Apontamentos de Ferramentas Digitais II 18/19

David Sousa-Rodrigues[1]

[1] <david.s.rodrigues@ipleiria.pt>
<prof.david.esad@gmail.com>

20 de Março de 2019

Conteúdo

Apontamentos de Ferramentas Digitais II 2
Comunicação entre o docente e a turma 2
Conceitos básicos de algoritmia 3
B1. Conceito de Algoritmo 3
B2. Partes de um Algoritmo 3
B3. Representações de um Algoritmo - Fluxograma 3
Introdução ao desenvolvimento de páginas Web 7
Escrever código segundo as melhores práticas 8
Escrever código que seja compatível com os standards 8
Utilizar elementos semânticos 8
O modelo da caixa. 9
Display 9
Block ou inline? 9
Outros valores menos utilizados para o display 10
Responsive Web Design 11
Tipografia 12
Importar fontes não instaladas no computador do utilizador 13
Utilizando uma fonte importada 13
Formatos suportados 14
Criar Listas Ordenadas e Desordenadas 15
Listas não ordenadas (UL) 15
Listas ordenadas (OL) 15
Utilização de listas 16
Personalização de listas 16
Projecto do semestre 17
Web Portfólio Onepage 17
ENQUADRAMENTO 17
PROJECTO 17
CRITÉRIOS DE AVALIAÇÃO 17
SUPORTES A ENTREGAR 18
REFERÊNCIAS 18
DATA DE ENTREGA 18

Apontamentos de Ferramentas Digitais II

Estes apontamentos seguem aproximadamente os tópicos do programa da cadeira de Ferramentas Digitais II, não sendo exaustivos na cobertura de todo o programa. Esta primeira versão encontra-se bastante incompleta e pretende apenas dar um pequeno enquadramento ao aluno. Os ficheiros fonte destes apontamentos encontram-se online no repositório https://github.com/sixhat/FDii sendo que serão continuamente melhorados. O repositório terá assim sempre a versão mais actual deste documento.

Comunicação entre o docente e a turma

Para facilitar a comunicação entre todos, foi criado um grupo de discussão no google groups. O grupo é https://groups.google.com/d/forum/ferramentas-digitais-ii e o meu email de contacto utilizado será o prof.david.esad@gmail.com.

Conceitos básicos de algoritmia

No reino do mundo digital tudo o que utilizamos é executado por programas de computador. Como utilizadores destes programas muitas vezes não pensamos como é que eles são capazes de fazer certas coisas, como desenhar num ecrã ou verificar a ortografia de um texto. Por detrás destas tarefas está a noção de programa. Um programa é um conjunto de instruções agrupadas em tarefas específicas que ditam a forma de funcionar de um computador de forma a cumprir determinadas tarefas.

Um programa é como uma receita de culinária. Contém a lista de ingredientes e as instruções de como manipular esses ingredientes para obter o prato final.

B1. Conceito de Algoritmo

B2. Partes de um Algoritmo

Um algoritmo é composto por diferentes tipos de unidades funcionais. De entras podemos assinalar:

- Instruções
- variáveis
- instruções condicionais
- ciclos (loops).

B3. Representações de um Algoritmo - Fluxograma

A visualização de um algoritmo facilita a compreensão dos passos que o algoritmo executa. Para isso foram desenvolvidos símbolos gráficos que ilustram o comportamento de um algoritmo. Estes diagramas / desenhos técnicos são conhecidos como fluxogramas de execução e foram adotados como um standard ISO (https://www.iso.org/standard/11955.html) a partir de um standard americano ANSI prévio.

Exercício - Desenhar um Algoritmo com yEd.

Para demonstrar como podemos visualizar um algoritmo vamos utilizar um software de diagramação chamado yEd. O yEd é um software de produção de diagramas gratuito que está disponível para as diversas plataformas de computação (Windows, Mac OS X e linux). Pode ser obtido em https://www.yworks.com/products/yed.

