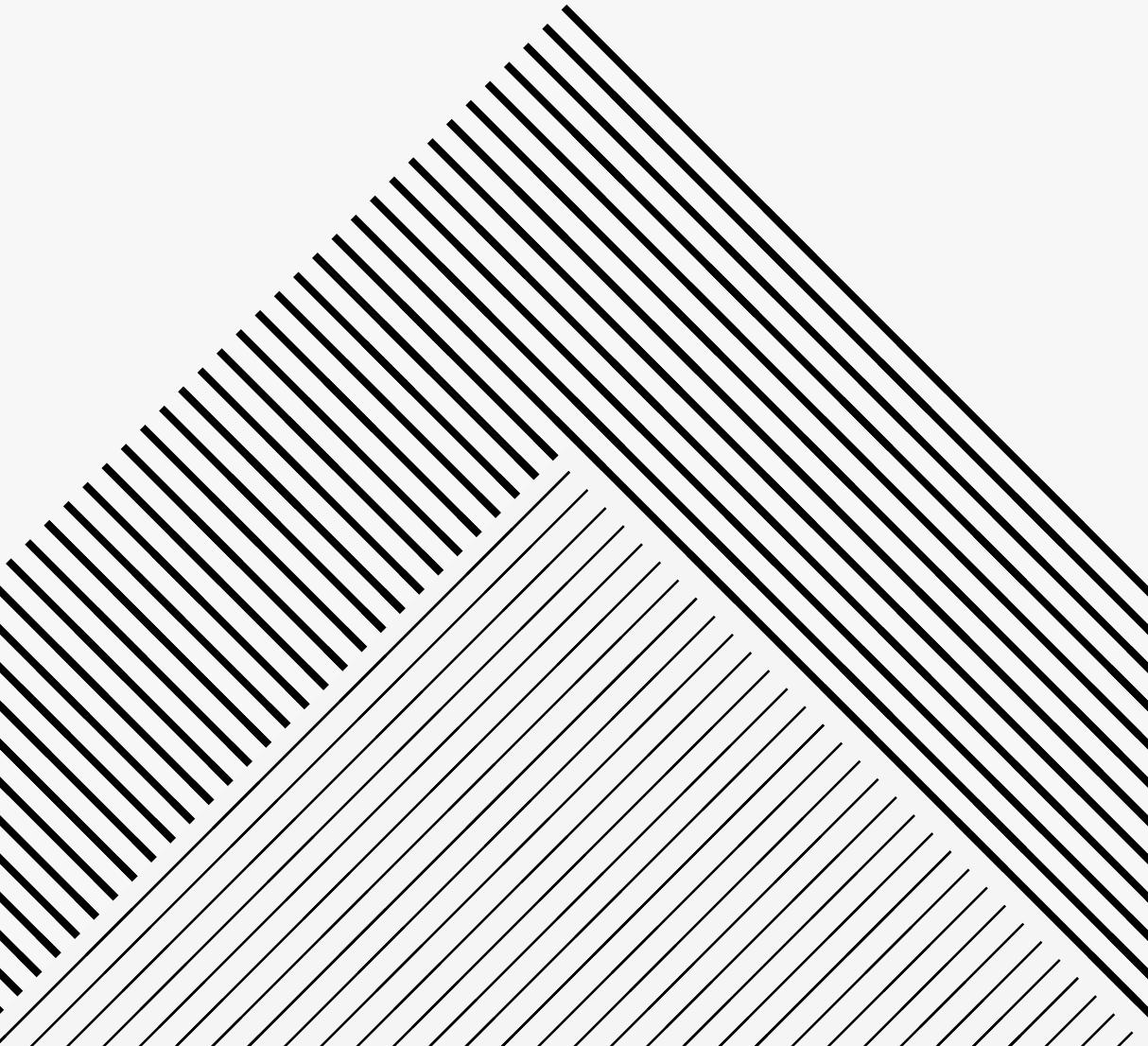


*Ebac*



→ Introdução ao web Design Responsivo.

28/09  
Módulo II

→ Layout Fluido e medijs Queries.

→ mobile First e Unidades Relativas.

→ Flexbox e CSS Grid.

Video 1

→ Introdução ao Design Responsivo -

\* O que é?

- Vai além da simples adaptação de layout.

- Responde às características do dispositivo para proporçãoizar a melhor experiência possível.

