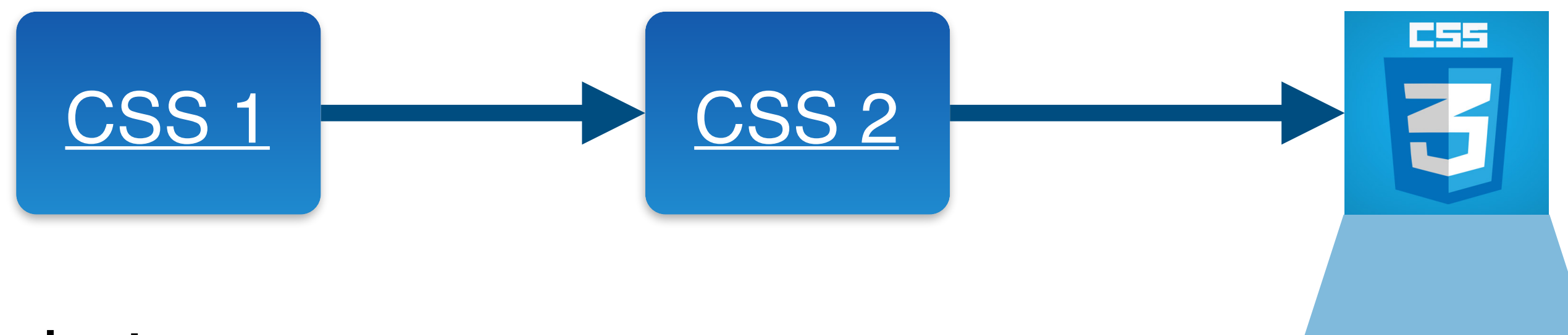
- “represents a **self-contained composition** in a document, page, application, or site, which is intended to be independently distributable or reusable”

HTML	CSS	OUTPUT
<pre>1 <article class="forecast"> 2 <h1>Weather forecast for Seattle</h1> 3 <article class="day-forecast"> 4 <h2>03 March 2018</h2> 5 <p>Rain.</p> 6 </article> 7 <article class="day-forecast"> 8 <h2>04 March 2018</h2> 9 <p>Periods of rain.</p> 10 </article> 11 <article class="day-forecast"> 12 <h2>05 March 2018</h2> 13 <p>Heavy rain.</p> 14 </article> 15 </article></pre>		Weather forecast for Seattle 03 March 2018 Rain.  04 March 2018 Periods of rain.  05 March 2018 Heavy rain. 

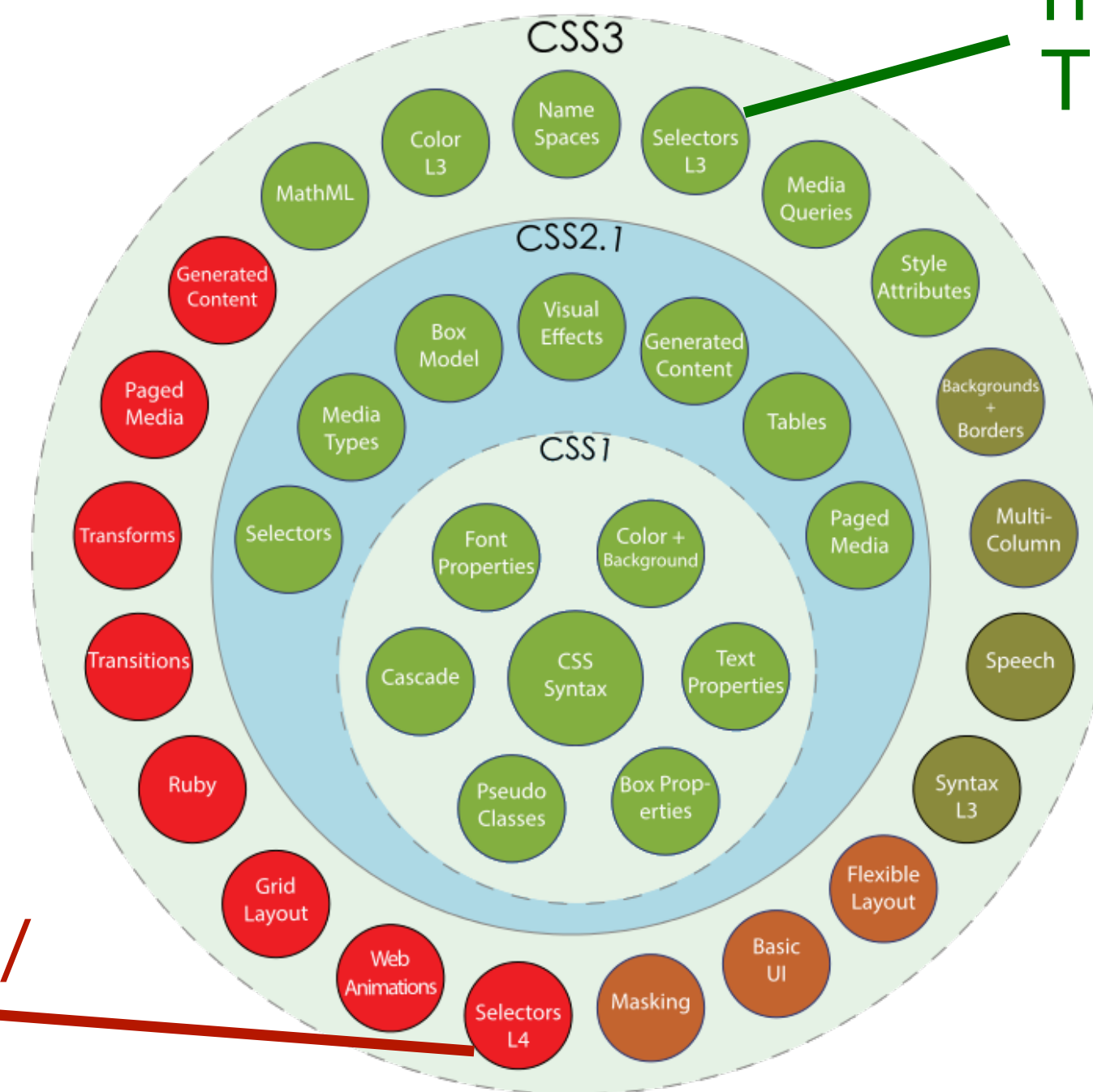
CSS

CSS

- **C**ascading **S**tyle **S**heets (CSS) é uma styling language para HTML



- CSS3 deixou de ter versões (níveis W3C)
- Cada módulo CSS3 evolui de forma independente

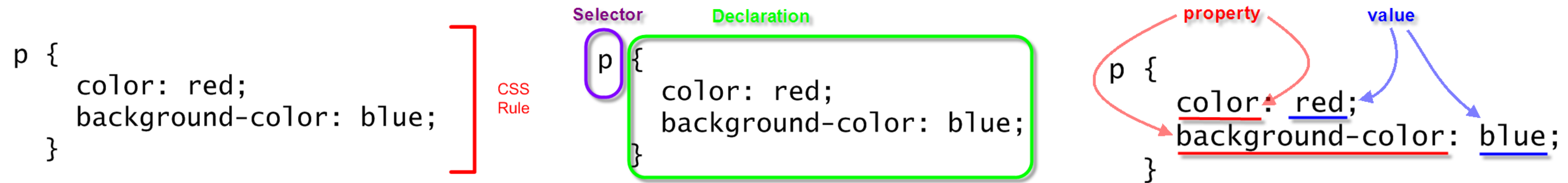


<https://www.w3.org/TR/selectors-3/>

<https://www.w3.org/TR/selectors-4/>

Estilos CSS

- Estilos (styles) definem formatação dos elementos HTML
- Anatomia de uma CSS rule



- **Programação declarativa**
 - O quê? CSS descreve um conjunto de regras independentes
 - Como? Browser trata de aplicar as regras
- CSS Snapshot (18/09/2025)

Cascading Style Sheets

- Documento HTML pode ter várias **Style Sheets**

- Externamente num ficheiro `.css`

```
<link rel="stylesheet" type="text/css" href="mystyle.css">
```

- Internamente ao documento (definida no `<header>` ou `<head>`)

```
<style> td {font-size: 0.8em;} </style>
```

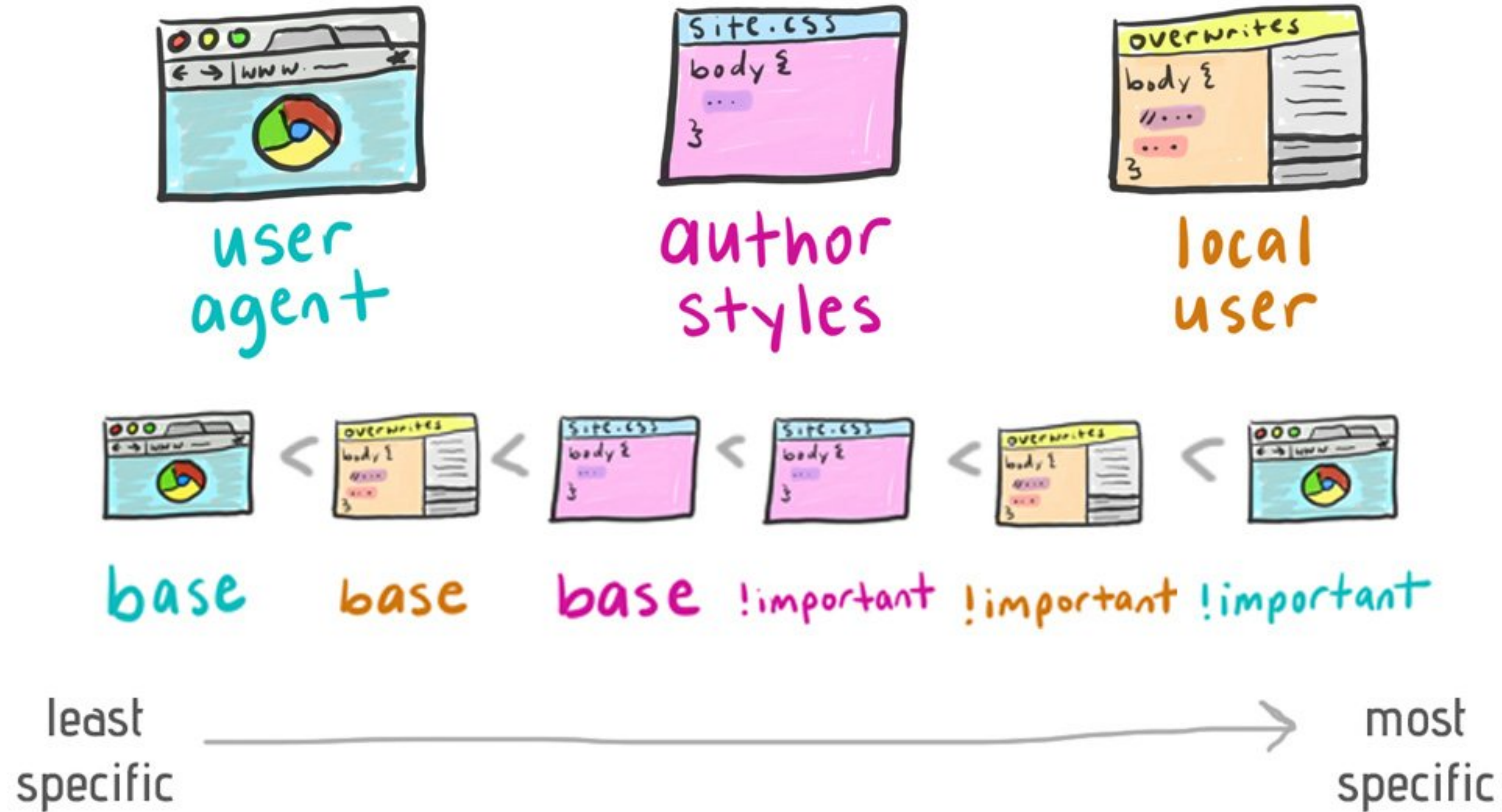
- Inline (para um elemento HTML)

```
<dt style="margin-bottom:20px;">...</dt>
```