A utilização de fluxogramas é de importância crucial para a compreensão de um programa de computador e para comunicar o funcionamento desejado de um processo ou algoritmo. São de tal forma importante que algumas linguagens de programação tentaram inclusive utilizar uma representação de fluxograma como a própria

linguagem de programação. Neste tipo de linguagem incluem-se **Flowgorithm** e **Raptor**.

A representação de um algoritmo computacional através de um fluxograma também não é a única forma de descrever um algoritmo. Outras representações existem, nomeadamente através da escrita do chamado pseudo-código, ou de utilização de outra linguagem visuais de modelação como **UML2**. No entanto os diagramas de fluxo de execução de um algoritmo, pela sua simplicidade e utilidade continuam a ser os diagramas mais utilizados.

Flowchart Symbol Cheat Sheet

Flowchart Symbol	Name (Alternates)	Description
	Process	An operation or action step.
	Terminator	A start or stop point in a process.
	Decision	A question or branch in the process.
	Delay	A waiting period.
	Predefined Process	A formally defined sub-process.
	Alternate Process	An alternate to the normal process step.
	Data (I/O)	Indicates data inputs and outputs to and from a process.
	Document	A document or report.
	Multi-Document	Same as Document, except, well, multiple documents.
	Preparation	A preparation or set-up process step.
	Display	A machine display.
	Manual Input	Manually input into a system.
	Manual Operation	A process step that isn't automated.
	Card	A old computer punch card.
	Punched Tape	An old computer punched tape input.
	Connector	A jump from one point to another.
	Off-Page Connector	Continuation onto another page.
	Transfer	Transfer of materials.
	Or	Logical OR
	Summing Junction	Logical AND
	Collate	Organizing data into a standard format or arrangement.
	Sort	Sorting of data into some pre-defined order.
	Merge (Storage)	Merge multiple processes into one. Also used to show raw material storage.
	Extract (Measurement) (Finished Goods)	Extract (split processes) or more commonly - a measurement or finished goods.
	Stored Data	A general data storage flowchart symbol.
	Magnetic Disk (Database)	A database.
	Direct Access Storage	Storage on a hard drive.
	Internal Storage	Data stored in memory.
	Sequential Access Storage (Magnetic Tape)	An old reel of tape.
	Callout	One of many callout symbols used to add comments to a flowchart
	Flow Line	Indicates the direction of flow for materials and/or information

Courtesy of BreezeTree Software - Makers of FlowBreeze Flow Chart add-In for Excel

Figura 1: Lista dos diversos diagramas existentes no standard de Fluxogramas.

Principais elementos para o desenho de fluxograms

Os principais elementos comuns a todos os fluxogramas são os que dizem respeito ao início e fim de um algoritmo, os que indicam instruções e os que indicam locais onde conforme uma dada condição, e cuja resposta é verdadeiro/falso, levará o algoritmo por caminhos de execução diferentes.

Princípio e fim do algoritmo

O princípio e fim de um algoritmo tem sempre que ser indicados por símbolos apropriados. Tradicionalmente o símbolo a utilizar é o observado na figura à esquerda, no entanto outras linguagens derivadas dos fluxogramas adoptaram outros símbolos. À direita os símbolos de start e stop utilizados pela linguagem de modelação UML2.

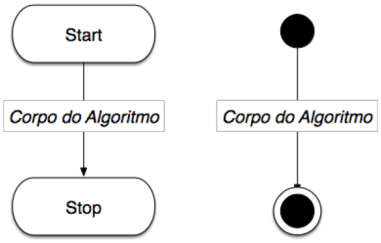
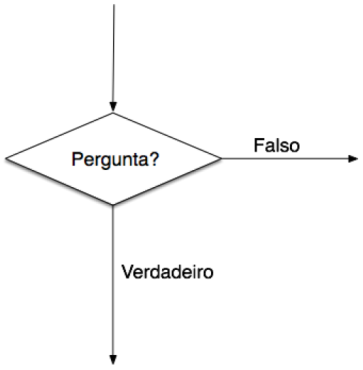


