

MÓDULO

PROGRAMAÇÃO WEB – HTML/CSS

UNIDADE

LAYOUTS

ÍNDICE

OBJETIVOS.....	3
INTRODUÇÃO.....	4
1. ESTRUTURA DE UM WEBSITE.....	5
1.1. ELEMENTOS DE LAYOUT HTML.....	6
2. TÉCNICAS DE LAYOUT	7
2.1. FRAMEWORK CSS.....	7
2.2. CSS FLOAT	8
2.3. CSS FLEXBOX.....	16
2.4. CSS GRID	24
CONCLUSÃO	33
AUTOAVALIAÇÃO	35
SOLUÇÕES.....	39
PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO.....	40
BIBLIOGRAFIA	41

OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Criar layouts de websites e compreender a sua estrutura e os elementos HTML que os compõem.
- Compreender as técnicas de layout.

INTRODUÇÃO



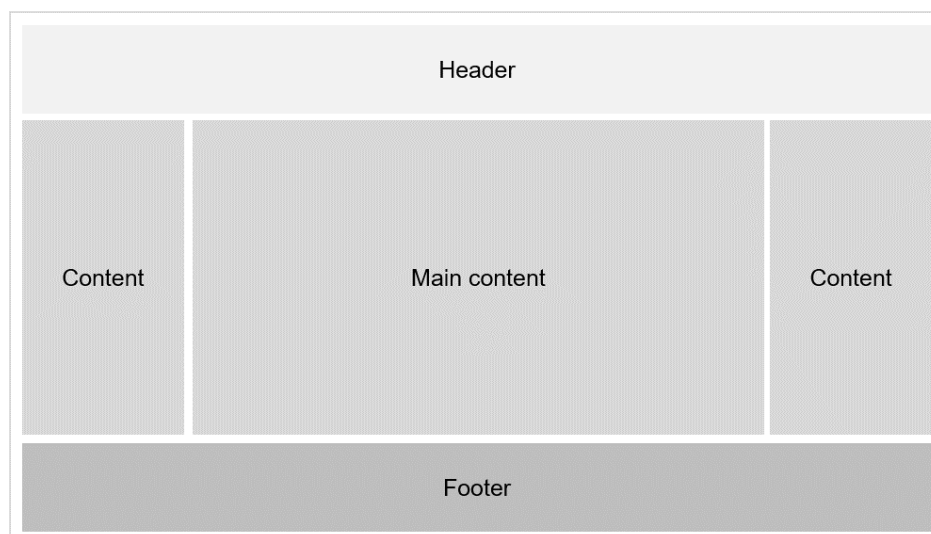
Hoje em dia, os websites são vistos como os cartões de visita de empresas, pessoas, negócios, etc. É o primeiro canal de exposição no mercado. Assim, a primeira impressão dada por um website é de extrema importância já que, se o visitante gostar do que viu, irá voltar e, muito provavelmente, recomendar.

Todas as informações contidas num website devem, por isso, ser claramente expostas e de forma agradável e atrativa. Os layouts dos sites devem ser organizados de forma a que captem a atenção dos visitantes, sendo de extrema importância que os mesmos sejam funcionais e fáceis de navegar.

Um layout criativo e funcional é um dos pontos mais importantes para que um website possa exercer a sua função de maneira correta.

1. ESTRUTURA DE UM WEBSITE

A estrutura de um website é, geralmente, dividida em cabeçalho, menu, conteúdo e rodapé. Obviamente, a escolha do layout depende muito das preferências de cada um.



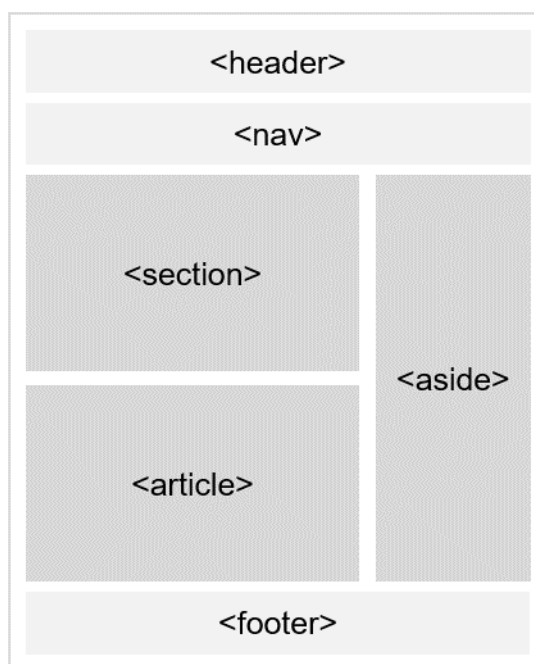
Estrutura de layout de um website.

