

Sumário

- Sumário.....0**
- Introdução ao CSS.....4**
 - O que é CSS?..... 4
 - História e evolução do CSS..... 5
 - A relação entre HTML, CSS e JavaScript.....9
 - Editores de código e ferramentas de desenvolvimento..... 10
- Sintaxe e Seletores do CSS..... 11**
 - Regras de estilo..... 11
 - Seletores simples e combinados..... 12
 - Seletores de atributo..... 13
 - Pseudo-classes e pseudo-elementos..... 15
 - Especificidade e herança..... 16
- Unidades e Valores do CSS..... 19**
 - Cores..... 19
 - Unidades de medida: absolutas e relativas..... 21
 - Valores numéricos e porcentagens..... 23
 - Funções e cálculos..... 23
 - Variáveis CSS (variáveis personalizadas)..... 25
- Propriedades CSS e técnicas de estilização..... 27**
 - Tipografia e texto..... 27
 - Fundo e cores..... 31
 - Espaçamento e Alinhamento..... 34
 - Margem e Preenchimento..... 34
 - Alinhamento de Texto..... 36
 - Alinhamento Vertical..... 36
 - Estilos de borda..... 38
 - Estilos de lista..... 40
- Layout e Posicionamento..... 42**
 - Display e tipos de caixas..... 42

Display Block.....	44
Display Inline.....	45
Display Inline-Block.....	46
Posicionamento e flutuação.....	48
Position.....	48
Position Static.....	48
Position Relative.....	49
Position Absolute.....	50
Position Fixed.....	51
Position Sticky.....	51
Float.....	54
Modelo de caixa (Box Model).....	57
Overflow e visibility.....	61
Overflow.....	61
Visibility.....	62
Flexbox.....	65
Propriedades e usos do Flexbox.....	67
Flex Container.....	67
flex-direction.....	68
flex-wrap.....	69
flex-flow.....	71
justify-content.....	73
align-items.....	75
align-content.....	77
Propriedades do Flex Item.....	80
order.....	80
flex-grow.....	82
flex-shrink.....	84
flex-basis.....	86
flex.....	89
align-self.....	92
Grid Layout.....	96
Propriedades e usos do Grid Layout.....	98

Definindo uma grade.....	98
grid-template-columns.....	98
Unidades absolutas e relativas:.....	100
Função repeat:.....	100
Função minmax:.....	100
Valor auto:.....	101
grid-template-rows.....	101
grid-auto.....	102
grid-auto-columns.....	102
grid-auto-rows.....	103
grid-gap (gap).....	104
Definições de alinhamento.....	106
justify-items.....	106
align-items.....	107
justify-content.....	108
align-content.....	109
Transições.....	111
Animações.....	114
Controle de animações e desempenho.....	117
will-change.....	117
animation-play-state.....	118
Desempenho e otimização.....	118
Responsividade e Design Adaptável.....	119
Viewport e escala.....	121
Media Queries.....	123
Imagens e vídeos responsivos.....	128
Imagens responsivas usando max-width.....	128
Imagens responsivas usando srcset e sizes.....	129
Imagens responsivas usando o elemento picture.....	129
Vídeos responsivos usando CSS.....	130
Acessibilidade e Inclusão no CSS.....	132
Princípios básicos de acessibilidade na web.....	132
Contraste de cores e legibilidade.....	133

Uso de unidades relativas para acessibilidade.....	136
Media Queries para acessibilidade.....	141
Metodologias CSS.....	146
BEM (Block, Element, Modifier).....	146
OOCSS (Object-Oriented CSS).....	147
SMACSS (Scalable and Modular Architecture for CSS).....	149
Organização e estrutura de projetos.....	152
Boas práticas e Otimização de CSS.....	154
Organização e legibilidade do código.....	154
Reuso e modularização.....	157
Testes automatizados e manuais.....	159
Testes manuais de CSS.....	159
Testes automatizados de CSS.....	160
Conclusão.....	162
Agradecimentos.....	163

Introdução ao CSS

O que é CSS?