Figura 2: Esq: início e fim de um fluxogram. Dir: os mesmos elementos em UML2

Condições

Normalmente é comum pensar se uma condição desenhada por um losango pode dar respostas de uma escolha múltipla, eg: "Qual a sua cor favorita do arco-íris?" esta pergunta permite diversas respostas e seria plausível ter um losango com 7 setas de saída. No entanto convém notar que essas 7 opções podem ser substituídas por uma série de condições *verdadeiro/falso* do género:

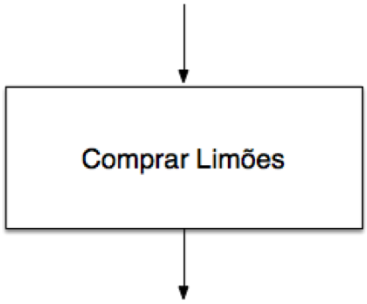
> A cor escolhida é azul? Se Verdadeiro seguir algoritmo, se Falso perguntar se a cor escolhida é amarelo? se Verdadeiro seguir algoritmo, se Falso perguntar se a core escolhida é .... e assim sucessivamente até se esgotarem as opções.



Esta forma de colocar as questões em termos de serem Verdadeiras ou Falsas parece complicar a tarefa de definir um algoritmo, mas permite utilizar elementos de programação muito simples e genéricos em termos de computação. Uma condição apenas possui dois estados. Ou é verdadeira, ou é falsa e esta simplicidade binária é que permite aos computadores construirem estruturas complexas a partir de elementos muito simples.

Instruções / Processos

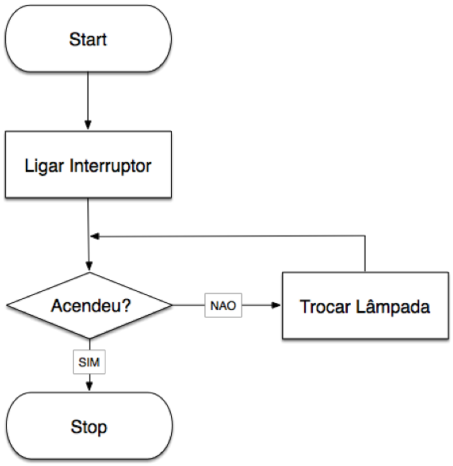
As instruções representam uma tarefa básica. São a unidade que executa algo e que após a sua conclusão retornam o controlo do algoritmo ao passo seguinte. As instruções são sempre executadas, não impõe condições para a sua execução e não terminam enquanto a tarefa nela indicada não estiver terminada.



Exemplo simples de um fluxograma:

Trocar uma lâmpada

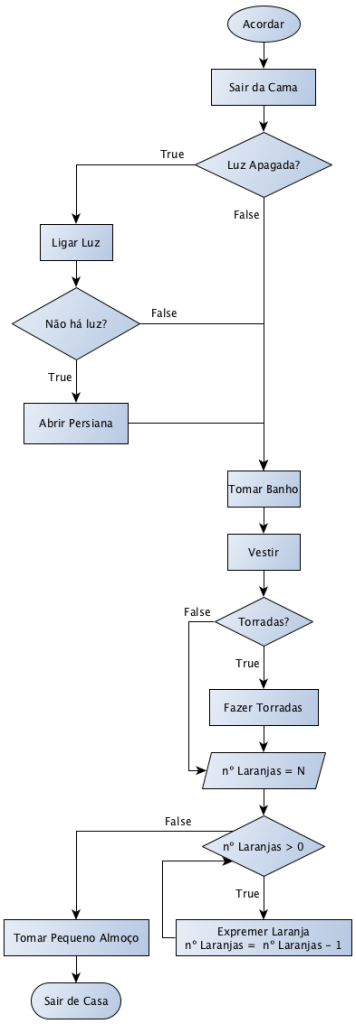
1. Ligar o interruptor
2. Acendeu?
3. Se sim continuar, caso contrário Trocar a lâmpada.
4. Fim.