Geralmente, os websites exibem o seu conteúdo em colunas, tal como acontece em jornais e revistas.

Dentro do conteúdo podem surgir vários elementos HTML, tais como elementos section, article, aside, entre outros.

1.1. ELEMENTOS DE LAYOUT HTML

O HTML tem diversos elementos que definem diferentes partes de uma página web. A figura abaixo mostra um esquema destes elementos e a sua normal disposição na página.



Esquema de disposição de elementos numa página web.

Todos estes elementos deverão estar contidos no <body> do documento HTML.

De seguida, damos uma breve explicação de cada um dos elementos:

- <header>: define o cabeçalho de um documento ou de uma secção.
- <nav>: define um conjunto de links de navegação.
- <section>: define uma secção de um documento.
- <article>: define um conteúdo independente e autocontido.
- <aside>: define o conteúdo fora do conteúdo em que está contido (como uma barra lateral).
- <footer>: define um rodapé de um documento ou de uma secção.

2. TÉCNICAS DE LAYOUT

Para criar layouts de várias colunas pode recorrer-se a uma de quatro técnicas diferentes, cada uma com as suas próprias características:

- Framework CSS.
- CSS float.
- CSS flexbox.
- CSS grid.

Nos pontos seguintes desta unidade vamos dar exemplos de como cada uma destas técnicas funciona.

2.1. FRAMEWORK CSS



Desenvolvimento

Frameworks CSS são conjuntos de componentes que provêm uma estrutura básica de elementos reutilizáveis, tendo uma arquitetura consistente de funcionalidade genérica sob a qual a aplicação será construída.


São utilizados principalmente para construção de páginas de websites e de sistema para Internet.

Fonte: Wikipedia

As frameworks são usadas quando se pretende criar layouts de forma fácil e rápida. Exemplos de frameworks são o W3.CSS e o Bootstrap. Mais à frente, neste módulo, abordar-se-á com mais profundidade o Bootstrap.

Estas bibliotecas dão estrutura à página web, e servem para construir layouts de forma rápida e simples. Estas estruturas são, por defeito, responsivas.

Geralmente são compostas por arquivos CSS e JS.


Desenvolvimento

Para saber mais sobre W3.CSS, visite a página W3Schools:

<https://www.w3schools.com/w3css/default.asp>

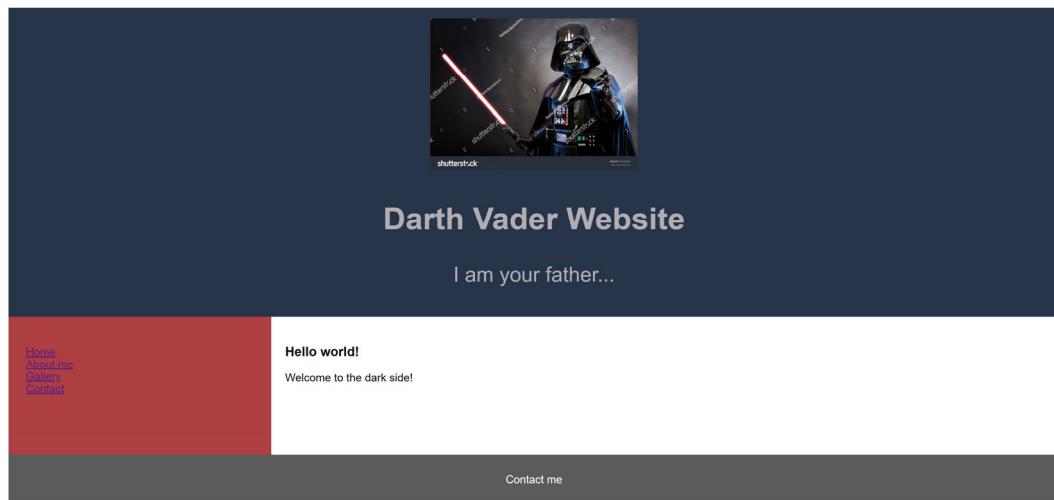
2.2. CSS FLOAT

É muito comum usar as propriedades float e clear do CSS quando se pretende construir layouts de websites. Esta técnica apresenta como desvantagem o facto de os elementos flutuantes estarem ligados ao fluxo do documento o que pode condicionar a flexibilidade da página.

