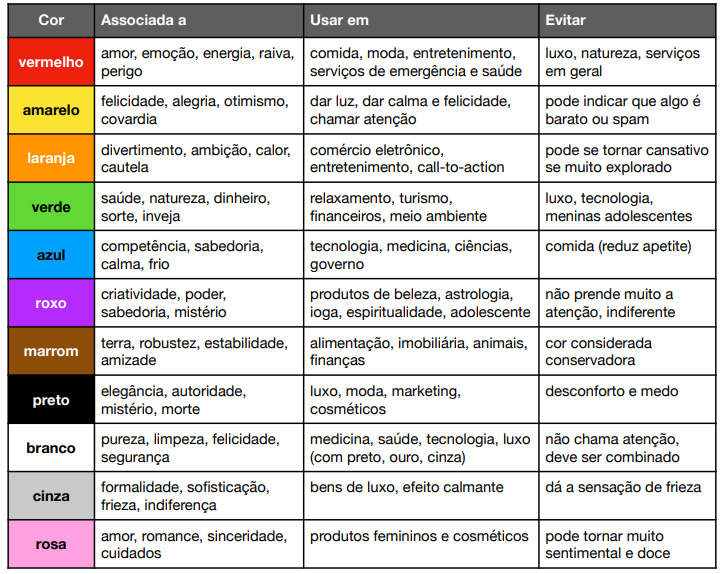
**Módulo 1 -** [https://www.youtube.com/MODULO1](https://www.youtube.com/watch?v=Ejkb_YpuHWs&list=PLHz_AreHm4dkZ9-atkcmcBaMZdmLHft8n&pp=iAQB)

**Capitulo 13 – Cores**

Aula sobre a teoria das cores e suas aplicações, exemplos de como usa-las e orientações de que algumas cores estão diretamente ligadas com diferentes situações ou objetos.

Mas vimos por aí várias marcas que aplicam cores que em nada se identificam com o seu conceito, e mesmo assim fazem sucesso. Não devemos fazer dessas exceções uma regra.

Vamos ter exemplos do que as cores estão diretamente ligadas, mas nunca se prenda a ela.

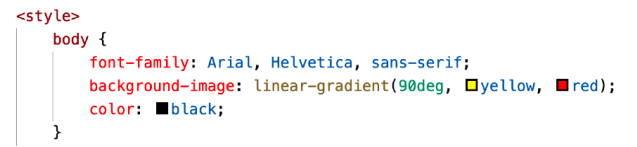


Uma ferramenta interessante, e gratuita para usarmos o conhecimento sobre as cores e muito completas para escolhermos as cores do nosso projeto, seria o [**Adobe Color**](https://color.adobe.com/pt/create/color-wheel), temos também o [**Paletton**](https://paletton.com/#uid=1000u0kllllaFw0g0qFqFg0w0aF), e por último e mais interessante uma ferramenta que gera cores aleatórias chama [**Coolors**](https://coolors.co/).

**Gradientes no CSS**

Já vimos quase tudo sobre cores em nosso material de estudo e como adiciona-las em nosso **CSS**, mas e se quisermos dar um efeito de gradiente.

Sim é estranho, mas o nosso navegador entender o gradiente como uma imagem, e para isso vamos usar a propriedade **background-image** e a função **lineal-gradiente**, que por si só é auto explicativa. Bom o primeiro parâmetro dessa função é o ângulo de inclinação, quer começa em 90 graus ou, para ser exato como na função ***90deg***, e os demais parâmetros são para indicar as cores do degrade.



É possível também criar outra forma de gradiente, os chamados gradientes radiais, que também são autoexplicativos.



Use o código dessa maneira para criar gradientes de forma circular, e também podemos editar ainda mais o nosso degrade colocando umas certa porcentagem ao lado das nossas cores, como : **o red 10%, yellow 40%, green 50%**.

Para adicionar transparência, em qualquer um desses modelo, basta descrever o valor que varia de 0.1 até 1, ao final das declarações dos valores das cores.

**HORA DE EXERCITAR**

Agora em nosso material de apoio, no modulo 2, vamos realizar o exercício 016.

**Capitulo 14 – Vamos falar das Fontes**

Material contanto historicamente como surgiu as Fontes, suas características e tipo. E assim como em cores, determinados tipos de letras se aplicam melhor em situações específicas.