CSS (Cascading Style Sheets) é uma linguagem de estilo utilizada para descrever a aparência e formatação de documentos escritos em HTML. O CSS permite que os desenvolvedores web controlem aspectos como cores, fontes, espaçamentos, posicionamento de elementos e muitos outros. A principal vantagem do CSS é a separação do conteúdo (HTML) e da apresentação (CSS), facilitando a manutenção e a atualização do design sem afetar a estrutura do conteúdo.

Gosto de comparar o desenvolvimento de um site, software ou até mesmo de aplicativo, com a construção de uma casa. Para construção de uma casa é fundamental que toda a idéia da obra esteja altamente especificada em um documento, a planta, e para o desenvolvimento de um site não pode ser diferente. No começo da carreira, muitas vezes cometemos o erro de pensar que documentar o projeto não é necessário e partimos diretamente para código, essa atitude muitas vezes gera projetos inconsistentes e de difícil manutenção. Assim como na engenharia civil, a documentação é fundamental, na engenharia de software o mesmo acontece, antes de “colocar a mão massa” e começar a programar, é imprescindível a análise de requisitos do sistema e a criação da documentação detalhada do projeto, onde será especificado todas as funcionalidades do software e como serão desenvolvidas.

Com todas as documentações em mãos aí sim a construção da casa começa, neste processo será desenvolvida toda a estrutura da obra, os pedreiros começam levantar as paredes e a casa já tem estrutura de casa, porém mesmo sendo funcional, a casa ainda não possui acabamento, está apenas no reboco, o que deixa o visual da construção não muito agradável.

Em analogia ao desenvolvimento de um site, esse processo de estruturação e levantamento das paredes pode ser identificado como o desenvolvimento dos nossos arquivos HTML, que tem como objetivo estruturar o visual do nosso site. Um layout todo desenvolvido em HTML é funcional, porém não é visualmente agradável, assim como a casa sem acabamentos.

Enfim chegamos na parte onde nossa casa começa a se tornar atrativa, é adicionado pisos, lustres, portas, janelas e para deixar um ar mais agradável é as paredes ganham cores, dessa forma a construção é concluída.

No desenvolvimento de um site, o responsável por realizar esta etapa é o CSS, com ele você irá deixar o HTML que é visualmente desagradável em algo incrível e totalmente estilizado. O CSS será responsável por realizar todas as estilizações do site aplicativo, mudando cores, tamanhos e posicionamentos.

Assim como não é possível construir uma obra bonita sem acabamentos, é totalmente inviável criar um site sem a utilização de CSS.

Acredito que a partir de hoje você não irá mais enxergar uma construção como antes.

História e evolução do CSS

O CSS, ou Cascading Style Sheets (Folhas de Estilo em Cascata), é uma linguagem de estilo que revolucionou a maneira como os desenvolvedores criam e projetam sites. Antes de explorar a história do CSS, é importante entender o contexto em que ele foi desenvolvido e como ele se encaixa no cenário mais amplo da web.

No início dos anos 1990, a World Wide Web começou a ganhar popularidade, graças ao trabalho de Tim Berners-Lee, que desenvolveu o

HTML (Hypertext Markup Language) e o HTTP (Hypertext Transfer Protocol). O HTML foi projetado para ser uma linguagem de marcação simples, permitindo que os autores de páginas da web estruturarem documentos com ênfase no conteúdo. No entanto, logo ficou claro que o HTML por si só não era suficiente para atender às crescentes demandas de design e personalização dos sites.

Em 1994, Håkon Wium Lie, na época trabalhando no CERN (Organização Europeia para a Pesquisa Nuclear), apresentou a ideia de folhas de estilo para a web. Lie colaborou com Tim Berners-Lee e Bert Bos para desenvolver o conceito e, juntos, eles apresentaram o primeiro rascunho da especificação CSS em 1996.

A ideia por trás do CSS era separar a apresentação (estilo) do conteúdo (HTML), permitindo maior flexibilidade e controle sobre o design dos sites. A primeira versão do CSS, conhecida como CSS1, foi lançada em dezembro de 1996 e oferecia um conjunto básico de recursos para estilizar páginas da web, incluindo fontes, cores e espaçamento.