Exercício: Desenhe um fluxograma para o algoritmo que executa todos os dias entre o momento que acorda e a saída de casa.

1. Acordar.
2. Sair da cama.
3. Se a luz estiver apagada, acendê-la. Se por algum motivo não houver luz, abrir persiana.
4. Ir tomar banho.
5. Vestir
6. Preparar o pequeno almoço para a família.
7. Perguntar se querem torradas. Se a resposta for afirmativa fazer torradas.
8. Fazer sumo de laranja. Enquanto tiver laranjas na fruteira esprema-as.
9. Tomar o pequeno almoço.
10. Sair de casa

Este exercício pode deve ser feito individualmente, e no final deve obter um diagrama de fluxo semelhante ao seguinte:



Introdução ao desenvolvimento de páginas Web

Nesta secção vamos aprender os fundamentos básicos do funcionamento da internet, e em particular de como são criadas as páginas Web que consultamos na WWW.

A WWW ou Web foi criada por Sir Tim Berners-Lee[2] quanto trabalhava no CERN na Suíça. A Web foi imaginada em 1989 como um sistema de gestão e partilha de informação. Na altura a informação digital era basicamente texto simples (sem formatações nem imagens) e surgiu então a primeira versão do HyperText Transfer Protocol (HTTP). O HTTP é a fundação para a comunicação de dados na Internet. O **hipertexto** é uma forma de adicionar informação relevante a um texto por forma a estabelecer ligações relevantes entre diferentes documentos. Estas ligações são normalmente designadas por hiperligações.

[2] Para mais informação em relação ao trabalho de Tim Berners-Lee consultar por exemplo https://en.wikipedia.org/wiki/Tim_Berners-Lee e http://info.cern.ch/Proposal.html

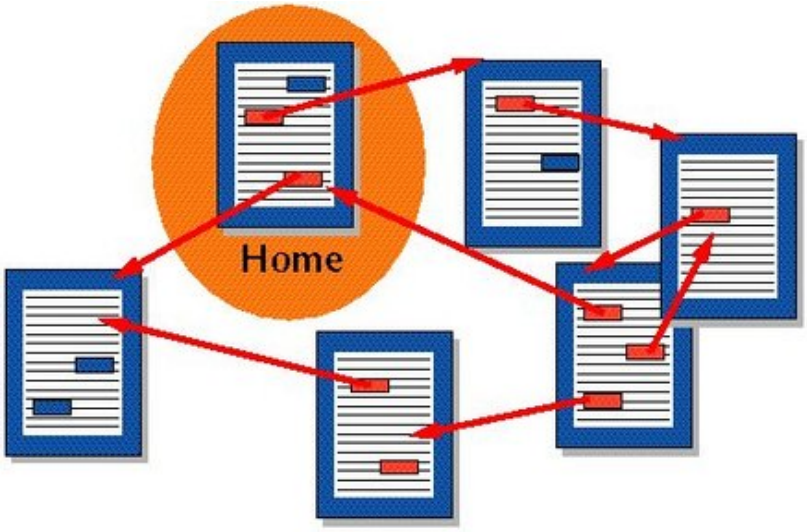


Figura 3: Exemplo de vários documentos de hipertexto conectados entre si por hipeligações. Imagem de Andreariverac, utilizada sob licença CC BY-SA 3.0 https://creativecommons.org/licenses/by-sa/3.0/deed.en

Escrever código segundo as melhores práticas

Ao longo das semanas anteriores vimos que é importante que o código `html` e `css` que escrevemos esteja correctamente estruturado. Intuitivamente a sua organização permite uma leitura e compreensão mais rápida e eficaz.