Cascading Style Sheets

- A **Cascade** é o algoritmo que corre no browser e define quais estilos **CSS** (de diversas fontes potencialmente conflitantes) se aplicam a cada elemento **HTML**
- Heurística de prioridade com base em diferentes fatores:
 - Origem:
 1. User
 2. Author (inline > interno > externo)
 3. User Agent (style sheet por defeito do browser, pode variar)
 - Especificidade:
`p[class="primary"]` > `p` > `*`

Cascading Style Sheets



<3 @una

Exemplo (HTML)

```
<!DOCTYPE html>
<html>
  <head>
    <title>My first styled page</title>
    <!--link rel="stylesheet" href="mystyle.css"-->
  </head>
  <body>
    <nav>
      <ul class="navbar">
        <li><a href="index.html">Home page</a>
        <li><a href="musings.html">Musings</a>
        <li><a href="town.html">My town</a>
        <li><a href="links.html">Links</a>
      </ul>
    </nav>
    <main>
      <h1>My first styled page</h1>
      <p>Welcome to my styled page!</p>
      <p>It lacks images, but at least it has style. And it has links, even if they don't go
anywhere&hellip;</p>
      <p>There should be more here, but I don't know
what yet.</p>
    </main>
    <address>Made 5 April 2004 by myself.</address>
  </body>
</html>
```

- [Home page](#)
- [Musings](#)
- [My town](#)
- [Links](#)

My first styled page

Welcome to my styled page!

It lacks images, but at least it has style. And it has links, even if they don't go anywhere...

There should be more here, but I don't know what yet.

Made 5 April 2004 by myself.

Exemplo (HTML + CSS)

```
body {
  font-family: Georgia, "Times New Roman", Times, serif;
  color: purple;
  background-color: #d8da3d }
h1 {
  font-family: Helvetica, Geneva, Arial,
  :::: SunSans-Regular, sans-serif }

nav {
  padding: 0;
  margin: 0;
  width: 10em }
nav > ul a {
  text-decoration: none }
a:link {
  color: blue }
a:visited {
  color: purple }
address {
  margin-top: 1em;
  padding-top: 1em;
  border-top: thin dotted }
:::
body { transition: padding-left 1s linear 0.1s; }

ul.navbar { transition: width 1s linear 0.1s; }
```

- [Home page](#)
- [Musings](#)
- [My town](#)
- [Links](#)

My first styled page

Welcome to my styled page!

It lacks images, but at least it has style. And it has links, even if they don't go anywhere...

There should be more here, but I don't know what yet.

Made 5 April 2004 by myself.

Exemplo (HTML + CSS)

```
@media (min-width: 400px) {  
  body { padding-left: 12em;}  
  nav {  
    position: absolute;  
    top: 2em;  
    left: 1em;}  
  nav > ul {  
    list-style-type: none; }  
  nav > ul > li {  
    background: white;  
    margin: 0.5em 0;  
    padding: 0.3em;  
    border-right: 1em solid black }  
}
```



CSS Zen Garden



A demonstration of what can be accomplished through CSS-based design. Select any style sheet from the list to load it into this page.

Download the example [HTML FILE](#) and [CSS FILE](#)

THE ROAD TO ENLIGHTENMENT

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, broken CSS support, and abandoned browsers.

We must clear the mind of the past. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WASP, and the major browser creators.

The CSS Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the time-honored techniques

MID CENTURY
MODERN
by Andrew Lohman

GARMENTS
by Dan Mall

STEEL
by Steffen Knoeller

APOTHECARY
by Trent Walton



CSS: Simple Selectors

- Definem o conjunto de elementos HTML aos quais uma regra se aplica

- `*` (qualquer elemento)

`* {...}`

- `element` (com tag `element`)

`h1 {...}`

- `.class` (com classe `class`)

`<p class="right">txt</p>`

`.right {...}`

- `#id` (com identificador `id`)

`<h1 id="chap1">txt</h1>`

`#chap1 {...}`

CSS: Simple Selectors (atributos)

- Selecionar elementos com base em atributos gerais
- `[attr]` (contém atributo `attr`)
- `[attr="value"]` (contém atributo `attr` com valor `value`)
- `[attr^="starts"]` (contém atributo `attr` com valor iniciado por `starts`)
- `[attr$="ends"]` (contém atributo `attr` com valor acabado em `ends`)
- `[attr*="anywhere"]` (contém atributo `attr` com substring `anywhere`)
- `[attr~="anywhere"]` (contém atributo `attr` com palavra `anywhere`)

```
<div class="box highlight big">
```

```
[class~="highlight"] {...} ≡ .highlight {...}
```

CSS: Compound Selectors

- Concatenação de selectors $\triangleq$ interseção de conjuntos

```
<div class="foo bar baz"></div>
```

```
.foo.bar {...}
```

```
 $\equiv$  [class~="foo"][class~="bar"] {...}
```

```
 $\equiv$  .foo and .bar {...}
```

```
<p class="center">Texto</p>
```

```
p.center {...}
```

```

```

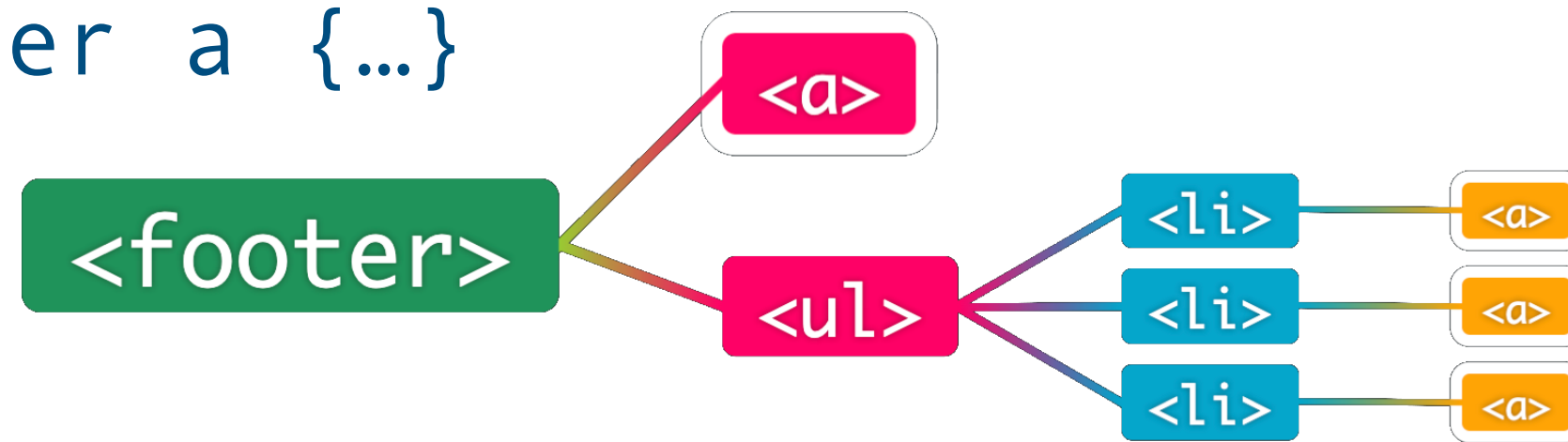
```
img[loading=lazy] {...}
```

CSS: Complex Selectors

- Múltiplos compound selectors separados por combinadores $\triangleq$ hierarquia de elementos

- Descendente

footer a {...}



- Próximo parente

h1 + h2 {...}

<h1>...</h1>

<h2>...</h2>

- Filho direto

ol.tasks > li {...}



- Parente subsequente

h1 ~ h2 {...}

<h1>...</h1>

<p>

<h2>...</h2>

CSS: Selector Lists

- Múltiplos complex selectors separados por vírgulas $\triangleq$ união de conjuntos

```
ul.bar li, .foo {...}
```

```
<ul class="bar">
```

```
  <li>Matches (li inside ul.bar)</li>
```

```
  <li class="foo">Matches twice (li inside ul.bar, and also .foo)</li>
```

```
</ul>
```

```
h1, h2 {...}
```

```
<div>
```

```
  <h1>Matches (h1)</h1>
```

```
  <h2>Matches (h2)</h2>
```

```
</div>
```

CSS: Pseudo-classes

- Identificar elementos com base nas suas características / estado; inspirado em JS
- Dinâmicas (activity-based)
 - Mouse over element :`hover`
 - Mouse down on element :`active`
 - Currently focused :`focus`
 - Contains currently focused :`focus-within`
 - Checked radio or checkbox :`checked`
 - Visited URL :`visited`
 - Non-visited URL :`link`