**Como aplicar na Prática:**

Basicamente para aplicar fontes no nosso código, vamos estilizar por meio da **CSS** usando a propriedade ***font-family***. Se indicarmos mais de uma fonte, uma na sequência da outra, o navegador vai seguir na ordem até que achei uma fonte que tenha para reproduzir. Seguindo no exemplo:



Aqui temos a seguinte situação, nosso títulos principais <h1> usarão preferencialmente a fonte **Franklin Gothic Medium**, porem essa mesma não existe em smartphones, que possuem a fonte **Arial** **Narrow**. Caso nenhuma dessas funcione, o navegador busca a mais genérica, como **Arial**, e em último caso se nada funcionar o sistema vai buscar uma fonte genérica sem serifa.

**Vamos falar sobre tamanhos**

Para especificar tamanho de fontes, existem várias medidas como **cm** (centímetros), **in** (polegadas), **pt** (pontos), **pc** (paicas), **px** (pixels), etc. Para tamanhos de fonte a serem exibidos na tela, o **W3C** recomenda o uso do **px** ou do **em**.

A medida **em** é uma das que gera mais dúvida nos alunos. Ela é uma medida referencial em relação ao tamanho original da fonte. O tamanho padrão de uma fonte é geralmente **16px**, isso equivale a **1em**.



**Outros estilos**

Outra maneira de dar estilos a nossas fonte é por meio da CSS usando as propriedades ***font-style*** para aplicar ***itálico*** ou ***font-weight*** para aplicar **negrito**, contudo se perde o fator semântico.



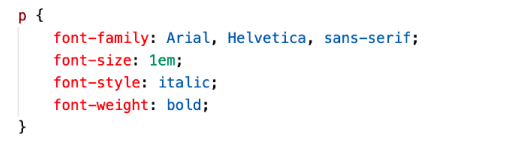
As duas propriedades aceitam valores diferentes:

* ***font-style***: para deixar a fonte em itálico vamos usar italic(mais compatível) ou oblique(menos compatível)
* ***font-weight***: para deixar a fonte em negrito vamos suar valores de 100 até 900, ou valores para deixar em negrito que podem ser:
  + **lighter**
  + **bold**
  + **bolder**

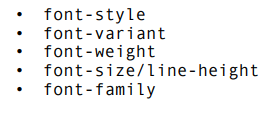
**Me dá uma mãozinha?**

As formatações em CSS são tão importantes que existem até atalhos que são chamados de ***shorthands***.

E um desses ***shorthands***, é o **font**, que no lugar de fazer várias configurações em várias linhas, se resume a uma única linha. Por exemplo:

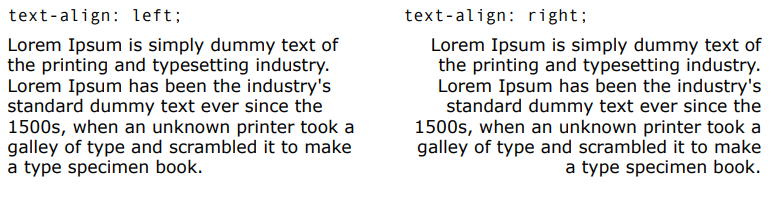
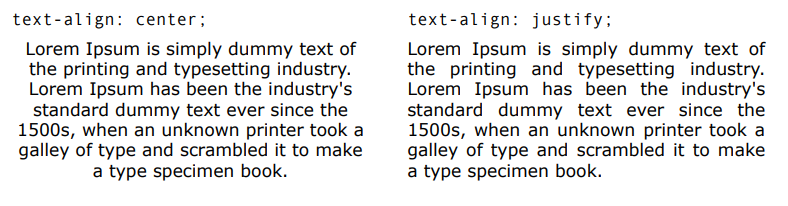
 

Porém temos que tomar cuidado com a ordem do atributos:



**Alinhamentos**

Existem quatro tipos de alinhamentos:

**Como descobrir uma fonte que está sendo usada em outro site?**

No capítulo anterior, te ensinei a usar a extensão **Colorzilla** para pegar uma cor que estava sendo usada em outro site. Agora vou te ensinar a usar a extensão **Fonts** **Ninja** do Google Chrome para capturar a fonte usada em componentes de texto.