- Se aplica a diferentes contextos: Celulares, Tablets, desktops e até leitores de tela.

→ Como criar?

- Layout fluido

- medijs Queries

- Medijs e imagem flexíveis.

## → Layout Fluido:

- Utiliza unidades de medidas relativas no CSS, como porcentagens (%), em, rem, em vez de unidades absolutas como pixels (px).

Ex: main {

    width: 80% /\* Em relação ao tamanho da tela

    font-size: 1.5em /\* O tamanho do texto será proporcional  
                        ao elemento pai.

## → Media Queries

- Permite definir estilos diferentes para diferentes tamanhos de tela.

- São a base do design responsivo moderno.

Ex: @media (max-width: 768px) {

    body {

        background-color: lightgray;

}

OBS: "Se" variar da tela e = ou < 768px, ele aplica o modelo do body.

## → Mídia e imagem Flexíveis:

- Elementos de mídia (imagens, vídeos, áudios e widgets) precisam se ajustar dinamicamente para evitar que fiquem cortados ou distorcidos

img {

max-width: 100% ; /\* Em relação ao container principal.  
height: auto ; /\* Mantém a proporção original.

## → Mobile-First

- É uma abordagem onde o design e desen. começa pelo dispositivo móvel.
- É uma resposta ao crescimento do uso de smartphones.

## ↳ Priorização de Conteúdo

- Em telas menores, o mais importante vem primeiro.

## ↳ Performance Optimizada

- Reduzir o tempo de carregamento é essencial.
- Técnicas como compressão de imagem e lazy loading ajudam na performance.

## ↳ Interações touch-friendly

- Elementos clicáveis devem ser **grandes e bem espacados** para facilitar o toque.

## ↳ Contexto de Uso

- Usuários podem estar em movimento, então o design deve ser **simples e eficiente**.

## → Layouts Fluidos com Unidades Relativas.

Vídeo 2  
Módulo 10

- Obs: A tag `meta name="viewport"`, informa ao Navegador que o layout deve ser ajustado ao tamanho real da tela do dispositivo.

## → Princípios Unidades Relativas:

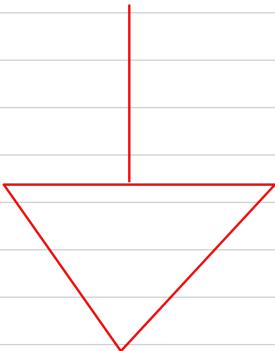
Unidade	Como funciona	Exemplo	Melhor para...
%	Baseado no tamanho do elemento pai	<code>width: 80%;</code>	Tamanhos de elementos fluidos
rem	Baseado no tamanho do root (html)	<code>font-size: 2rem;</code> <i>base: 16px.</i>	Escalabilidade uniforme
em	Baseado no tamanho do elemento pai	<code>padding: 1.5em;</code>	Layouts modulares
vw vh	Baseado na largura/altura da tela	<code>height: 100vh;</code>	Layouts de tela cheia

## ↳ Imagens Responsivas:

- Em dispositivos menores, carregar imagens grandes **afecta desempenho**.
- Usar imagens menores **economiza dados móveis**.

```
<img  
    <!-- Imagem padrão (fallback caso o navegador não suporte srcset) --&gt;<br/>    src="imagem-padrao.jpg"  
    <!-- SRCSET: Define versões alternativas da imagem + suas larguras --&gt;<br/>    srcset="imagem-pequena.jpg 600w,      <!-- Versão para telas até ~600px --&gt;<br/>          imagem-media.jpg 1200w,        <!-- Versão para telas até ~1200px --&gt;<br/>          imagem-grande.jpg 1800w"       <!-- Versão para telas grandes --&gt;<br/>    <!-- SIZES: Define quanto espaço a imagem ocupa em diferentes breakpoints --&gt;<br/>    sizes="(max-width: 600px) 100vw,     <!-- Em telas pequenas: ocupa 100% da largura da tela --&gt;<br/>          (max-width: 1200px) 50vw,      <!-- Em telas médias: ocupa 50% da largura --&gt;<br/>          33vw"                      <!-- Em telas grandes: ocupa 33% da largura --&gt;<br/>  
>
```