O CSS1 foi um grande passo à frente, mas logo ficou evidente que havia necessidade de melhorias e recursos adicionais. O CSS2 foi lançado em 1998 e trouxe uma série de novos recursos, como posicionamento absoluto, suporte para mídia impressa e aural, pseudo-elementos e pseudo-classes, entre outros.

No entanto, o desenvolvimento do CSS2 revelou a complexidade de implementar todos esses novos recursos de uma só vez. Como resultado, a abordagem para o desenvolvimento do CSS mudou, passando a adotar um modelo mais modular. Em vez de criar uma única especificação monolítica, o CSS foi dividido em módulos menores que podem ser desenvolvidos e aprimorados independentemente.

Essa mudança levou à introdução do CSS3, que começou a ser desenvolvido em 1999 e tem sido aprimorado continuamente desde então. O CSS3 introduziu uma série de recursos inovadores, como animações, transições, gradientes, transformações, Flexbox e Grid Layout. Esses

recursos permitiram que os desenvolvedores criassem designs mais ricos e interativos, melhorando a experiência do usuário e a usabilidade dos sites.

O CSS tem sido um componente essencial na evolução da web, permitindo que os desenvolvedores criem sites visualmente atraentes e funcionais. A história do CSS é uma história de inovação e progresso, mostrando como a comunidade de desenvolvimento da web trabalhou em conjunto para enfrentar os desafios e atender às demandas crescentes dos usuários e das tecnologias em evolução.

Hoje, o CSS continua a evoluir, com a introdução de novos recursos e melhorias nas especificações existentes. O desenvolvimento do CSS é conduzido pelo World Wide Web Consortium (W3C), a organização internacional que estabelece padrões para a web. O W3C trabalha com desenvolvedores, designers, fabricantes de navegadores e outras partes interessadas para garantir que o CSS continue a atender às necessidades do mundo digital em constante mudança.

Além das melhorias contínuas nas especificações CSS existentes, novos módulos e recursos estão sendo propostos e desenvolvidos, como o CSS4, que busca expandir ainda mais as capacidades de design e interação na web. Algumas das áreas em que o CSS está avançando incluem:

Melhorias na acessibilidade: o CSS está se tornando cada vez mais focado em garantir que os sites sejam acessíveis a todos os usuários, independentemente de suas habilidades ou dispositivos utilizados.

Design responsivo: com a variedade de dispositivos e tamanhos de tela disponíveis atualmente, o CSS está se adaptando para ajudar os desenvolvedores a criar designs que funcionem bem em diferentes contextos e resoluções de tela.

Personalização do usuário: o CSS está explorando novas maneiras de permitir que os usuários personalizem a aparência dos sites de acordo com suas preferências e necessidades.

Integração com outras tecnologias: à medida que a web se torna mais complexa, o CSS está trabalhando em conjunto com outras tecnologias, como HTML, JavaScript, Web Components e outras APIs, para criar experiências na web mais ricas e integradas.

A história do CSS é um exemplo do compromisso contínuo da comunidade de desenvolvimento da web com a inovação e a melhoria. À medida que a web continua a evoluir, é provável que o CSS também evolua, trazendo novos recursos e capacidades para ajudar os desenvolvedores e designers a criar sites mais atraentes, funcionais e acessíveis para todos os usuários.

A relação entre HTML, CSS e JavaScript

HTML, CSS e JavaScript são as três principais tecnologias usadas para criar websites e aplicações web. O HTML (Hypertext Markup Language) é responsável pela estrutura e conteúdo do documento, o CSS define a aparência e o estilo, enquanto o JavaScript é uma linguagem de programação utilizada para adicionar interatividade e funcionalidades avançadas.

Em um projeto web, o HTML cria a estrutura básica, o CSS é aplicado para estilizar os elementos e o JavaScript é usado para manipular os elementos e adicionar comportamento dinâmico.