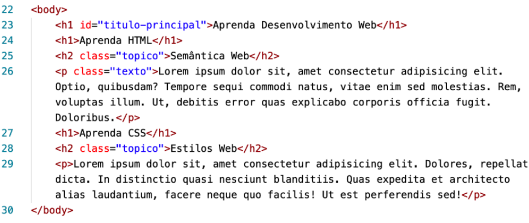
**Hora de exercitar**

Realize o **exercício 017** junto do material de apoio.

**Capitulo 15 – Seletores personalizados em CSS**

**Personalizando Seletores**

Para dar mais poder as CSS, precisamos aprender a usar seletores de **id** **(#)** e **class** **(.)**. identificando um elemento único com um **id** ou agrupando vários elemento com **class**.

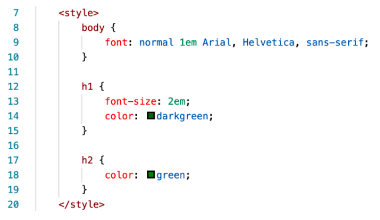


**LINHA** **23**: vimos que **<h1>** principal se diferencia das demais que estão nas **LINHAS 24 e 27**, pois não possuem um **id**.

LINHAS 25e 28: agora percebemos que as linhas possuem a mesma **class**.

Um **id** vai **identificar** um único elemento na página atual, e permite que criamos estilizações especiais para um elemento isolado. Já um **class** vai identificar uma **classe** a qual um ou mais elementos pertencem, compartilhando as características.

Agora baseado no código HTML visto anteriormente, vamos usar a tag <style> para criar um estilo



Oque fizemos aqui foi aplicar os estilos nas **tag’s** **<h1> e <h2>** e em **body**. Até agora não fizemos nada utilizando um seletor para uma configuração personalizada.

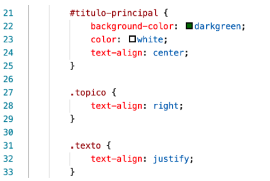


Vamos notar que **<h1>** notamos que continua com a mesma estilização mesmo tendo seletores id diferentes.

Isso acontece porque não realizamos nenhuma configuração visual utilizando esses seletores que falamos anteriormente.

Não adiante adicionar um seletor **id** ou **class** em um elemento e não personalizar o estilo usando esses seletores.

Vamos adicionar algumas declarações com seletores dentro da área **<style>**.



No código colocamos seletores com simbologias novas, isso porque não são seletores para elemento **HTML** e sim para **id** e **class**.

**LINHA** **21**: criamos um seletor personalizado para o identificador título-principal. E como está no código, colocamos o **sinal** **de** **#** quando é um **id**.

**LINHAS** **27** **e** **31**: seletores que vão receber declarações para as classes tópico e texto. Todos os seletores class vão receber o **sinal** **de** **ponto** que vira antes do nome da classe.

Ao adicionar o seletores o resultado final fica bem diferente:



**Propriedades** **Herdadas**

Já falamos anteriormente sobre o 3 tipos de **CSS** que podemos criar e foi dito que qualquer **CSS** **inline** se sobrepões aos outros **CSS** que tiverem sendo aplicados no mesmo seletor.

Bom agora como estamos falando de seletores personalizados, ou explicar como funciona a hierarquia.



Nessa ocasião eu usei o seletor de tag para dar a propriedade de cor e tamanho para o elemento **h1**, e assim os dois **h1** em questão vai receber a mesma propriedade.

Quando eu uso o setor, somente um **h1** receber propriedades a mais ou sobrescritas, como assim? Suponhamos que o seletor id funciona como o **CSS** **inline**, ele tem mais poder que o seletor de **tag**, porém ele adiciona propriedades, e quando for a mesma propriedade, a propriedade do seletor **id** vai ter mais força.

Analisando o código eu posso dizer que color teve sua cor alterada e ganhou duas novas propriedades.



Mas vale notar que como o seletor **id** está em apenas um dos **h1**, somente um deles vai receber essas novas propriedades.