Video 3  
FlexBox



## → Flexbox (Flexible Box Layout)

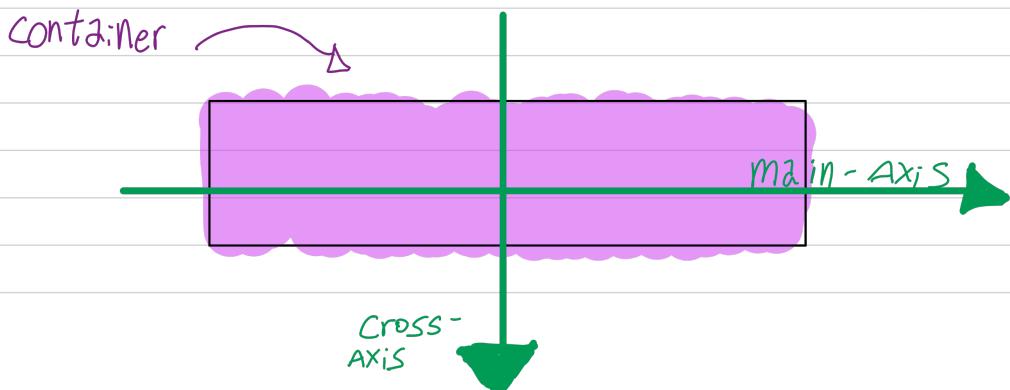
- Um Modelo de layout CSS que organiza elementos de forma eficiente, permitindo controle total sobre alinhamento, espaçamento e distribuição.

### ↳ Vantagens:

- Simples de usar, poucas regras.
- Perfeito para alinhamento horizontal e vertical.
- Facilita a distribuição de espaço entre elementos.
- Adapta-se auto a diferentes tamanhos de tela.

## → Eixos do flexbox

- Eixo principal (main-axis): Direção Principal dos itens.
- Eixo secundário (cross-axis): Perpendicular ao eixo principal.



→ CSS Grid Layout  
↳ Por que usar?

Vídeo  
9

- Float e position exigem muitas correções manuais
- O Flexbox é ótimo para uma dimensão (linhas ou colunas), mas não para as duas ao mesmo tempo.
- Layouts podem quebrar facilmente quando a tela é redimensionada.

↳ Definição:

- O CSS Grid Layout é um sistema de layout bidimensional que permite organizar elementos em linhas e colunas, facilitando a criação de páginas bem estruturadas.

↳ Vantagens:

- Organização intuitiva em linhas e colunas.
- Controle total sobre alinhamento e espaçamento.
- Layouts flexíveis e responsivos com poucas linhas de código.
- Redução de código em comparação com métodos antigos.

```
.container {  
    display: grid;  
    grid-template-columns: 200px 200px 200px; /* Três colunas fixas */  
    grid-template-rows: 100px 100px; /* Duas linhas fixas */  
}
```

## ↳ Valores dinâmicos

- Podemos usar frações (Fr) para distribuir espaço auto.

```
.container {  
    display: grid;  
    grid-template-columns: 1fr 2fr 1fr; /* A segunda coluna será duas vezes maior */  
    grid-template-rows: 1fr 1fr 1fr;  
    min-height: 100vh;  
}
```

## ↳ Definir a área de um elemento:

- Posicionamos dentro do container com **grid-column** e **grid-row**.

```
/* CSS */  
  
header {  
    grid-column: 1 / 4; /* Ocupa da primeira até a terceira coluna */  
    grid-row: 1 / 2; /* Ocupa apenas a primeira linha */  
    text-align: center;  
}
```

row	column 1fr	2fr	1fr
1fr	Header	Header	Header
1fr			
1fr			

↳ linhas

• Podemos também definir a utilização do espaço com a atribuição **grid-template-areas**.

```
<!-- HTML -->
...
<body>
  <div class="container">
    <header>
      <h1>Bem-vindo ao Meu Site</h1>
    </header>
    <main>
      <h2>Sobre Nós</h2>
      <p>Somos uma empresa dedicada a fornecer...</p>
      <p>Nosso compromisso é entregar qualidade...</p>
    </main>
  </div>
</body>
```

```
/* CSS */
.container {
  display: grid;
  grid-template-areas:
    "header header header header"
    "sidebar main main aside"
    "footer footer footer footer";
  min-height: 100vh;
}

header {
  grid-area: header;
  text-align: center;
}

main {
  grid-area: main;
  text-align: center;
```