No entanto é necessário ter em atenção que a industria utiliza normalmente convenções que nem sempre são representadas nos *standards*. Essas práticas levam a que se tenha desenvolvido um conjunto de boas práticas cujas regras muitas vezes não estão codificadas em nenhum documento.

Aqui seguem-se alguns conselhos que devem ter quando estiverem a escrever código para os vossos projectos.

Escrever código que seja compatível com os standards

Muito embora os browsers modernos tenham alguma flexibilidade em interpretarem código que não esteja de acordo com o standards, tal não quer dizer que o venham a fazer no futuro. A escrita de código de acordo com os standards[3] é de primordial importância. Por exemplo:

[3] O último standard do HTML publicado é o 5.2 e pode ser encontrado online em https://www.w3.org/TR/html/

Errado:

```
<p id="intro">Este texto que <strong>vos deixo</p></strong>
<p id="intro">Não será assim muito extenso.
```

Correcto:

```
<p class="intro">Este texto que <strong>vos deixo<strong></p>
<p class="intro">Não será assim muito extenso.</p>
```

O primeiro caso tem diversos erros nomeadamente a utilização do `id` *intro* por multiplas vezes e o *nesting* errado do `p` e do `strong` no final da primeira linha. Na segunda linha o parágrafo não foi fechado com a *tag* correspondente pelo que também está errada. Com alta probabilidade os dois exemplos seriam renderizados pelo browser de forma semelhante. No entanto a primeira forma é claramente errada.

Utilizar elementos semânticos

O HTML 5 possui mais de 100 elementos diferentes. Apesar de parecerem muitos elementos, eles ajudam a dar significado à página web permitindo aos browsers inferirem o objetivo e importância de cada bloco de conteúdo. O HTML 5 introduziu novos elementos semânticos que facilitam e clarificam o que cada conteúdo é.

O modelo da caixa.

Já vimos como é que diversos elementos HTML podem ser utilizados para definir diversos tipos de conteúdos—cabeçalhos, parágrafos. No entanto ainda não discutimos como é que estes conteúdos são desenhados pelos browsers. Este capítulo vai tentar responder à questão *como é que o browser desenha o conteúdo html?*

Display

A propriedade **display**[4] do CSS permite controlar como os elementos HTML serão posicionados em relação aos seus pares e também definir como é que os elementos HTML seus descendentes se devem organizar.

[4] Mais informação sobre as diferentes propriedades do **display** pode ser obtida em https://developer.mozilla.org/en-US/docs/Web/CSS/display

- a propriedade **display** define duas qualidades dos elementos quando estão a gerar a caixa onde são desenhados.
  - **tipo de display externo** - ou seja, como é que o elemento se vai inserir no fluxo da página juntamente com os outros elementos.
  - **tipo de display interno** - ou seja, como os elementos filhos se vão distribuir.

Block ou inline?

O primeiro aspecto importante a considerar é que os elementos tem que ser colocados no layout da página juntamente com outros elementos. Este fluxo de elementos HTML tem que se ajustar segundo alguma regra. As duas principais alternativas são **block** e **inline**.

Os elementos em **block** vão ocupar a largura máxima possível do contentor onde estiverem inseridos. Os elementos **inline** pelo contrário vão ocupar apenas o espaço estritamente necessário para albergar o conteúdo. Exemplos de elementos **block** incluem o elemento html *DIV* que serve para criar um bloco estrutural de conteúdo. Por outro lado a tag *A* é um elemento cujo display é feito **inline** uma vez que a âncora estará por exemplo inserida num texto.

- **Block**: utilizados para conteúdos largos, como headings e elementos estruturais
- **Inline**: utilizados para pequenas quantidades de código, por exemplo para um enfatizar uma data ou um email.

Para definir esta propriedade em css basta modificar a propriedade **display**.[5]

[5] Exemplo prático disponível em https://codepen.io/sixhat/pen/NJLZRV

```
p {
  display: block;
}
```

```
p {
  display: inline;
}
```

Nestes exemplos o parágrafo *P* vais ser desenhado ocupando 100% do espaço disponível (**block**) ou utilizando o espaço estritamente necessário (**inline**).