**Pseudoclasses e pseudoelementos**

Uma **pseudoclasse** **CSS** é uma palavra-chave adicionada às declarações de um seletor

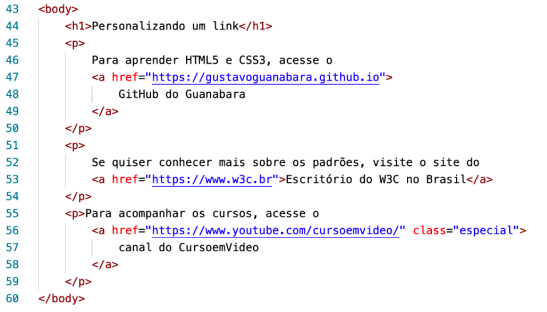
após um sinal de dois pontos e especificam um estado especial de um elemento.

Existem várias pseudoclasses para estilos, podemos citar :**hover, :visited, :active, :checked, :empty e :focus.**

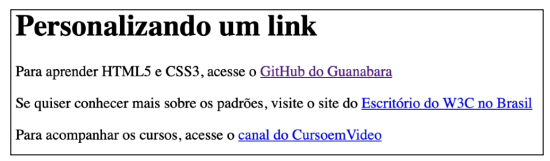
Já um **pseudoelemento** **CSS** é uma palavra-chave adicionada às declarações de um

seletor após dois sinais de dois pontos e permitem que você formate um pedaço

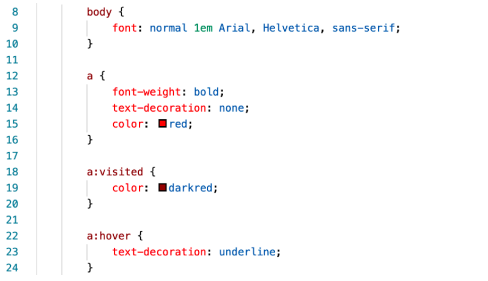
específico do elemento referenciado. Os principais **pseudoelementos** usados nas **CSS** **são ::before, ::after, ::first-letter, ::first-line**.



Note aqui que esse código HTML somente o último parágrafo que recebe um link tem uma classe.



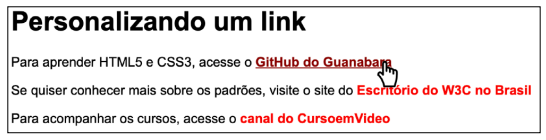
Na imagem de cima o primeiro link aparece diferente dos demais, isso acontece porque é um link que já foi visitado. Por padrão links inéditos aparecem em azul. Vamos alterar essas propriedades dentro de style.



LINHA 12: declaramos que os linka vão receber novas propriedades, colocando o texto em negrito, removendo o sublinhando e alterando a cor do texto para vermelha.

LINHA 18: com a pseudoclasse **:visited** alteramos o parâmetro de cor para o link já visitado.

LINHA 22: com a pseudoclasse **:hover** quando passamos com o mouse por cima do link o sublinhado volta a aparecer.

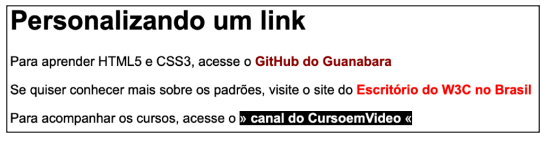


Por final vamos adicionar algumas configurações relacionadas a casse adicionada.



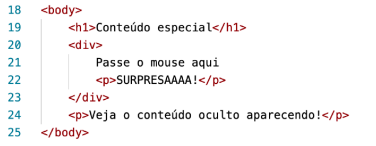
**LINHA 26**: alteramos valores de cor, cor de fundo e sublinhado, mas que esses parâmetros só estarão visíveis quando o mouse passar por cima do link em questão.

**LINHAS 32 e 37**: vamos adicionar um símbolo » antes e outro símbolo « depois usando os pseudoelementos **::before e ::after**, respectivamente



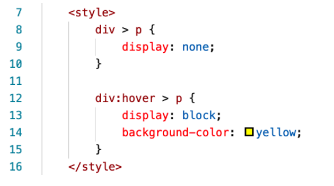
**Mais um exemplo**

Mais um exemplo para o uso de pseudoclasse. Nosso **HTML** vai ter o seguinte corpo.