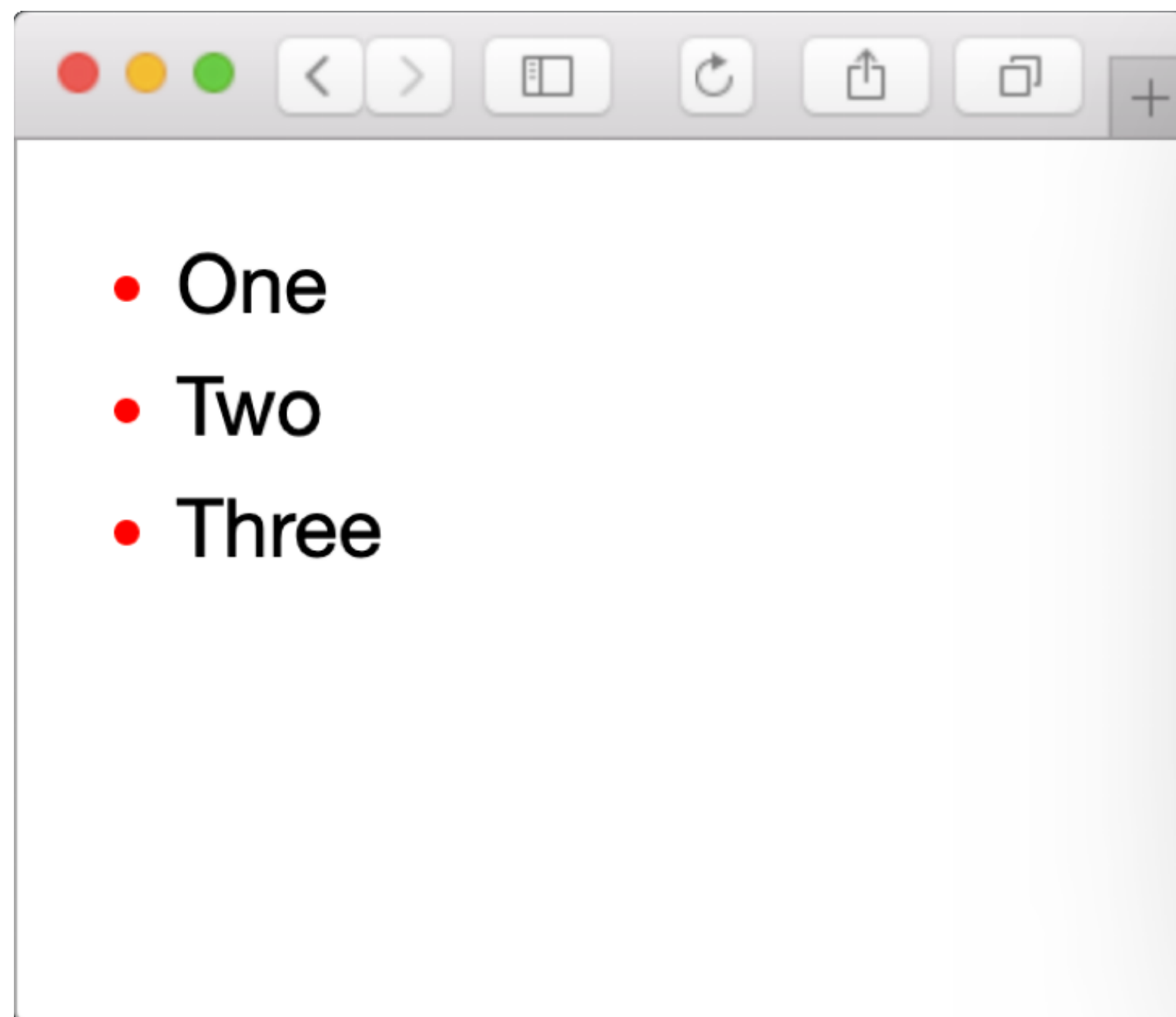
CSS: Pseudo-classes

- Estruturais (DOM-based)
 - Primeiro filho de elemento `:first-child`
 - Último filho de elemento `:last-child`
 - Único filho de elemento `:only-child`
 - Filhos em posição ímpar `:nth-child(odd)`
 - Filhos cujo índice `n` está no intervalo `[0..-n+3[` `:nth-child(-n + 3)`
 - Elemento que tem um descendente com classe `child` `:has(.child)`
- Negação `:not(:hover)`

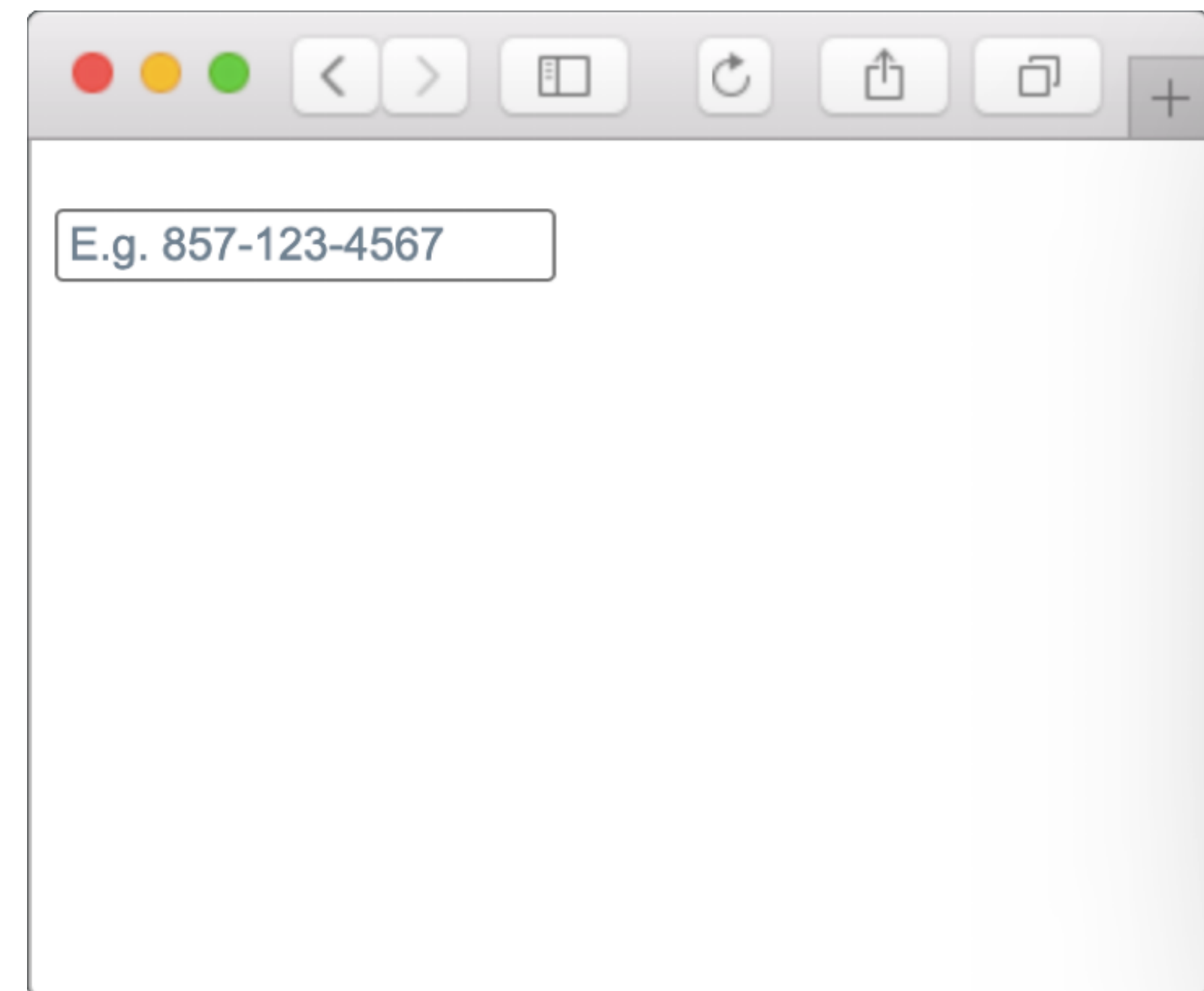
CSS: Pseudo-elements

- Identificar partes de um elemento

- ```
li::marker {
 color:red;
}
```



- ```
input::placeholder {  
  color:slategrey;  
}
```

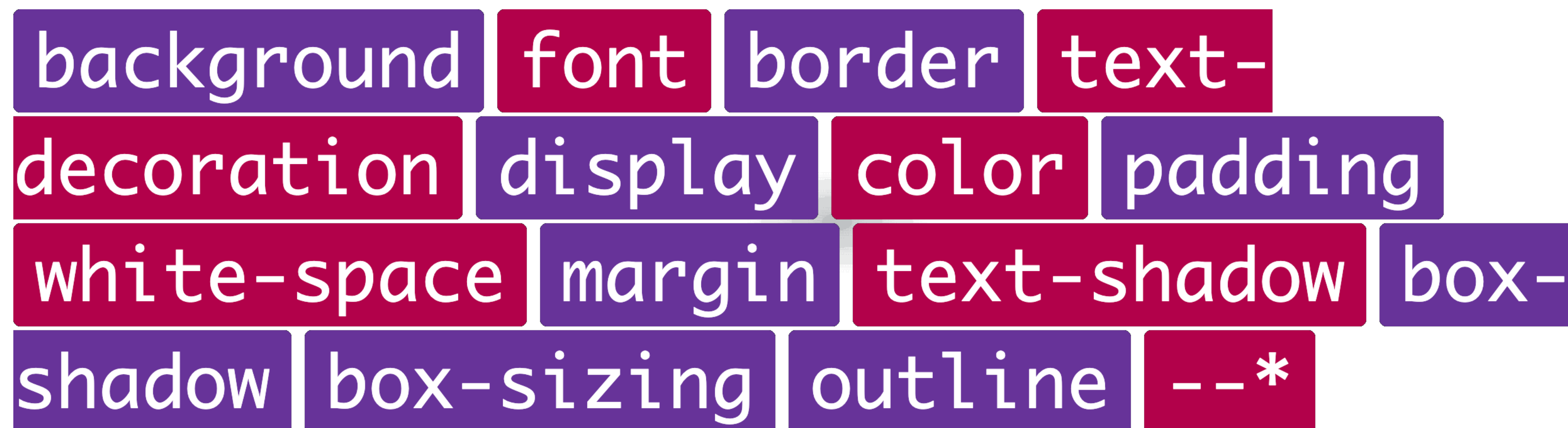


CSS: Propriedades

- Estilo
 - Cores
 - Fontes
 - Espaçamento
 - Transformações
 - Transições e animações
 - ...
- Layout: posicionar os elementos na página

CSS: Propriedades (Inherited)