De vez em quando pode-se também remover um elemento do ecrã. Isso também é possível de se fazer utilizando a propriedade **display** com o valor **none**.

```
p.hidden {
  display: none;
}
```

Neste caso o parágrafo da class `hidden` não será desenhado pelo browser e será removido do fluxo de elementos a desenhados—isto implica que não vai afectar o posicionamento dos restantes elementos.

Outros valores menos utilizados para o display

Para além dos valores **block**, **inline** e **none**, há também outros menos utilizados mas que convém ter presente na hora de programar uma página web[6] :

- **inline-block** - elementos formatados com este valor tem um comportamento misto entre inline e block. Enquanto os conteúdos interiores são formatados como um bloco o elemento em si formata-se com sendo inline. -Pode ser útil para fazer sistemas de navegação (menus).
- **list-item** - para fazer com que o elemento se comporte como uma lista com um sinal numa caixa de marca de lista.
- **table*** - vários valores, fazem com que o elemento se comporte como uma tabela

[6] A especificação dos valores para o display foi definida ao longo de diversas versões do CSS. Mais informação sobre o que cada versão do CSS adicionou pode ser encontrada em https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Display#Specifications

Responsive Web Design

A programação de páginas Web em HTML5 é standard e são muito raros os casos em que se utiliza ainda HTML4 ou anteriores. O HTML5 veio simplificar tremendamente o trabalho de desenvolvimento com a introdução de novos elementos semânticos (novas tags estruturais), assim como suporte para audio e vídeo. E mesmo nos casos em que os browsers mais antigos não suportam todos os elementos do HTML5 (leia-se Microsoft Internet Explorer 9 e anteriores) é possível servir um *polyfill*[7] aos browsers que lhes permite renderizar correctamente os elementos.

[7] **polyfill** é uma técnica que permite através de um programa de JavaScript simular o suporte para tags que browsers antigos não conseguem renderizar.

Tipografia

A tipografia assume um papel primordial na comunicação visual e isto é naturalmente também importante nas páginas Web. Apesar de a programação de uma página conter o texto num formato agnóstico, é possível definir todos os aspectos de tipografia que o browser deve utilizar para renderizar esse bloco de texto.

É no entanto possível não dar nenhuma indicação sobre as características tipográficas do texto da página web, sendo que nesses casos o browser irá assumir um conjunto de padrões e aplicá-los ao texto existente no HTML.

Para evitar isso é importante que o web designer especifique claramente qual o tipo de fonte que quer utilizar.

O standard define que qualquer fonte que esteja instalada no sistema operativo do utilizador possa ser utilizada, assim como qualquer fonte que possa ser descarregada em formato apropriado, aquando do carregamento da página web.

```
font-style
font-family
font
```

Para utilizar uma determinada fonte é necessário definir a família da fonte que se quer utilizar. No CSS isso é conseguido definindo a propriedade **font-family** que pode receber uma lista de fontes separadas por vírgulas e.g.

```
p {
  font-family: "Arial", "Roboto", sans-serif;
}
```

Neste exemplo os elementos do tipo `P` serão renderizados por uma de três fontes, “Arial”, “Roboto” ou a fonte sem serifa que o browser utilizar por defeito. A utilização será dada pela primeira fonte da lista que o browser conhecer, passando às opções seguintes sempre que as fontes mais à esquerda não estejam disponíveis.

No exemplo anterior é ainda possível ver que utilizamos um nome genérico para fontes sem serifa, o **sans-serif**. É possível definir famílias de fontes genéricas que os browsers interpretarão de acordo com os padrões definidos pelo programador. As famílias genéricas que todos os browsers suportam são o **sans-serif** anterior e ainda o **serif** para fontes serifadas, **cursive** para fontes cursivas que imitam a escrita humana, **fantasy** para fontes de fantasia e ilustração, e **monospace** para fontes monoespaçadas.