Vejamos o exemplo de um layout feito através do elemento float.

Exemplo

Imagine que pretende construir um website semelhante ao da figura abaixo. Este site é constituído por um cabeçalho <header>, o conteúdo <section> é composto por duas colunas, uma com uma barra de navegação <nav> e outra com um artigo <article> e, por fim, um rodapé <footer>.



Cada elemento foi definido da seguinte forma:

■ <header>:

```
<header>

    <h2>Darth Vader Website</h2>

    <p>I am your father...</p>

</header>
```

■ <section>:

```
<section>

    <nav>

        <ul>

            <li><a href="#">Home</a></li>

            <li><a href="#">About me</a></li>

            <li><a href="#">Gallery</a></li>

            <li><a href="#">Contact</a></li>

        </ul>

    </nav>

</section>
```

```
</nav>

<article>

    <h1>Hello world!</h1>

    <p>Welcome to the dark side!</p>

</article>

</section>
```

■ <footer>:

```
<footer>

    <p>Contact me</p>

</footer>
```

Passamos agora a explicar as folhas de estilo utilizadas como forma de construção do layout pretendido.

```
* {

    box-sizing: border-box;

}
```

A propriedade `box-sizing` define como a largura e altura dos elementos são calculadas, e se incluem ou não o preenchimento das bordas. Neste caso definiremos a `box-sizing` com o atributo `border-box`, o que irá resultar na inclusão do `padding` e `border` nas altura e largura do elemento.

De lembrar que o seletor `*` seleciona todos os elementos do documento.

Para que sejam criadas duas colunas uma ao lado da outra (os elementos `nav` e `article` inseridos na `section`), temos de lhes atribuir a propriedade `float:left`.

```
nav {  
    float: left;  
    width: 25%;  
    height: 200px;  
    background: #ae3f41;  
    padding: 20px;  
}  
  
article {  
    float: left;  
    padding: 20px;  
    width: 75%;  
    background-color: #ffffff;  
    height: 200px;  
}
```

Para garantirmos que nada flutua ao lado das duas colunas da <section>, teremos de atribuir um pseudo-selector :after à <section>, que garanta que nada flutue após esta secção. Assim:

```
section:after {  
    content: "";  
    display: table;  
    clear: both;  
}
```

Neste caso, incluímos ainda uma nova propriedade que vai fazer com que as colunas apareçam uma por cima da outra em screens mais pequenos, tornando, assim, o layout responsivo. Isto acontece com a introdução da propriedade `@media` (`max-width: 600px`). Mais à frente neste módulo, iremos abordar este assunto com maior profundidade.

```
@media (max-width: 600px) {  
  nav, article {  
    width: 100%;  
    height: auto;  
  }  
}
```

Esta estilização faz com que, para screens até 600px (e apenas nesta condição), as duas colunas (nav e article) ocupem 100% da largura do screen, sendo a sua altura calculada automaticamente. Retirámos assim a condição de flutuarem uma ao lado da outra.

O código final compilado será:

```
<!DOCTYPE html>  
<html lang="pt">  
<head>  
  <title>Layout Examples</title>  
  <meta charset="UTF-8">  
  <meta name="description" content="Layout examples">  
  <meta name="author" content="Sara Granja">  
<style>  
  * {  
    box-sizing: border-box;
```

```
}  
  
body {  
    font-family: Arial, Helvetica, sans-serif;  
}  
  
/* Estilizar o header */  
header {  
    background-color: #26354a;  
    padding: 15px;  
    text-align: center;  
    font-size: 30px;  
    color: #b4aeb2;  
}  
  
/* Criar duas colunas/caixas que flutuam uma ao lado da outra*/  
nav {  
    float: left;  
    width: 25%;  
    height: 200px;  
    background: #ae3f41;  
    padding: 20px;  
}  
  
/* Estilização dos links do menu */  
nav ul {  
    list-style-type: none;  
    padding: 5px;  
}  
  
article {  
    float: left;
```

```
padding: 20px;

width: 75%;

background-color: #ffffff;

height: 200px;
}

/* Apagar informação após as colunas*/
section:after {
    content: "";
    display: table;
    clear: both;
}

/* Estilização do footer */
footer {
    background-color: #5a5c5b;
    padding: 10px;
    text-align: center;
    color: white;
}

/* Responsive layout - faz com que as duas colunas, em screens
menores de 600px, apareçam uma em cima da outra*/
@media (max-width: 600px) {
    nav, article {
        width: 100%;
        height: auto;
    }
}

</style>
</head>
```



```
<body>

  <header>

    <h2>Darth Vader Website</h2>

    <p>I am your father...</p>

  </header>

  <section>

    <nav>

      <ul>

        <li><a href="#">Home</a></li>

        <li><a href="#">About me</a></li>

        <li><a href="#">Gallery</a></li>

        <li><a href="#">Contact</a></li>

      </ul>

    </nav>

    <article>

      <h1>Hello world!</h1>

      <p>Welcome to the dark side!</p>

    </article>

  </section>

  <footer>

    <p>Contact me</p>

  </footer>

</body>

</html>
```