- Algumas propriedades são herdadas (passadas implicitamente de pais para filhos)
- Outras não (definidas só para o elemento atual)

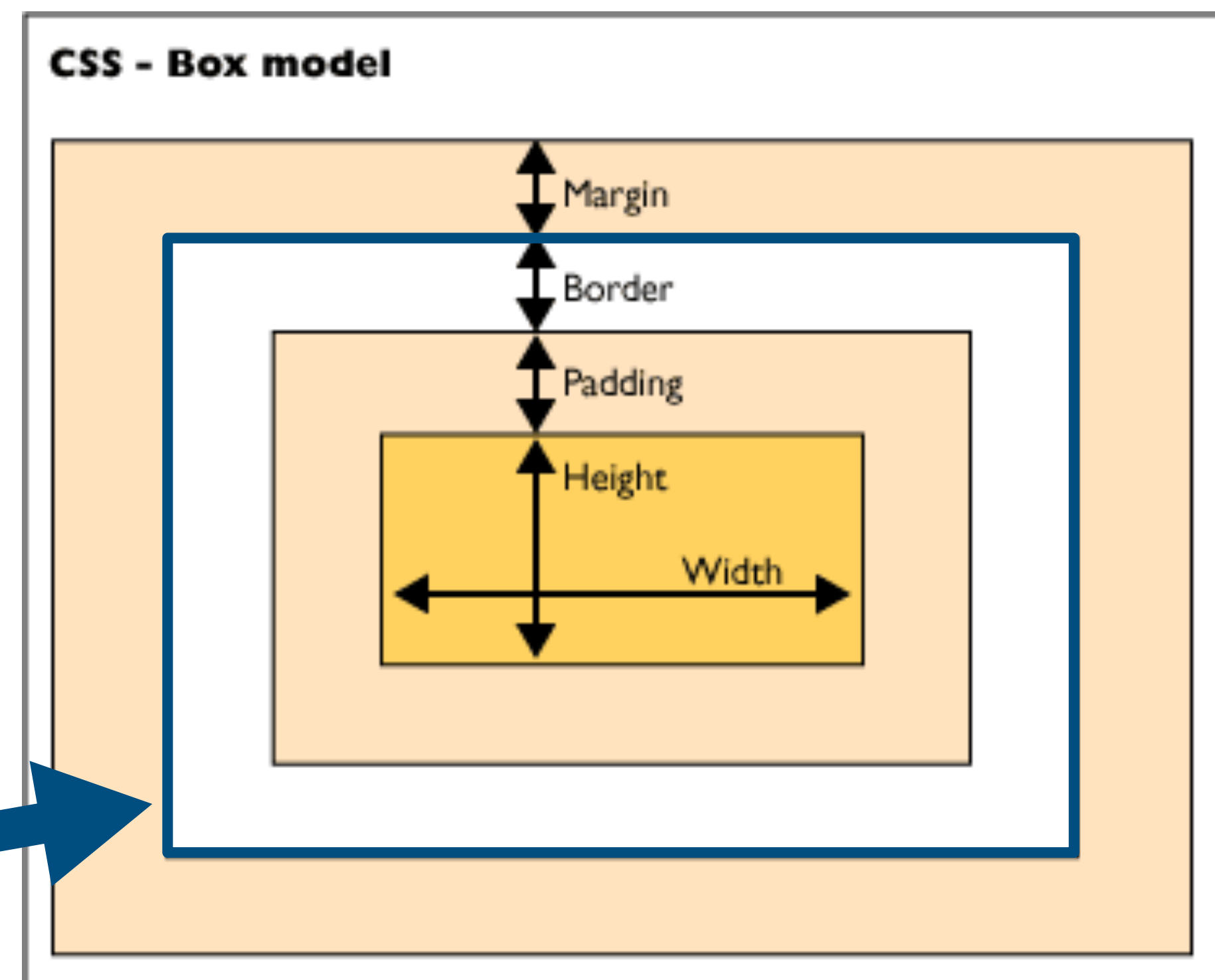


Herdadas Não herdadas



CSS: Propriedades (Layout)

- Todos os elementos são uma “box”
- Propriedades determinam como “boxes” são desenhadas e se relacionam com outras “boxes”
 - `width/height` do conteúdo
 - `property` ou `property-X`
 - $X = \{ \text{top, bottom, left, right} \}$
 - Largura efetiva do elemento
 - `width + padding + border + margin`
 - `* {box-sizing: border-box; }`

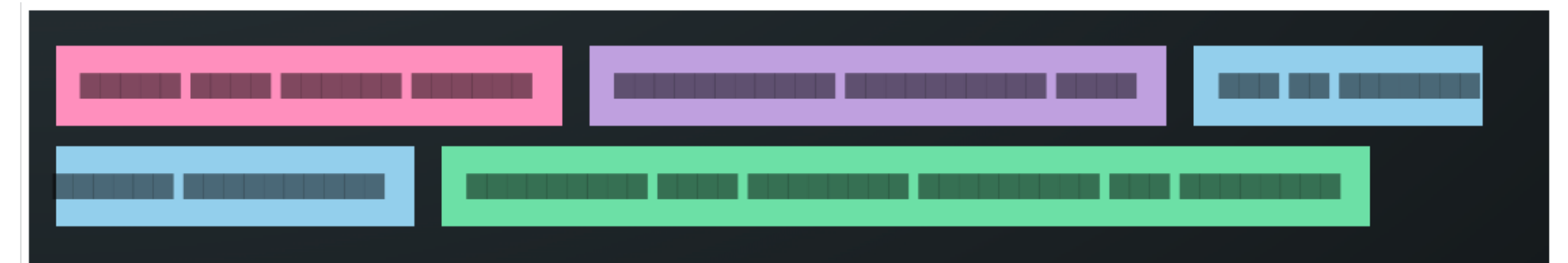


CSS: Propriedades (Tamanhos)

- CSS suporta várias unidades de tamanho
 - Absolute (mesmo tamanho em qualquer contexto)
 - Exemplos: `px`, `cm`, `mm`, `in`, `pt`
 - Relative (dependem de outros fatores)
 - Font-relative (dependem do tamanho do texto)
 - Exemplos: `em`, `ch`, `rem`, `ex`
 - Viewport-relative (dependem da janela)
 - Exemplos: `vh`, `vw`, `vmín`, `vmax`
 - Property-specific (comportamento variável, nem sempre aceite)
 - Percentagens, e.g., `50%vh`
 - Keyword `auto`

CSS: Propriedades (`display`)

- `display: block;`
 - Laid out from top to bottom
 - Line break before and after
 - Box as wide as all available space
 - Box as tall as all contents (wrapped)
 - Text wrapping inside box
 - We can set `width`, `height`, `margin`
- `display: inline;`
 - Laid out from left to right
 - Inline with surrounding text
 - Box as wide as all contents
 - Box as tall as one line
 - Text wrapping by box fragmentation
 - `width`, `height`, `margin` have no effect.



CSS: Propriedades (`display`)

- Outras opções de `display`
 - `inline-block`
 - Comporta-se essencialmente como um `inline`, mas com wrapping de `block`
 - We can set `width`, `height`, `margin-top`, `margin-bottom`
 - `none`
 - Não é mostrado
 - `contents`
 - Apenas são mostrados os seus conteúdos

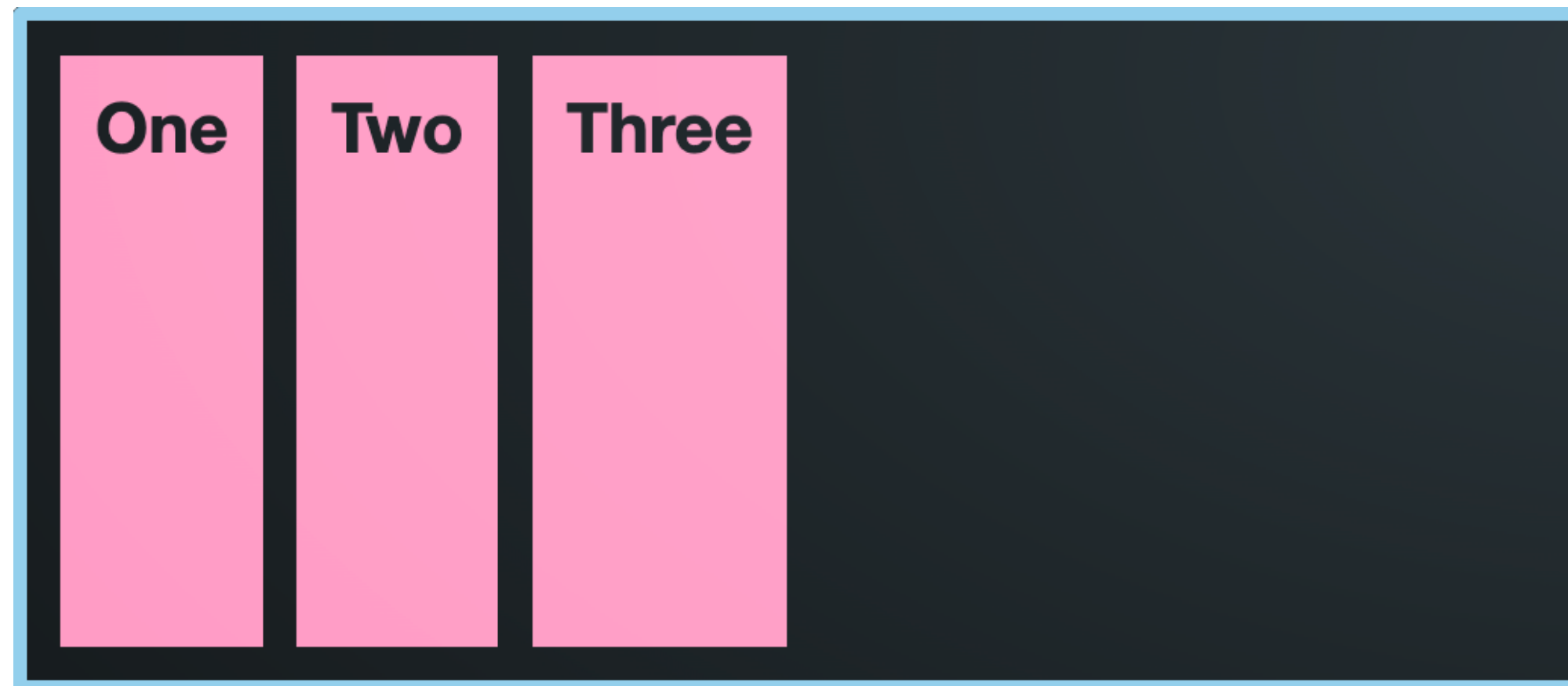
CSS: Propriedades (display)