Normalmente a especificação das fontes a utilizar não necessita as aspas quando o nome da fonte não tem espaços no nome. No exemplo anterior Arial e Roboto podiam ter sido escritas sem as

aspas. No caso de utilizarmos fontes com espaços no nome então é necessário utilizar as aspas e.g. “Times New Roman”.

Importar fontes não instaladas no computador do utilizador

Muitas vezes o designer pretende utilizar certos tipos de letra que não estão disponíveis nos computadores dos utilizadores. Para essas situações o designer pode importar tipos de letra a partir de recursos externos. Para isso utiliza-se a regra CSS **@font-face**[8].

[8] **@font-face** devem ser declaradas imediatamente no topo do nosso ficheiro CSS de forma a poder utilizar a fonte no CSS.

```
@font-face {
  font-family: MyHelvetica;
  src: local("Helvetica Neue Bold"),
       local("HelveticaNeue-Bold"),
       url(MgOpenModernaBold.ttf);
  font-weight: bold;
}
```

Neste exemplo o comando `@font-face` é utilizado para definir um tipo de letra chamado “MyHelvetica” que vai tentar encontrar o tipo de letra no computador do utilizador utilizando o comando **local**. Caso o browser não encontre nenhum dos dois ficheiros definidos com o comando local, então vai utilizar um ficheiro online definido pelo comando **url**. Para além do mais este exemplo define que este tipo de letra será utilizado a negrito.

A utilização de tipos de letra externos é de grande utilidade para se desenharem websites que não fiquem restritos a tipos de fonte padrão ou aos instalados nos dispositivos dos clientes. Tem no entanto o inconveniente de obrigar a um maior tráfego de dados para o cliente e naturalmente custos extra para o servidor.

Também é necessário ter em linha de conta que a distribuição de tipos de letra pode não estar enquadrado no licenciamento do tipo de letra aquando da sua compra. Muitas vezes a “compra” licencia apenas a produção de material impresso e não a distribuição online. É necessário portanto ter algum cuidado com o licenciamento destes componentes digitais para evitar violações das legislações de propriedade intelectual.

Utilizando uma fonte importada

Para utilizar um tipo de letra importado com comando **@font-face** basta utilizar a propriedade **font-family** como anteriormente.

```
p {
  font-family: MyHelvetica;
}
```

Neste exemplo a fonte MyHelvetica carregada anteriormente é aplicada a todos os elementos do tipo parágrafo (`p`).

Formatos suportados

As fontes a utilizar podem estar em diversos formatos (otf, ttf, woff, woff2) e nem todos são suportados pelos browsers actuais. Assim é conveniente oferecer mais do que um formato na definição de **@font-face,** utilizando diversos elementos **url** (separados por vírgula).

```
src: url(ideal-sans-serif.woff) format("woff"),
     url(basic-sans-serif.ttf) format("opentype");
```

Para obter os ficheiros nos diversos formatos pode utilizar um serviço online como o Font Squirrel Generator[9].

[9] Font Squirrel Generator - https://www.fontsquirrel.com/tools/webfont-generator

Criar Listas Ordenadas e Desordenadas

Em HTML por vezes temos que criar listas *e.g.* lista de contactos numa aplicação de páginas brancas, ou a lista de hiperligações recomendadas para restaurantes num website de review de restaurantes, ou uma lista de outras notícias num website noticioso.

A criação de listas é relativamente simples e o HTML incorpora dois tipos de listas fundamentais: Listas não ordenadas (unordered) e ordenadas (ordered).