2.3. CSS FLEXBOX

Flexbox é um módulo CS3 que ajuda os elementos a comportarem-se de forma mais previsível em diferentes ecrãs de dispositivos.

É composto por duas partes elementares, a primeira seria o elemento pai ou o container, que contém os elementos que se pretendem mostrar e, em seguida, o próprio conteúdo. Para tornar um container flexível, basta usar a classe `display: flex`.

Exemplo

```
.flex-container {  
  display: flex;  
}  
  
<div class="flex-container">  
  <div>1</div>  
  <div>2</div>  
  <div>3</div>  
</div>
```

Existem diferentes propriedades de flex, sendo que as mais comuns são:

- **flex-direction:**

Permite escolher a direção em que as caixas são colocadas:

- ☐ `flex-direction: column` (de cima para baixo).
- ☐ `flex-direction: column-reverse` (de baixo para cima).

- ❑ `flex-direction: row` (horizontal, da esquerda para a direita).
- ❑ `flex-direction: row-reverse` (horizontal, da direita para a esquerda).

■ **flex-wrap:**

Propriedade que permite acumulação:

- ❑ `flex-wrap: wrap`; (se não couberem, acumulam).
- ❑ `flex-wrap: nowrap`; (valor por defeito, não acumulam).
- ❑ `flex-wrap: wrap-reverse`; (wrap inverso).

■ **flex-flow:**

É uma combinação entre `flex-wrap` e `direction`:

- ❑ `flex-flow: row wrap`;

■ **justify-content:**

Permite a distribuição do conteúdo:

- ❑ `justify-content: center`; (conteúdo centrado).
- ❑ `justify-content: flex-start`; (começa a ser colocado no início do container).
- ❑ `justify-content: flex-end`; (começa a ser colocado no final do container).
- ❑ `justify-content: space-around`; (separa o conteúdo deixando espaços antes, entre e depois).
- ❑ `justify-content: space-between`; (separa os conteúdos deixando espaço entre eles).

■ **align-items:**

Permite a distribuição vertical do conteúdo:

- ❑ `align-items: center`; (coloca os conteúdos ao centro).
- ❑ `align-items: flex-start`; (coloca o conteúdo na parte superior do container).
- ❑ `align-items: flex-end`; (coloca o conteúdo no final do container).
- ❑ `align-items: stretch`; (estica o conteúdo para que o mesmo ocupe todo o container).

Vejamos o exemplo do ponto anterior, usando agora a propriedade `flex`.

Exemplo

As alterações serão:

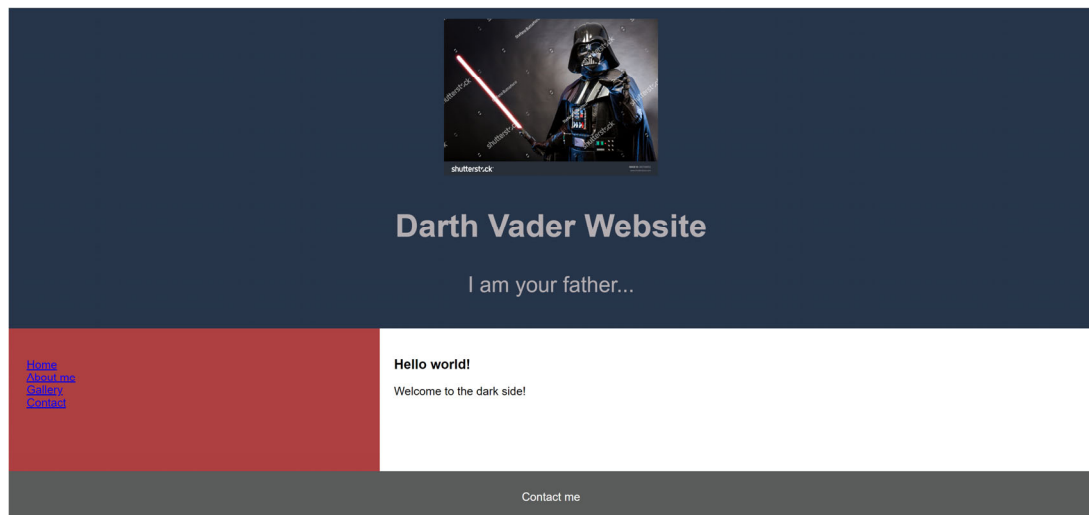
```
section {  
    display: -webkit-flex; /* pra ser suportado pelo Safari */  
    display: -ms-flex; /* pra ser suportado pelo IE */  
    display: flex;  
}
```

```
nav {  
    -webkit-flex: 1; /* pra ser suportado pelo Safari */  
    -ms-flex: 1; /* pra ser suportado pelo IE */  
    flex: 1;  
    background: #ae3f41;  
    padding: 20px;  
}
```

```
article {  
    -webkit-flex: 1; /* para Safari*/  
    -ms-flex: 1;  
    flex: 1;  
    padding: 20px;  
    width: 75%;  
    background-color: #ffffff;  
    height: 200px;  
}
```

```
@media (max-width: 600px) {  
  section {  
    -webkit-flex-direction: column;  
    flex-direction: column;  
  }  
}
```

O resultado será:



O código final compilado será:

```
<!DOCTYPE html>
<html lang="pt">
<head>
  <title>Layout Examples</title>

  <meta charset="UTF-8">
  <meta name="description" content="Layout examples">
  <meta name="author" content="Sara Granja">

<style>
* {
  box-sizing: border-box;
}

body {
  font-family: Arial, Helvetica, sans-serif;
}

header {
  background-color: #26354a;
  padding: 15px;
  text-align: center;
  font-size: 30px;
  color: #b4aeb2;
}

section {
```

```
display: -webkit-flex;
display: -ms-flex;
display: flex;
}

nav {
  -webkit-flex: 1;
  -ms-flex: 1;
  flex: 1;
  background: #ae3f41;
  padding: 20px;
}

nav ul {
  list-style-type: none;
  padding: 5px;
}

article {
  float: left;
  padding: 20px;
  width: 75%;
  background-color: #ffffff;
  height: 200px;
}

article {
  -webkit-flex: 2;
  -ms-flex: 2;
```

```
    flex: 2;
    padding: 20px;
    width: 75%;
    background-color: #ffffff;
    height: 200px;
}

footer {
    background-color: #5a5c5b;
    padding: 10px;
    text-align: center;
    color: white;
}

@media (max-width: 600px) {
    section {
        -webkit-flex-direction: column;
        flex-direction: column;
    }
}

</style>
</head>

<body>

    <header>

        <h2>Darth Vader Website</h2>
```



```
<p>I am your father...</p>

</header>

<section>
  <nav>
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">About me</a></li>
      <li><a href="#">Gallery</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </nav>

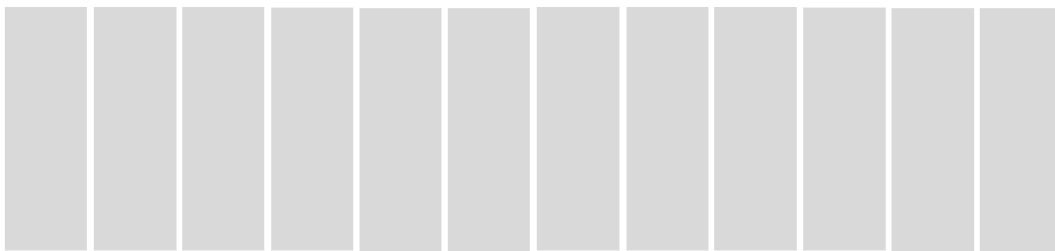
  <article>
    <h1>Hello world!</h1>
    <p>Welcome to the dark side!</p>
  </article>
</section>

<footer>
  <p>Contact me</p>
</footer>

</body>
</html>
```