Voltando a analogia com uma construção de uma casa, onde o HTML é a estrutura básica, tais como as paredes e o telhado, o CSS sendo os acabamentos, como pinturas e pisos. O JavaScript nesta comparação poderia equivaler a automatização da casa, tais como sensores de presença para ligar luzes, portão eletrônico e dentre outras coisas.

O JavaScript consegue deixar o nosso site que é estático em algo dinâmico, altamente intuitivo. Com esta linguagem de programação você é capaz de manipular os elementos da página web e adicionar eventos que serão executados quando algo acontecer. Um exemplo seria modificar um texto quando um botão for clicado.

Editores de código e ferramentas de desenvolvimento

Existem muitos editores de código disponíveis para escrever CSS, HTML e JavaScript.

Alguns dos editores mais populares incluem:

- Visual Studio Code
- Sublime Text
- Atom
- Brackets

Esses editores de código geralmente oferecem recursos como realce de sintaxe, autocompletar, e integração com outras ferramentas e extensões úteis.

Além dos editores de código, os desenvolvedores também utilizam ferramentas de desenvolvimento integradas aos navegadores (como Chrome DevTools, Firefox Developer Tools e Safari Web Inspector) para inspecionar e modificar o CSS, HTML e JavaScript diretamente no navegador, facilitando a depuração e a experimentação.

Sintaxe e Seletores do CSS

Regras de estilo

As regras de estilo CSS são fundamentais na formatação dos elementos HTML. Cada regra de estilo é composta por um seletor e um bloco de declarações. O seletor indica a quais elementos HTML a regra se aplica, e o bloco de declarações contém uma ou mais declarações CSS. Cada declaração consiste em uma propriedade e um valor, separados por dois pontos, e termina com um ponto e vírgula.

Exemplo de uma regra de estilo CSS:

```
p {  
  font-size: 16px;  
  color: #333;  
}
```

Neste exemplo, o seletor é "p", que se refere a todos os elementos de parágrafo na página HTML. O bloco de declarações contém duas declarações, uma para a propriedade "font-size" e outra para a propriedade "color". A primeira declaração define o tamanho da fonte dos parágrafos em 16 pixels, enquanto a segunda declaração define a cor do texto como um tom escuro de cinza (#333).

A sintaxe das regras de estilo é essencial para garantir que o navegador entenda e aplique corretamente as formatações desejadas. É importante seguir a estrutura correta ao escrever suas regras de estilo e garantir que cada declaração termine com um ponto e vírgula para evitar erros de interpretação.

Seletores simples e combinados

Seletores simples são usados para selecionar elementos baseados em tipos de elemento, classes ou IDs. Seletores combinados permitem combinar seletores simples para criar seletores mais específicos e direcionados.

Tipo de elemento: Os seletores de tipo de elemento são usados para aplicar estilos a todos os elementos de um determinado tipo. Por exemplo, `h1`, `p` e `button` são seletores de tipo de elemento que selecionam todos os elementos `h1`, parágrafos e botões na página, respectivamente.

Classe: Os seletores de classe são usados para aplicar estilos a elementos que compartilham uma classe específica. As classes são atributos HTML que permitem agrupar elementos com características semelhantes. Para selecionar elementos por classe, use o símbolo do ponto (.) seguido pelo nome da classe. Por exemplo, `.destaque` e `.botao` são seletores de classe que selecionam todos os elementos com as classes "destaque" e "botao", respectivamente.

ID: Os seletores de ID são usados para aplicar estilos a um único elemento com um ID específico. Os IDs são atributos HTML que devem ser únicos em toda a página. Para selecionar um elemento por ID, use o símbolo do cardinal (#) seguido pelo nome do ID. Por exemplo, `#cabecalho` e `#rodape` são seletores de ID que selecionam os elementos com os IDs "cabecalho" e "rodape", respectivamente.

Seletores combinados permitem combinar seletores simples para selecionar elementos com base em suas relações e características combinadas. Por exemplo:

Descendente: Os seletores descendentes são usados para selecionar elementos que são descendentes de outros elementos. Para criar um seletor descendente, separe os seletores dos elementos pai e filho por um espaço.