Listas não ordenadas (UL)

Estas listas são as mais comuns são produzidas pela tag `UL` (unordered list). Dentro dessa tag ul cada elemento da lista é indicado pela tag `li`.

```
<ul>
      <li>Elemento 1</li>
      <li>Elemento 2</li>
      <li>Elemento 3</li>
      <li>Elemento 4</li>
</ul>
```

Resulta em:

- Elemento 1
- Elemento 2
- Elemento 3
- Elemento 4

Listas ordenadas (OL)

Estas listas são menos comuns mas também importantes. São utilizadas quando se pretende que os elementos possuam um número de ordenação.

```
<ol>
      <li>Elemento 1</li>
      <li>Elemento 2</li>
      <li>Elemento 3</li>
      <li>Elemento 4</li>
</ol>
```

Resulta em:

1. Elemento 1
2. Elemento 2
3. Elemento 3
4. Elemento 4

Utilização de listas

As listas são muitas vezes utilizadas para criar sistemas de navegação com menus e submenus uma vez que podem ser colocadas listas dentro de listas (*nesting*) permitindo dessa forma criar hierarquias complexas.

Personalização de listas

Utilizando CSS é possível alterar consideravelmente a forma como as listas são mostradas pelos browsers.

Para isso é necessário definir a propriedade `list-style-type`

```
ul {
 list-style-type: circle;
}
```

vai produzir uma lista em que os marcadores são círculos:

- Elemento 1
- Elemento 2
- Elemento 3
- Elemento 4

A propriedade `list-style-type` aceita diversos tipos de elementos: `disc`, `circle`, `square`, `decimal`, `lower-roman`, `upper-roman`, `lower-greek`, `lower-latin` são apenas alguns dos elementos suportados para os marcadores dos items das listas.

Projecto do semestre

Web Portfólio Onepage

ESCOLA SUPERIOR DE ARTE E DESIGN
CALDAS DA RAINHA
Unidade Curricular: Ferramentas Digitais II
Ano letivo: 2018/2019
Curso: Design Gráfico e Multimédia
Ano: 1º ano
Semestre: 2º Semestre
Docentes: David Sousa-Rodrigues, Miguel Carradas

ENQUADRAMENTO

Quando pensamos em internet, provavelmente pensamos imediatamente em websites, mas hoje em

dia a internet é muito mais do que isso: conteúdos; aplicações; ferramentas; experiência; velocidade;

acessibilidade; partilha; poder; etc. Se a estes conceitos juntarmos a diversidade de devices com capacidade de output — computadores, telefones, tablets, relógios, etc — estaremos então a falar de um complexo ecossistema centrado no indivíduo enquanto utilizador.

Esta multiplicidade de suportes e meios apresenta desafios distintos e requer abordagens particulares. O projecto que se propõe de seguida, é potenciado e ao mesmo tempo condicionado pelas especificidades e possibilidades da internet e dos diferentes devices onde esta está presente.

PROJECTO

Desenvolver um portfólio baseado no conceito one page website, que deve ser visualizado em 3 versões (320px, 768px e 1200px), sendo assim ajustável a diferentes dimensões de ecrã. A programação deve ser compatível com as versões mais recentes dos seguintes browsers: Chrome, Firefox, Safari, Opera e IE.

O projecto deve ter pelo menos as seguintes secções de conteúdos: Sobre, Trabalho e Contactos.

CRITÉRIOS DE AVALIAÇÃO

— Aplicação dos conhecimentos adquiridos na cadeira

— Construção semântica (só podem utilizar nomes em pt para designar ID's e CLASSES)

— Capacidade de aplicação de novos conhecimentos

— Criatividade

— Tipografia

— Mockups dos layouts

SUPORTES A ENTREGAR

— Maquetas em PDF ou JPG

— Pasta com os ficheiros de implementação (html, css, jpg, etc)

— Publicação online (em servidor a fornecer pelo docente)

REFERÊNCIAS

http://www.semplicelabs.com/

https://onepagelove.com/

https://www.awwwards.com/websites/single-page/

DATA DE ENTREGA

a definir pelo docente.

Este projecto pode ser realizado individualmente ou em grupos de 2 alunos. Bom trabalho.