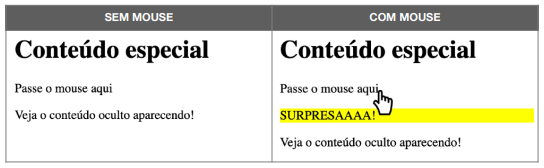
**LINHA 20**: criamos um bloco especial com a **tag <div>**. Uma das grandes vantagem é que podemos usar outras tag’s dentro dela, assim como o parágrafo na **LINHA** **22**.

Vamos agora criar o seguinte código CSS



**LINHA** **8**: vai esconder o parágrafo que está dentro da **div**, que no **CSS** é representado como **DIV > P**, através da propriedade **display** com o valor **none**.

**LINHA 12**: vai fazer o parágrafo escondido aparecer, com o fundo pintado de amarelo, apenas quando o mouse passar sobre ele.



**Hora de exercitar**

Vá no seu material de apoio e realize o **exercicio 017** que esta realacionado com o materia entregue até agora.

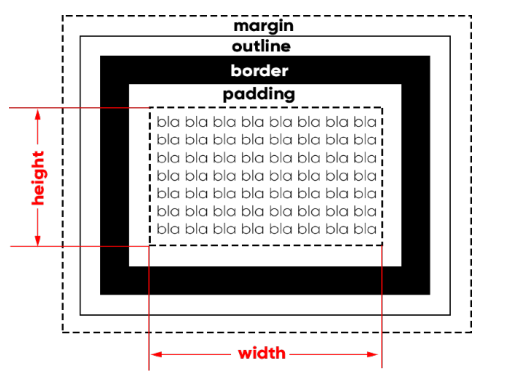
**Capitulo 16 – Modelo de Caixas**

**O que é uma caixa?**

De forma simples é baseado em um conceito chamado **box model**, a grende maioria dos elemtenos HTML esão em forma de caixas. São ***containers*** que armazezam conteudos ou outras caixas.

**Anatomia de uma caixa**

Olhe atentamente o diagrama a seguir, que é exatamente o já citado **modelo de caixa**:



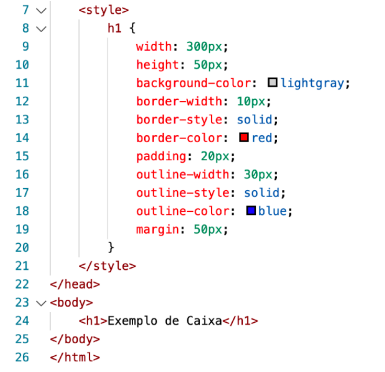
Tudo começa com o **conteúdo** (***content***), representado por “blá blá blá”. Por padrão todos os elementos não vêm com **padding**, **border**, **outline** e nem **margin**. Curioso notar que <body> tem por padrão *margin* de 8px.

Todo conteúdo possui uma **largura** (**width**) e uma **altura** (**height**) e a esse conjunto de propriedades, damos o nome de **box-size** (tamanho da caixa). O tamanho da caixa não inclui as medidas de **padding**, **border**, **outline** e **margin**.

Entre a borda e o conteúdo - da borda para dentro - temos o preenchimento (**padding**) e da borda para fora, temos a margem (**margin**).

Entre a margem e a borda, podemos determinar o contorno (**outline**) que é muito pouco utilizado, mas existe.

Vamos criar um exemplo simples para exemplificar todos esses componentes, configurando as propriedades do modelo de caixa de um título <h1>.



**Linha 24**: foi criado o elemento **<h1>** que será modificado, e se trada de uma caixa.

**Linha 9 e 10**: configuram o size (altura e largura).

**Linha 12 e 14**: configuram a borda.

**Linha 15**: configura uma borda interna (preenchimento)

**Linha 16 a 18**:vai usar parte da margem para criar um contorno.

**Linha 19**: cria uma margem para o lado de fora.

**TAMANHO TOTAL**: para calcular a largura e altura total de um elemento na tela, some os tamanhos do **conteúdo + preenchimento + borda + margem**. O contorno não vai entrar nessa conta, pois utiliza parte da medida da margem.



Olhando de perto, podemos analisar as medidas configuradas no código apresentado. As medidas de height e width (300x50) são medidas apenas pela parte pontilhada do conteúdo.