Por exemplo, `div p` seleciona todos os elementos `p` que estão dentro de uma `div`, independentemente de quantos níveis de aninhamento existam entre eles.

Filho: Os seletores de filho são usados para selecionar elementos que são filhos diretos de outros elementos. Para criar um seletor de filho, separe os seletores dos elementos pai e filho por um sinal de maior que (`>`). Por exemplo, `div > p` seleciona apenas os elementos `p` que são filhos diretos de uma `div`, ignorando aqueles que estão aninhados mais profundamente.

Irmãos adjacentes: Os seletores de irmãos adjacentes são usados para selecionar elementos que compartilham o mesmo pai e são adjacentes um ao outro na estrutura do documento. Para criar um seletor de irmãos adjacentes, separe os seletores dos elementos irmãos por um sinal de adição (`+`). Por exemplo, `h1 + p` seleciona o primeiro elemento `p` imediatamente após um `h1` que compartilha o mesmo elemento pai.

Irmãos generalizados: Os seletores de irmãos generalizados são usados para selecionar elementos que compartilham o mesmo pai e são irmãos de outros elementos, independentemente de sua posição relativa. Para criar um seletor de irmãos generalizados, separe os seletores dos elementos irmãos por um til (`~`). Por exemplo, `h1 ~ p` seleciona todos os elementos `p` que são irmãos de um `h1` e compartilham o mesmo elemento pai.

Seletores de atributo

Os seletores de atributo permitem selecionar elementos com base nos atributos HTML e seus valores. Esses seletores são úteis para aplicar estilos a elementos que possuem atributos específicos ou atributos com valores específicos. Os seletores de atributo podem ser simples ou combinados, dependendo de como são usados.

Presença de atributo: Para selecionar elementos que possuem um atributo específico, independentemente de seu valor, use colchetes ([]) com o nome do atributo. Por exemplo, `input[type]` seleciona todos os elementos `input` que possuem o atributo `type`.

Valor de atributo exato: Para selecionar elementos com um atributo que possui um valor exato, use colchetes ([]) com o nome do atributo, seguido por um sinal de igual (=) e o valor entre aspas. Por exemplo, `input[type="text"]` seleciona todos os elementos `input` com o atributo `type` igual a `"text"`.

Valor de atributo contendo palavra: Para selecionar elementos com um atributo cujo valor contém uma palavra específica, use colchetes ([]) com o nome do atributo, seguido por um til e sinal de igual (~=) e a palavra entre aspas. Por exemplo, `input[class~="destaque"]` seleciona elementos `input` cujo atributo `class` contenha a palavra `"destaque"`.

Outros operadores podem ser usados para combinar seletores de atributo de maneira mais complexa, como selecionar elementos cujo valor do atributo comece com, termine com ou contenha uma sequência específica de caracteres.

Pseudo-classes e pseudo-elementos

Pseudo-classes e pseudo-elementos são seletores especiais que permitem selecionar elementos com base em seu estado, posição na estrutura do documento ou características específicas que não podem ser alcançadas usando apenas seletores simples e combinados.

Pseudo-classes: As pseudo-classes são usadas para selecionar elementos com base em seu estado ou posição no documento. Elas são precedidas por dois pontos (:). Exemplos de pseudo-classes incluem **:hover**, **:focus**, **:active**, **:nth-child()**, **:first-child**, e **:last-child**. Esses seletores podem ser usados para aplicar estilos a elementos em resposta a ações do usuário, como mover o cursor sobre um elemento, ou para estilizar elementos específicos dentro de uma lista de itens.

Exemplo de uso de pseudo-classes:

```
button:hover {  
  background-color: #0070f3;  
}  
  
li:nth-child(odd) {  
  background-color: #f3f3f3;  
}
```

No primeiro exemplo, a pseudo-classe **:hover** é usada para alterar a cor de fundo dos botões quando o cursor do usuário está sobre eles. No segundo exemplo, a pseudo-classe **:nth-child(odd)** é usada para aplicar uma cor de fundo diferente aos itens ímpares de uma lista.

Pseudo-elementos: Os pseudo-elementos são usados para selecionar