- Por defeito (depende de User Agent Stylesheet)
 - `block`
 - `<div>` `<h1>` `<h2>` `<h3>` `<h4>` `<h5>` `<h6>` `<p>` `<body>` `<html>`
 - `inline`
 - `<a>` `<span>` `<strong>` `<em>` `<mark>` `<code>` `<cite>` `<abbr>`
 - `inline-block`
 - `<button>` `<input>` `<textarea>` `<select>`
- Replaced elements (não controlados por CSS)
 - Flutuam como um `inline-block`, mas conteúdos não são afetados por CSS
 - `<img>` `<video>` `<audio>` `<iframe>` `<canvas>` `<option>` `<object>` `<embed>`

CSS: Propriedades (`display: flex`)

- Distribuir items proporcionalmente, cálculo de margens automáticas

- `flex-direction: row;`



- `align-items`: alinhamento ao longo do eixo principal

- `flex-direction: column;`

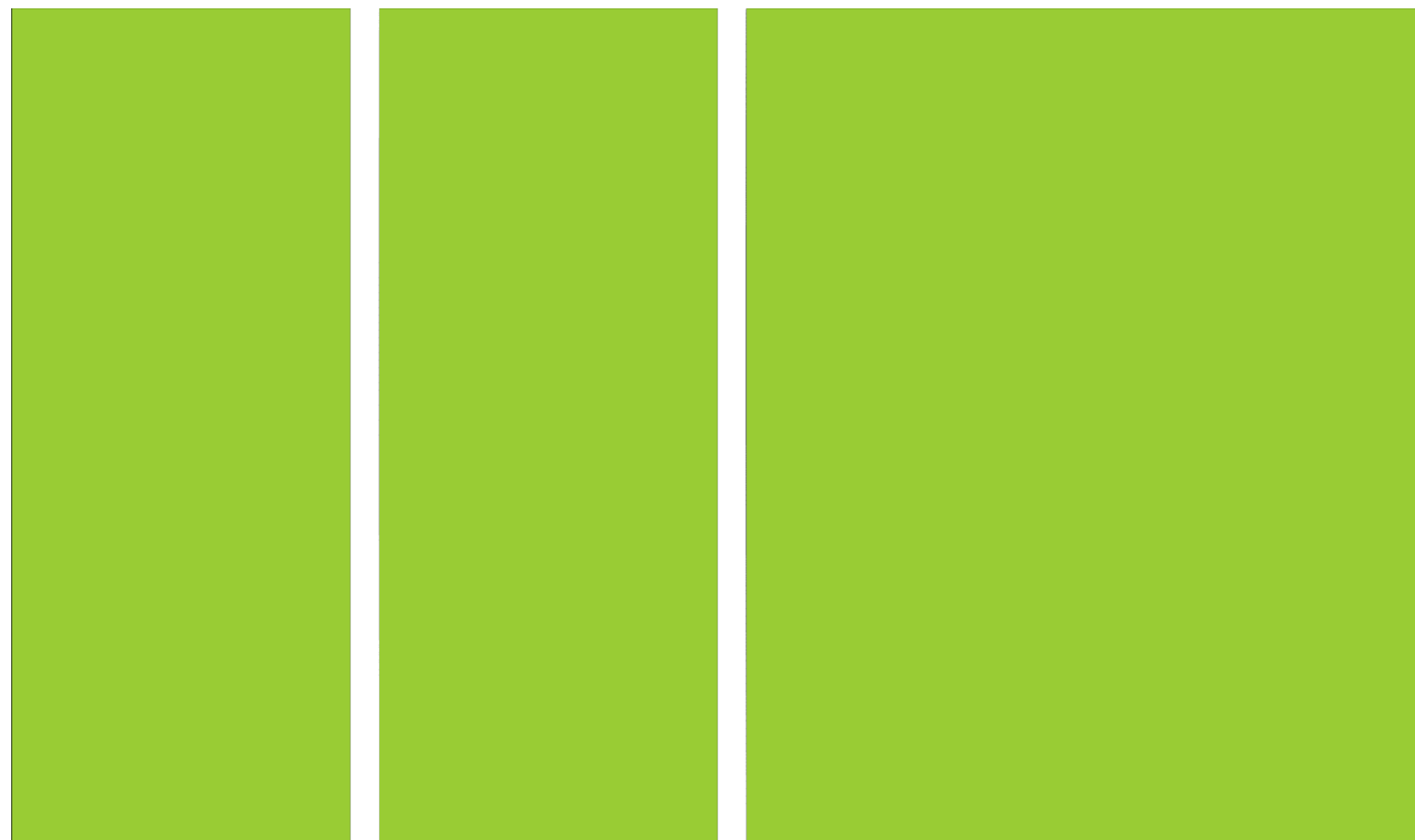


- `justify-content`: alinhamento ao longo do eixo perpendicular

CSS: Propriedades (`display`)

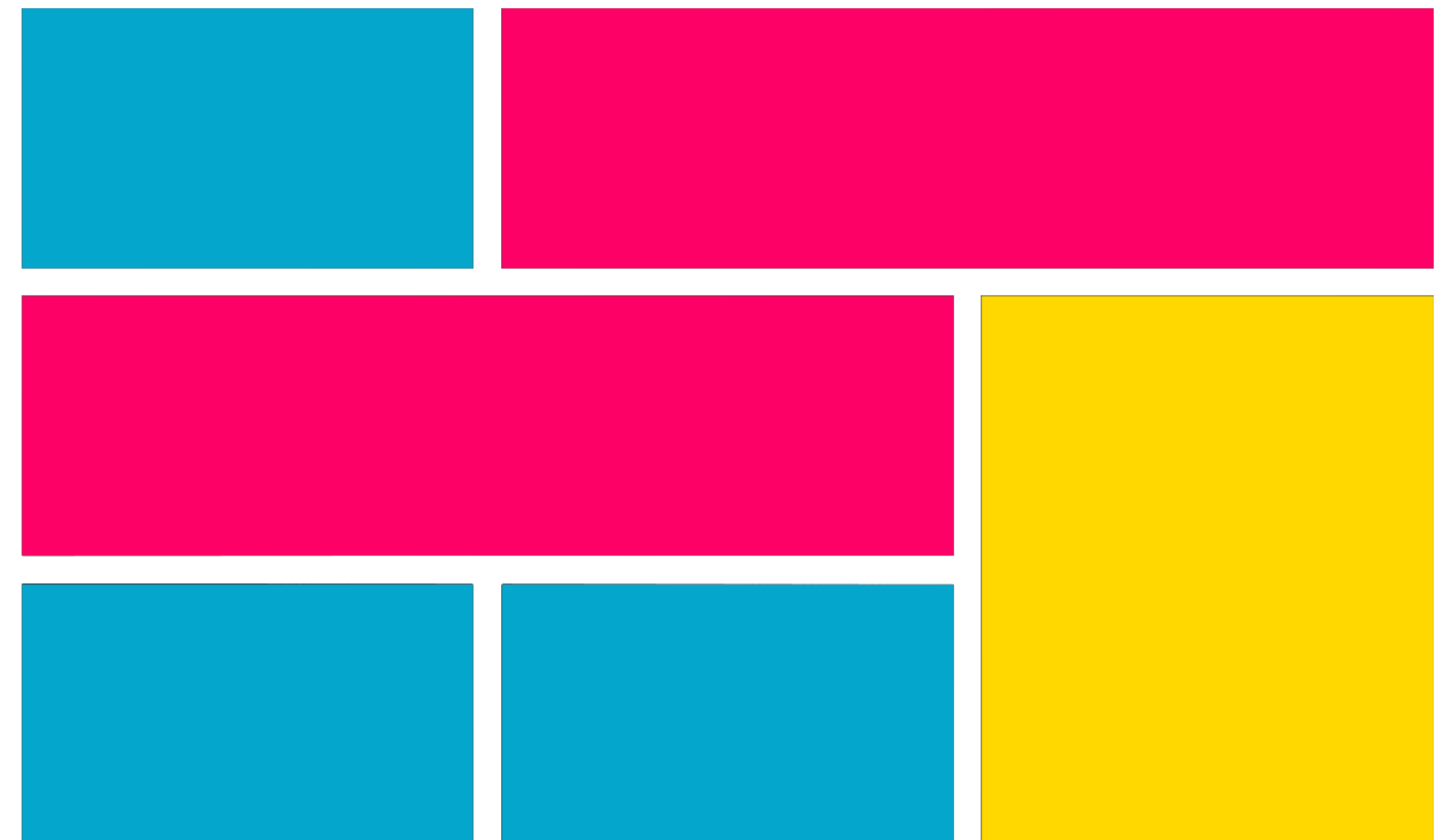
- `flex`

- Uma dimensão



- `grid`

- Duas dimensões



CSS: Propriedades (`display: grid`)