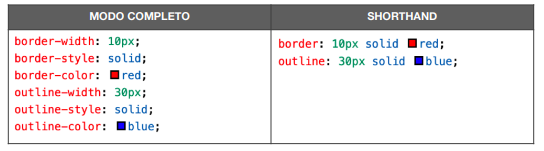
A border de 10px ficou em vermelho e o outline de 30px ficou em azul. O padding de 20px fica da borda para dentro e a margin de 50px fica da borda para fora.

Sendo assim, a medida total que essa caixa vai ocupar é de 50 + 10 + 20 + 300 + 20 + 10 + 50 = **460px de largura** e 50 + 10 + 20 + 50 + 20 + 10 + 50 = **210px de altura**.

**NOVIDADE DAS CSS3**: Existe a nova propriedade **box-sizing** onde podemos definir que as dimensões height e width não são medidas apenas a partir do **conteúdo** (**content-box**) e sim pela **borda** (**border-box**).

**Dá para simplificar?**

As configurações de borda e contorno também possuem ***shorthands***, para simplificar o código.

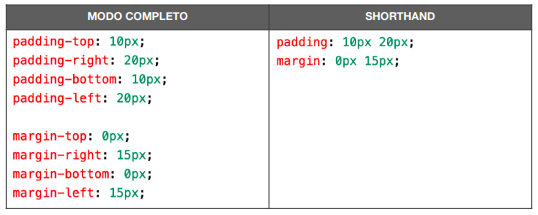


**Preenchimento e margem personalizados**

Todo elemento de caixa possui quatro valores para padding e quatro para margin, sempre nessa mesma ordem: superior (**top**), direita (**right**), inferior (**bottom**), esquerda (**left**).



Ou apenas para representar duas medidas



**Margens no automático**

Para centralizar um bloco independentemente do tamanho da janela vamos usar a declaração:



Assim todo bloco vai centralizar automaticamente ao centro da janela.

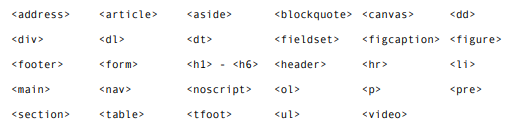
**TIPOS DE CAIXAS**

**Caixas do tipo BLOCK-LEVEL**

São do tipo que ocupam 100% da largura total do elemento, e se não estiver em nenhuma outra caixa, vai ocupar a largura total de <body>.

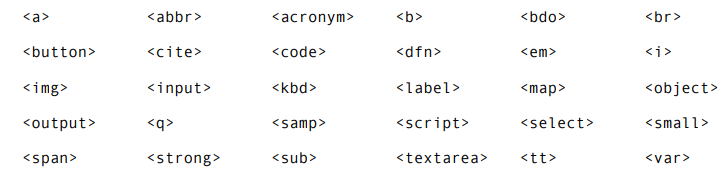
O elemento **block-level** mais conhecido é <div>, e também tem suas variações semânticas vindas graças ao **HTML5**: **<main>**, **<section>**, **<aside>**, etc

Outros elementos **HTML5** **block-level**:



**Caixas do tipo INLINE-LEVEL**

Esse elemento vai começar exatamente do ponto onde ele foi inserido, e sua largura vai ocupar apenas o espaço relativo ao seu conteúdo.



**Grouping Tag’s e Semantic Tag’s**

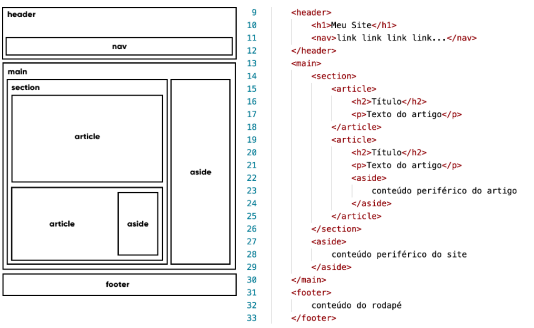
A linguagem **HTML** antiga tinha por padrão apenas duas formas de agrupamento **<div>** e **<span>**. Que a única diferença entre elas era que uma é um elemento agrupador do tipo ***block-level*** e a outra é ***inline-level***.

Com a chegada do **HTML5** chegaram as tag’s semânticas de agrupamento, não que **<div>** e **<span>** deixaram de existir ou ficaram obsoleto, mas que atualmente já não faz tanto sentido usar pois são tag’s não semânticas.