2.4. CSS GRID

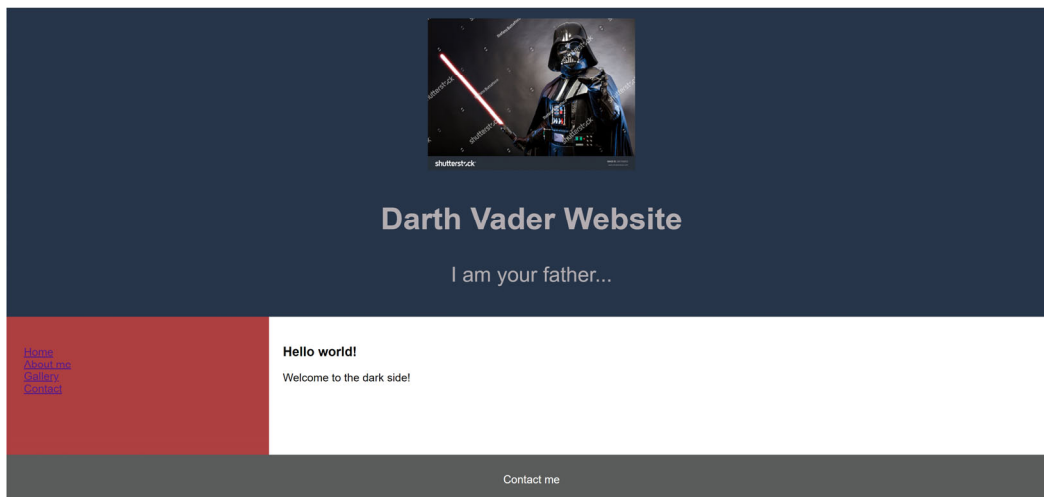
O módulo CSS Grid Layout é um sistema baseado em linhas e colunas, que distribui o conteúdo entre as mesmas utilizando classes, aplicando-lhes larguras segundo as necessidades. O CSS Grid baseia-se num layout que contém 12 colunas.



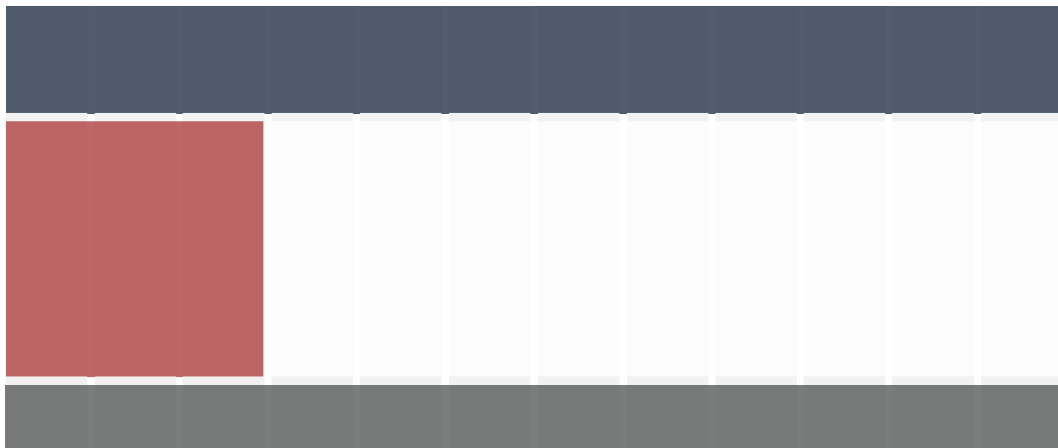
Esquema do módulo CSS Grid Layout.

Vejamos, de seguida, como aplicar este modelo ao exemplo dos pontos anteriores.

Exemplo



Se observar a imagem referente ao layout do exemplo, podemos concluir que esta página tem um layout semelhante ao da imagem abaixo.



Neste caso o header e o footer vão ocupar as 12 colunas do layout, o nav irá ocupar 3 das 12 colunas e o article ocupará as restantes 9 colunas.

Assim, em vez de definir as várias secções como temos estado a fazer até ao momento, iremos criar divs.