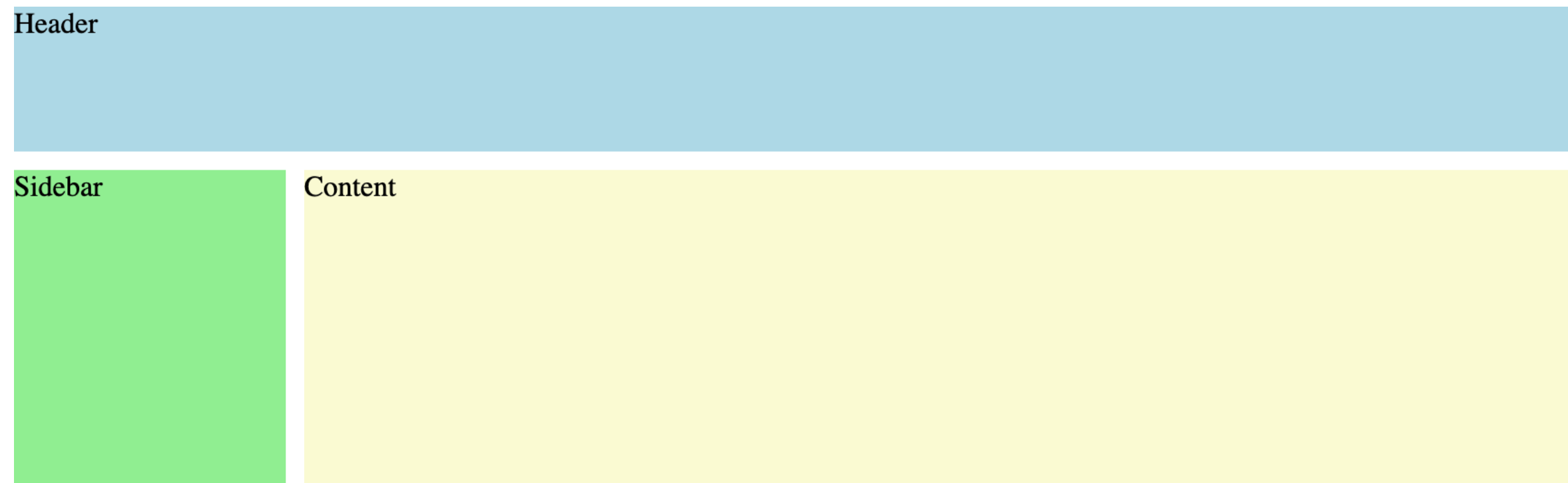
- Permite criar layouts em duas dimensões (linhas e colunas) com facilidade e precisão
- Tamanho de cada coluna / linha (`grid-template-columns`, `grid-template-rows`)
- Espaçamento entre colunas e linhas (`gap`)
- Permitir que items ocupem mais do que uma célula (`grid-row: start / end`, `grid-column: start / end`)

```
.grid-container {  
  display: grid;  
  grid-template-columns: 150px 1fr;  
  grid-template-rows: 80px 1fr 50px;  
  gap: 10px;}  
  
.header {  
  grid-column: 1 / 3;  
  grid-row: 1 / 2;    }  
  
.sidebar {  
  grid-column: 1 / 2;  
  grid-row: 2 / 3;    }  
  
.content {  
  grid-column: 2 / 2;  
  grid-row: 2 / 3;}
```

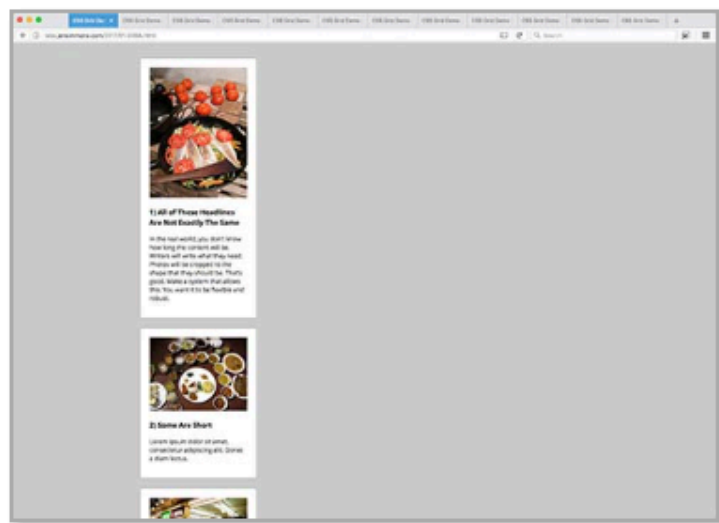
Header

Sidebar

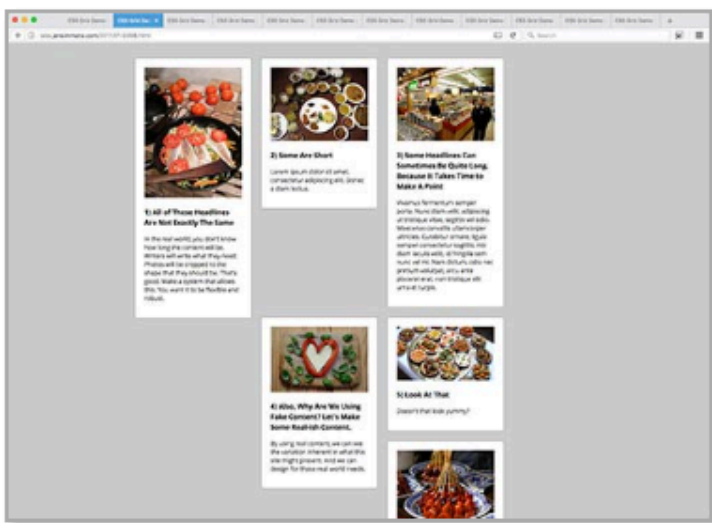
Content



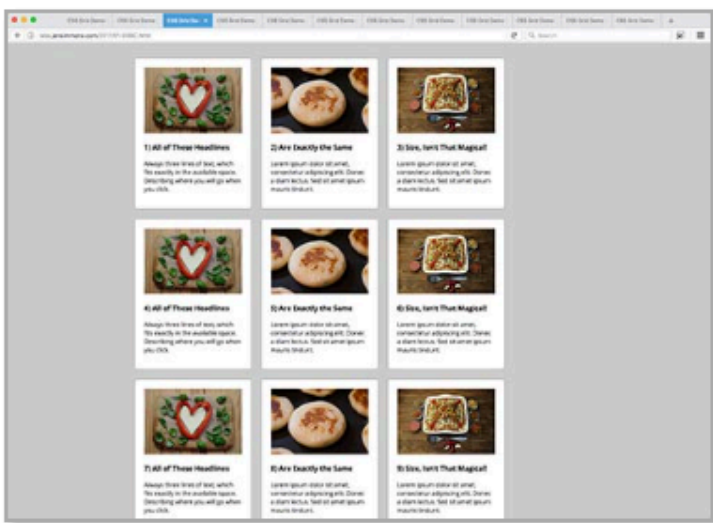
Exemplo: CSS Layouts



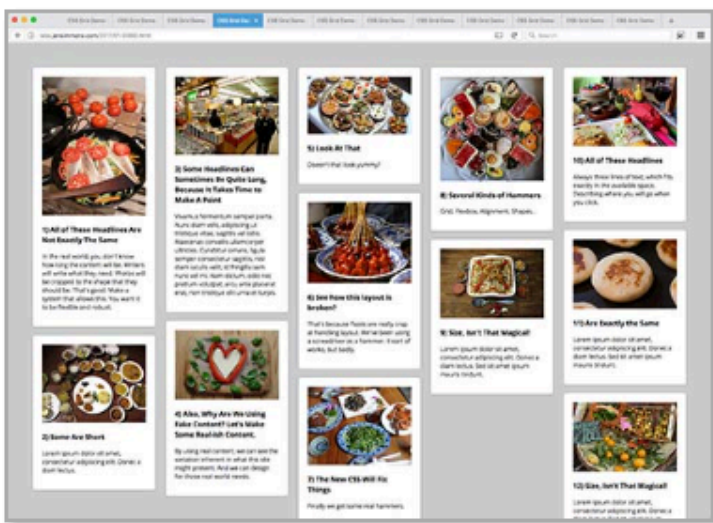
A: Flow layout



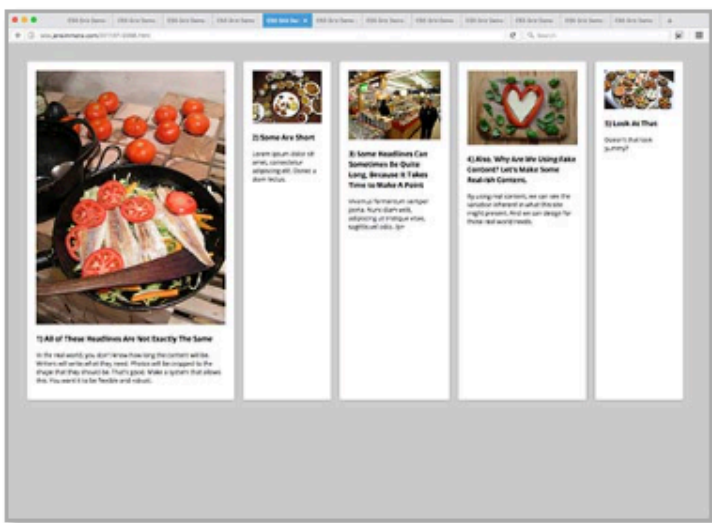
B: Traditional Floats — broken



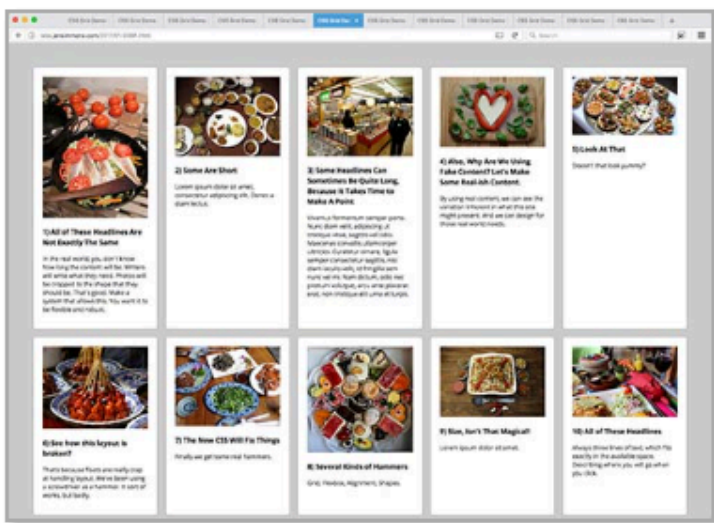
C: Traditional Floats — "fixed"



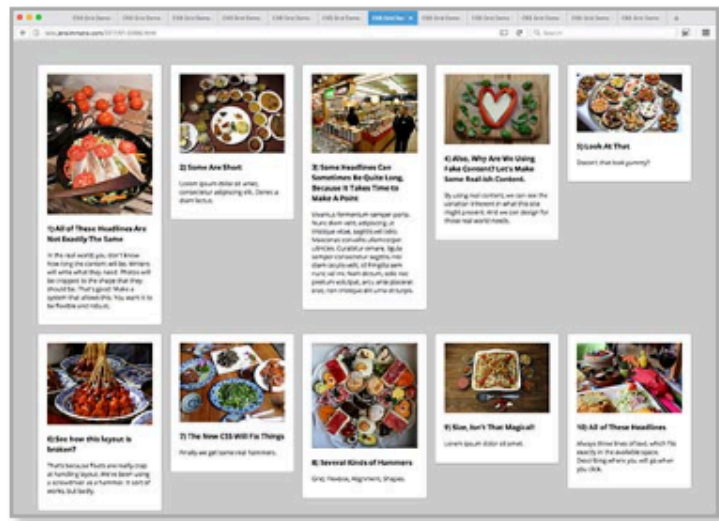
D: Column Layout (using Multicolumn)