* **HEADER**: cria áreas relativas a cabeçalhos. Normalmente inclui títulos **<h1>** - **<h6>** e subtítulos. Podemos também criar menus de navegação.
* **NAV**: define uma área que possui os links de navegação. Um **<nav>** pode estar dentro de um **<header>**.
* **MAIN**: é um agrupador para delimitar o conteúdo. Normalmente concentra as seções, artigos e conteúdos periféricos.
* **SECTION**: cria seções para a sua página. ela pode contar o conteúdo diretamente no seu corpo ou dividir os conteúdos em artigos com conteúdo específicos, “uma seção é uma agrupamento temático, tipicamente com um cabeçalho”.
* **ARTICLE**: é um elemento que vai conter um conteúdo que pode ser lido de forma independente e dizem respeito a um mesmo assunto, podemos usar um **<article>** para delimitar um post de blog ou fórum.
* **ASIDE**: delimita um conteúdo periférico e complementar. Normalmente um conteúdo **<aside>** está posicionado ao lado de um determinado texto.

**MÚLTIPLOS** **NÍVEIS**: A sua criatividade e planejamento vai definir a estrutura do seu site. Sendo assim, é possível ter um ou mais dentro de uma ou até mesmo criar dentro de um . Não existem limitações quanto a isso.

* **FOOTER**: cria um rodapé para o site inteiro, seção ou artigo, é um conteúdo que não faz parte diretamente do conteúdo nem é um conteúdo periférico (o que caracterizaria um **<aside>**), mas possui informações sobre autoria do conteúdo, links adicionais, mapa do site, documentos relacionados.

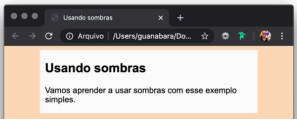


**Sombra nas caixas**

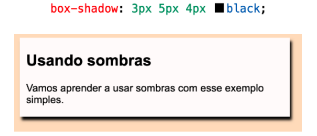
As sombras são muito uteis para dar volume, para entender vamos criar exemplos:



LINHA 22: temos apenas um <article> com um breve conteúdo



Linha 17: adicionamos uma declaração especial para adicionar sombra.



Veja que uma sombra é adicionada assim que declaramos box-shadow junto dos seus 4 valores:

1. **Deslocamento horizontal** (h-offset): quanto a sombra vai andar para o lado direito (valores negativos causam deslocamento para a esquerda)
2. **Deslocamento vertical** (v-offset): quanto a sombra vai andar para baixo (valores negativos causam deslocamento para cima)
3. **Embaçamento** (blur): quanto a sombra vai se espalhar pelo fundo
4. **Cor** (color): cor da sombra. É possível usar transparência.

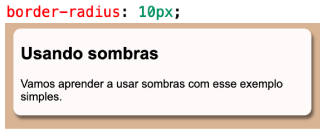
**MUITO CUIDADO**! Não exagere no uso de sombras, pois elas podem tornar o seu efeito visual muito pesado. Evite também usar sombras coloridas. Olhe ao seu redor e perceba que as sombras são sempre pretas. Use cores rgba() para obter uma transparência que cause efeitos mais suaves.

**BORDAS DECORADAS**

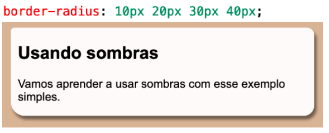
As bordas nem sempre precisam ser iguais, podem ter alguns detalhes especiais.

**Vértices arredondadas**

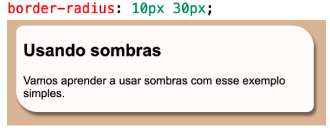
Podemos declarar border-radius para que os cantos se arredondem, entendendo assim que as bordas fiquem com formatos diferentes:



Todas as bordas tem o mesmo valor, e ficam com o mesmo formato.



Aqui podemos notar que demos um valor diferente para cada vértice, e analise que cada uma delas tem um formato diferente se analisarmos o valor declarado.



Podemos declarar dois valões, mas sempre vai começar da esquerda cima e intercalando com a parte inferior, que iria começar da direita.

**TENHO DESAFIOS PRA VOCÊ**

No material apoio tente resolver o desafio **d010**. Evite copiar códigos e tente fazer tudo sozinho.