- Uma div de classe header onde estará o conteúdo do cabeçalho:

```
<div class="header">

    <h2>Darth Vader Website</h2>

    <p>I am your father...</p>

</div>
```

- Uma div de classe row onde estará o conteúdo do nav e do article. Sendo que o nav ocupa três colunas estará contido numa div de classe col-3 e, seguindo o mesmo raciocínio, o article estará contido numa div de classe col-9:

```
<div class="row">
  <div class="col-3">
    <nav>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">About me</a></li>
        <li><a href="#">Gallery</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>
  </div>
  <div class="col-9">
    <article>
      <h1>Hello world!</h1>
      <p>Welcome to the dark side!</p>
    </article>
  </div>
</div>
```

- Por último definimos uma div de classe footer, onde estará o conteúdo do cabeçalho:

```
<div class="footer">
  <p>Contact me</p>
</div>
```

Quanto à estilização, e não entrando em pormenor no que toca aos estilos de letra, cores, etc., temos de definir as classes col-. Tendo em conta que uma col-12 ocupará 100% da largura do layout, a mesma deve ser definida com width:100%. Uma col-6, ocupará uma width de 50%. Uma col-1, ocupará uma largura de 100%/12, ou seja, 8.33%, e assim sucessivamente.

É necessário, portanto, incluir na secção <style> as seguintes propriedades:

```
.col-1 {width: 8.33%;}  
.col-2 {width: 16.66%;}  
.col-3 {width: 25%;}  
.col-4 {width: 33.33%;}  
.col-5 {width: 41.66%;}  
.col-6 {width: 50%;}  
.col-7 {width: 58.33%;}  
.col-8 {width: 66.66%;}  
.col-9 {width: 75%;}  
.col-10 {width: 83.33%;}  
.col-11 {width: 91.66%;}  
.col-12 {width: 100%;}
```

Neste caso só irão ser usadas as col-3 e col-9, já que caso não se defina, o documento assume que a div ocupa 100% da largura.

Para que todas estas classes "col-" flutuem ao lado umas das outras, deverá aplicar a propriedade float:left a todas elas, o que se faz da seguinte forma:

```
[class*="col-"] {  
    float: left;  
}
```

Os restantes estilos serão os já usados nos pontos anteriores, sendo que apenas terá de ter em atenção o nome das divs e classes.

O código final compilado será:

```
<!DOCTYPE html>

<html lang="pt">

<head>

  <title>Layout Examples</title>

  <meta charset="UTF-8">

  <meta name="description" content="Layout examples">

  <meta name="author" content="Sara Granja">

<style>

* {

  box-sizing: border-box;

}

body {

  font-family: Arial, Helvetica, sans-serif;

}

.row::after {

  content: "";

  clear: both;

  display: table;

}

.header {

  background-color: #26354a;

  padding: 15px;

  text-align: center;
```

```
    font-size: 30px;

    color: #b4aeb2;
}

nav {

    background: #ae3f41;

    padding: 20px;
}

nav ul {

    list-style-type: none;

    padding: 5px;
}

article {

    padding: 20px;

    background-color: #ffffff;
}

.footer {

    background-color: #5a5c5b;

    padding: 10px;

    text-align: center;

    color: white;
}

[class*="col-"] {

    float: left;
}

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}
```

```
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}

</style>

</head>

<body>

    <div class="header">

        <h2>Darth Vader Website</h2>

        <p>I am your father...</p>

    </div>

    <div class="row">

        <div class="col-3">

            <nav>

                <ul>

                    <li><a href="#">Home</a></li>

                    <li><a href="#">About me</a></li>

                    <li><a href="#">Gallery</a></li>

                    <li><a href="#">Contact</a></li>

                </ul>

            </nav>

        </div>

    </div>

</body>

</html>
```



```
        </nav>

    </div>

    <div class="col-9">

        <article>

            <h1>Hello world!</h1>

            <p>Welcome to the dark side!</p>

        </article>

    </div>

</div>

<div class="footer">

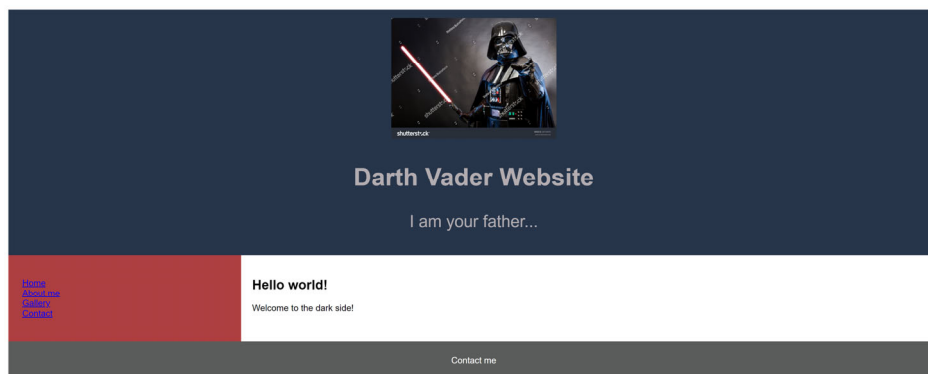
    <p>Contact me</p>

</div>

</body>

</html>
```

O resultado final será o esperado:



CONCLUSÃO

Esta unidade dedicou-se ao estudo e à compreensão dos layouts CSS e da estrutura de um website.