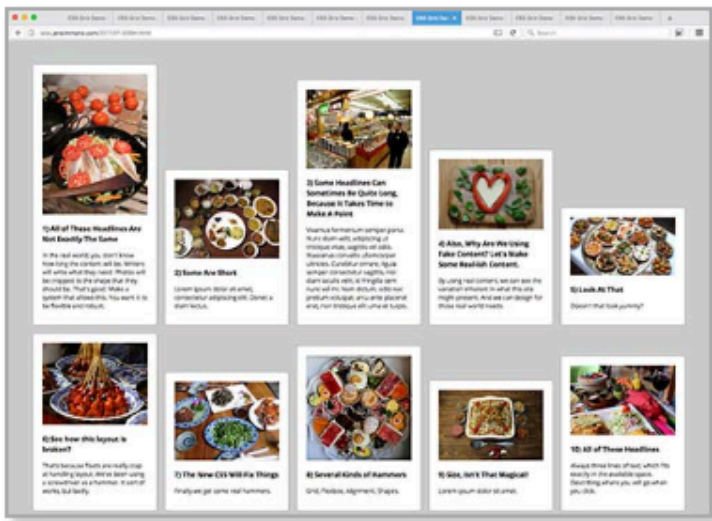
E: Flexbox



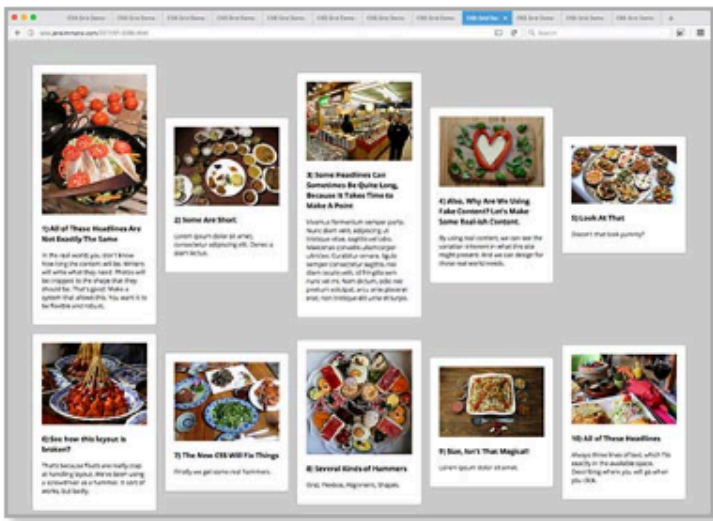
F: CSS Grid — stretch in both dimensions



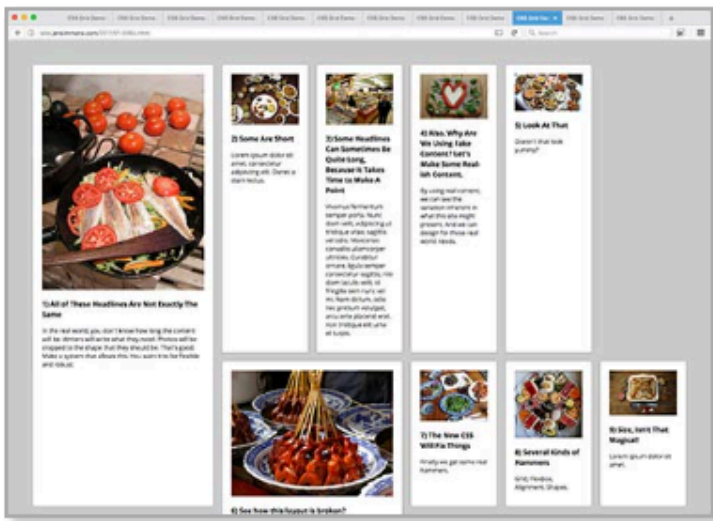
G: CSS Grid — align items start



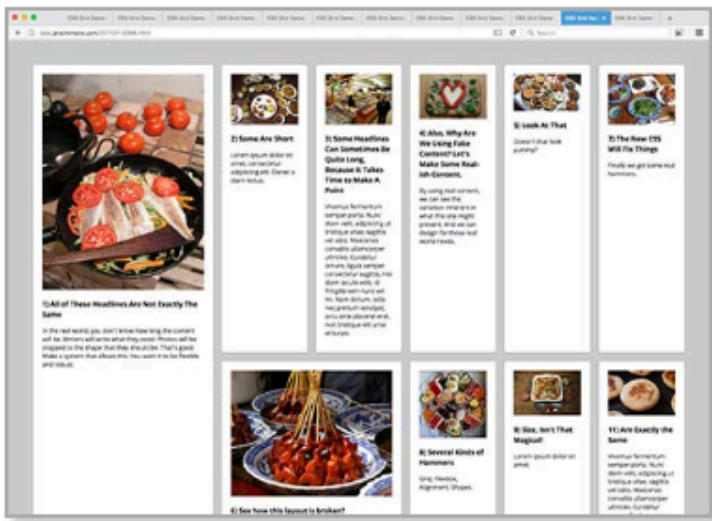
H: CSS Grid — align items end



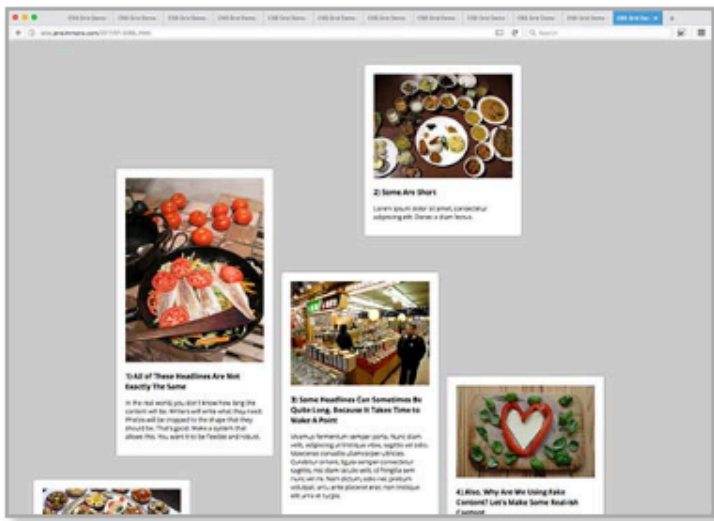
I: CSS Grid — align items center



J: CSS Grid — differently sized items



K: CSS Grid — differently sized items — dense



L: CSS Grid — explicitly placed items (for wide browsers only)

CSS: Variables

- DRY (Don't Repeat Yourself) Principle

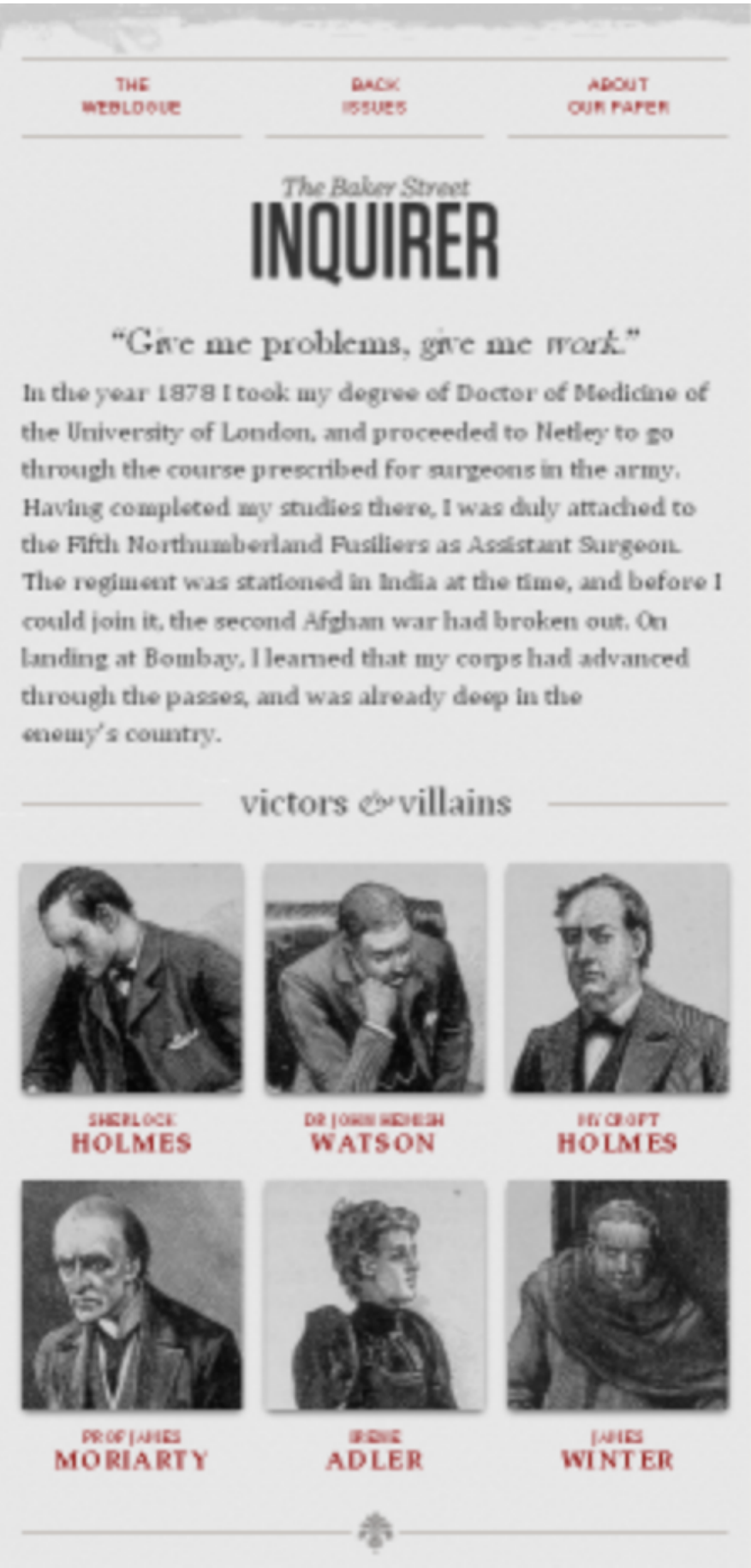
“Every piece of knowledge must have a single, unambiguous, authoritative representation within a system”

(Andy Hunt and Dave Thomas, The Pragmatic Programmer)