Aprofundou-se o conceito de `<body>`, analisando cada uma das secções que poderão estar nele contida, entre elas, o `<header>`, o `<nav>`, a `<section>`, o `<article>`, o `<aside>` e o `< footer>`.

Abordaram-se ainda as técnicas mais conhecidas de construção de layout em CSS, tais como Frameworks CSS, CSS Float, CSS Flexbox e CSS Grid.

Tendo em conta os tópicos lecionados, deverá agora ser capaz de usar qualquer uma das técnicas demonstradas com o fim de criar layouts de páginas web.

AUTOAVALIAÇÃO

1. Onde deverão estar incluídos os elementos `<header>`, `<section>` e `<footer>` de um documento HTML?
 - a) No `<body>` do documento HTML.
 - b) No `<head>` do documento HTML.
 - c) Em qualquer secção do documento HTML.
 - d) Logo após a tag `</body>` do documento HTML.

2. Qual é o elemento que define um cabeçalho de um documento HTML?
 - a) `<head>`.
 - b) `<cabeçalho>`.
 - c) `<body>`.
 - d) `<header>`.

3. O que são o W3.CSS e o Bootstrap?
 - a) São métodos de construção de layout pertencentes ao CSS float.
 - b) São Frameworks CSS.
 - c) São métodos de construção de layout pertencentes ao CSS Flexbox.
 - d) São métodos de construção de layout pertencentes ao CSS Grid.

4. Se incluir a propriedade `box-sizing: border-box` a um elemento HTML, a largura e altura deste elemento são calculadas tendo em conta:
- a) A largura e altura do elemento, incluindo o padding mas não as bordas.
 - b) A largura e altura do elemento, incluindo as bordas, mas não o padding.
 - c) A largura e altura do elemento, incluindo as bordas e o padding.
 - d) Apenas a largura do elemento, excluindo as bordas e padding.
5. Para que duas colunas inseridas numa secção fiquem ao lado uma da outra deverá ser atribuída que propriedade:
- a) `float: left`.
 - b) `float: side`.
 - c) `float: none`.
 - d) `float: side-by-side`.
6. Se pretender acumular flexboxes em forma de coluna, que propriedade deve ser usada?
- a) `flex-direction: row`.
 - b) `flex-direction: column`.
 - c) `justify-content`.
 - d) `flex: column`.
7. O layout em que se baseia o CSS Grid é composto por quantas colunas?
- a) 10 colunas.
 - b) 6 colunas.
 - c) É definido pelo web designer.
 - d) 12 colunas.

8. Se definir uma div com classe col-6, e tendo em conta o CSS Grid, qual deverá ser a largura da mesma?
- a) 100% da largura do screen.
 - b) 50% da largura do screen.
 - c) 25% da largura do screen.
 - d) 8.33% da largura do screen.
9. Uma div que ocupe 100% da largura da janela de visualização, e tendo por base o CSS Grid, terá que classe associada?
- a) col-1.
 - b) col-6.
 - c) col-12.
 - d) col-4.
10. Uma div que ocupe 75% da largura da janela de visualização, e tendo por base o CSS Grid, terá que classe associada?
- a) col-1.
 - b) col-6.
 - c) col-12.
 - d) col-9.

SOLUÇÕES

1.	a	2.	d	3.	b	4.	c	5.	a
6.	b	7.	d	8.	b	9.	c	10.	d

PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

Consulte a página da W3Schools, referente ao W3.CSS, para aprofundar o conhecimento desta framework.

- <https://www.w3schools.com/w3css/defaultT.asp>.

BIBLIOGRAFIA

- Willard, Wendy (2009), *HTML: A Beginner's Guide*. California: McGraw Hill.
- W3schools (2020), *W3.CSS Tutorial*. Disponível em: <https://www.w3schools.com/>. Consultado a 20 de dezembro de 2020.
- Wikipedia (2020), Frameworks CSS. Disponível em: https://pt.wikipedia.org/wiki/Frameworks_CSS. Consultado a 20 de dezembro de 2020.
- Imagens provenientes de Shutterstock.