- Em CSS, **evitar duplicação utilizando variáveis** (reparem na herança)

```
:root {  
  --primary-bg-color: #1e90ff;}  
  
body {  
  background-color: var(--primary-bg-color);}
```

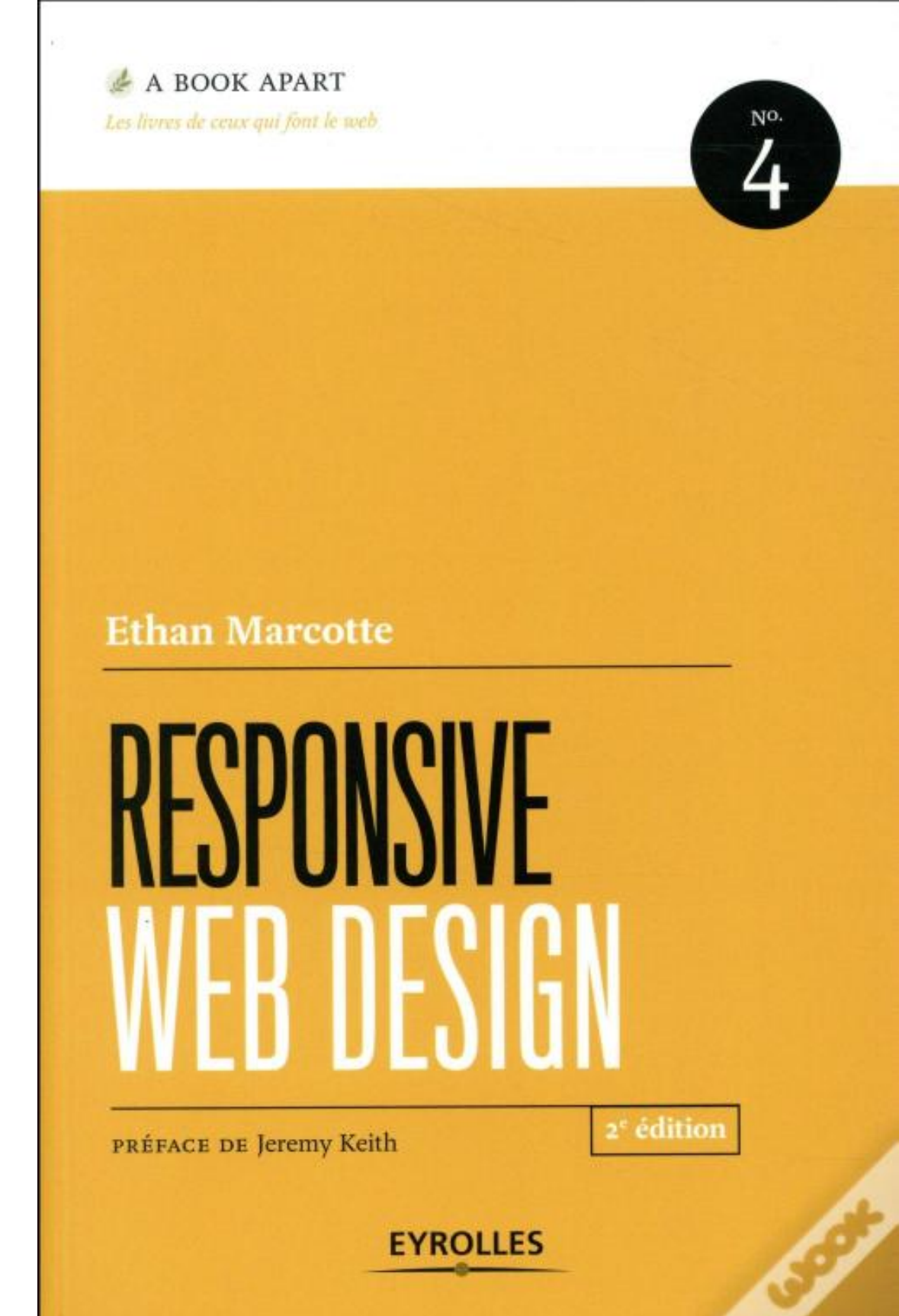

Responsive Web Design



Responsive Web Design

- “Think beyond the desktop, and craft designs that respond to your users’ needs — no matter how large or small the display.”
- 💡 Por defeito:
 - Smartphones renderizam páginas num viewport virtual, e encolhem o resultado
- “Mobile is a magnifying glass for your usability problems.”

(Josh Brewer)



Resilient Web Design

- “It was as though the web design community were participating in a shared consensual hallucination. Rather than acknowledge the flexible nature of the browser window, they chose to settle on one set width as the ideal... even if that meant changing the ideal every few years.”
- “Look at the tools we use for designing on the web. It used to be Photoshop and then it was Sketch. Now it’s Figma. Whatever it is, they’re very precise high-fidelity pixel-perfect tools. They’re very imperative. So we take imperative approach and then we try to translate it into a declarative language like CSS which is supposed to be about setting boundary conditions, leaving the browser to figure out the details.”

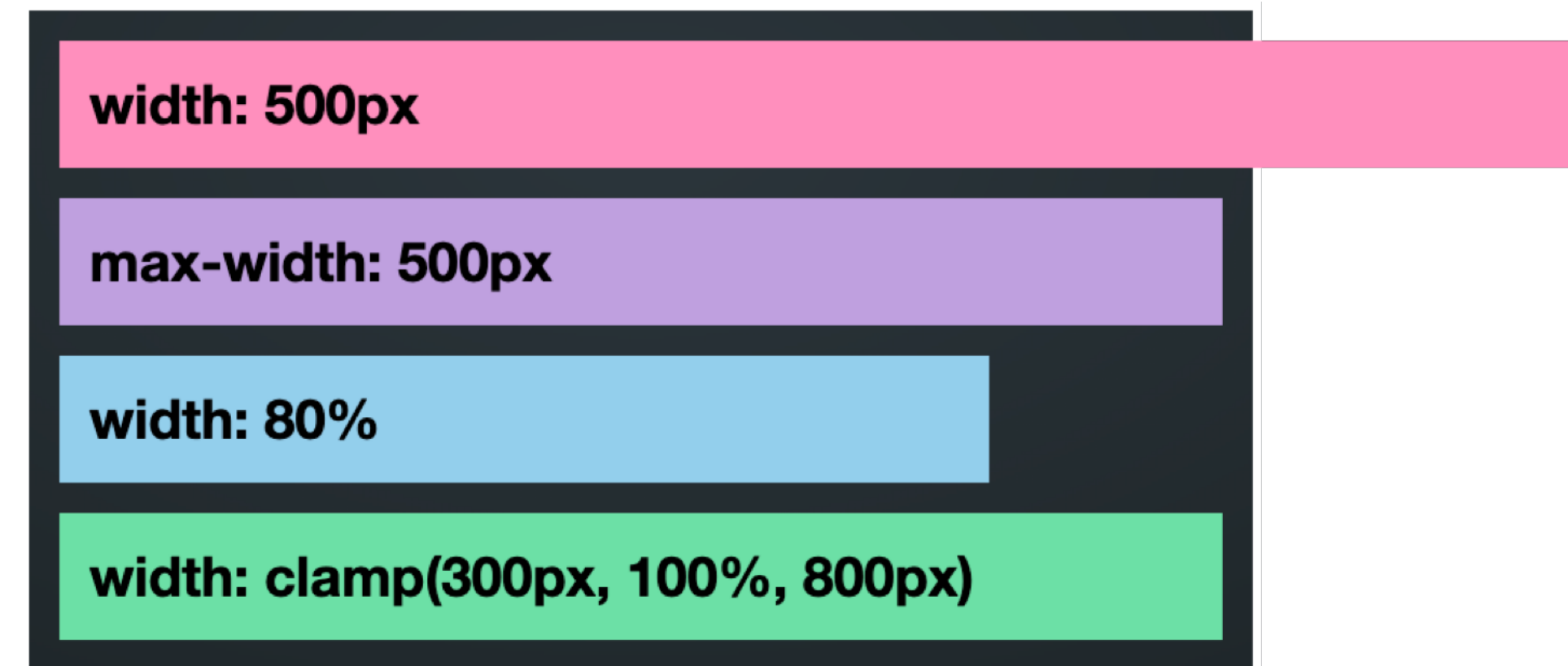
(Jeremy Keith)

- “Be the browser’s mentor, not its micromanager.”

(Andy Bell)

A More Declarative Web Design

- + Avoid absolute widths
- + Do not optimize for specific devices
- + Use relative units
- + Use fluid grids
- + Keep layouts simple
- + **Media queries** (uma forma de aplicar regras CSS com base nas características do browser e do dispositivo)
 - Páginas adaptam-se ao dispositivo para melhorar usabilidade e aparência em qualquer tamanho e orientação do ecrã



CSS: Media queries

- Media features
 - Exemplos: `width`, `height`, `resolution`, `color`, `color-index`, ...

```
@media (width <= 600px) {...}
```

```
@media (600px < width <= 1024px) {...}
```

```
@media (width > 1024px) {...}
```

```
@media (max-width: 600px) {...}
```

```
@media (min-width: 601px) and (max-width: 1024px) {...}
```

```
@media (min-width: 1025px) {...}
```

CSS: Media queries

```
@media (min-width: 300px) {  
    body { background-color: deeppink; }
```

```
    @media (min-height: 500px) {  
        body {
```

```
            background-image:  
                linear-gradient(  
                    to right,  
                    gold,  
                    transparent  
                );
```

```
        }
```

```
    }
```

```
}
```

O mesmo
que um
AND

CSS: Media queries

Colunas laterais de 10px, colunas do meio ocupam 3/4 e 1/4 do restante espaço.

```
@media screen and (width >= 601px) {  
  .grid-container {  
    display: grid;  
    grid-template-columns: 10% 60% 20% 10%;  
    grid-template-rows: auto;  
    grid-template-areas:  
      ". header header ."  
      ". nav nav ."  
      ". main aside ."  
      ". footer footer .";  
  }  
  main {  
    margin-right: var(--small-margin);  
  }  
}
```

Linhas com altura ajustada ao conteúdo.

```
@media screen and (width <= 600px) {  
  .grid-container {  
    display: grid;  
    grid-template-columns: 1fr;  
    grid-template-rows: auto;  
    grid-template-areas:  
      "header"  
      "nav"  
      "aside"  
      "main"  
      "footer";  
  }  
  main {  
    margin-right: 0px;  
  }  
  aside {  
    margin-bottom: var(--small-margin);  
  }  
}
```

Uma só coluna a ocupar todo o espaço disponível