Introdução ao HTML5 e CSS3 – parte 3

Profª Mª Denilce Veloso
denilce.veloso@fatec.sp.gov.br
denilce@gmail.com

Transition

O que a propriedade transition faz é comparar os valores das propriedades em comum, entre os dois estados do link ou de qualquer outro elemento, assim ela modifica suavemente os valores quando há a ativação da função. Esta é uma técnica simples e que serve para manipularmos transições básicas como cor, tamanho, posição etc.

Se o navegador detecta que há uma propriedade no primeiro estado, mas não no segundo, ele não faz a transição desta propriedade, apenas das propriedades em comuns.

Transition – Ex.: 17UsandoTransition.html (1)

```
!DOCTYPE html>
<html lang=”pt-br”>

<head>
    <meta charset="utf-8">
    <title>CSS Transition</title>
<--! Criado o estilo no próprio HTML -->
    <style>
a {
            color: white;
            -webkit-transition: 0.5s linear;
        }  /* duração e qual propriedade  será alterada*/
```

Transition – Ex.: 17UsandoTransition.html (2)

```
a:hover {
        color: black;
        background: red;
        -webkit-transition: 0.5s linear;
/* duração  0.5s */
    }

        ul li ul {
                display: none;
            }

            /* passou mouse lista não ordenada que está dentro da lista será mostrado em bloco*/
            ul li:hover ul {
                display: block; }
  </style>
<head>
```

Pseudo classes: hover, focus, visited, etc. controlam estado dos elementos

Transition – Ex.: 17UsandoTransition.html (3)

```
</head>

<body>
   <a href="#">Olá</a>

<ul>
                    <li><a href="#">Home</a></li>
                    <li><a href="#">Produtos</a>
```

Transition – Ex.: 17UsandoTransition.html (4)

```
        <ul>
                                <li><a href="#">Carros</a></li>
                                <li><a href="#">Motos</a></li>
                                <li><a
href="#">Caminhões</a></li>
                                <li><a href="#">Barcos</a></li>
                        </ul>
                </li>
                <li><a href="#">Sobre</a></li>
                <li><a href="#">Contato</a></li>
        </ul>
</body>

</html>
```

Transform 2 D

Manipula a forma com que o elemento aparecerá na tela. Poderão ser manipulados: perspectiva, escala e ângulos.

Scale - modificará a dimensão do elemento. Ele aumentará proporcionalmente o tamanho do elemento levando em consideração o tamanho original do elemento.

Skew - modificará os ângulos dos elementos. Poderão ser modificados os ângulos individualmente dos eixos X e Y: Ex.:-

```
webkit-transform: skewY(30deg);
webkit-transform: skewX(30deg);
```

Translation - moverá o elemento no eixo X e Y.

Rotate - rotaciona o elemento levando em consideração seu ângulo, especialmente quando o ângulo é personalizado com o transform-origin. (o padrão é canto superior esquerdo)

** *ver exemplo: 17_2UsandoTransform.html*

Transform 3 D

Transformações 3D são semelhantes a transformações 2D. As propriedades básicas são:

translate3d,**scale3d**, **rotateX**, **rotateY** e **rotateZ**., **translate3d** e **scale3d** possuem três argumentos, para os eixos X, Y e Z.

*** ver exemplo: 17_3UsandoTransform.html*

CSS – Reset

Por padrão, todos os elementos HTML possuem um estilo incorporado, e isso varia de navegador para navegador, não existe um padrão. Com isso podem ocorrer vários problemas ao trabalhar com o CSS.

Para evitar esse tipo de problema, é aconselhável utilizar técnicas de CSS Reset, ou seja, iniciar o CSS com todos as tags HTML sem qualquer estilo padrão definido pelo browser. Existem várias técnicas. Uma bastante conhecida é a proposta por Eric Meyer (veja arquivo: reset.css)

** Uma outra proposta é o normalize.css

Testar o site em vários dispositivos

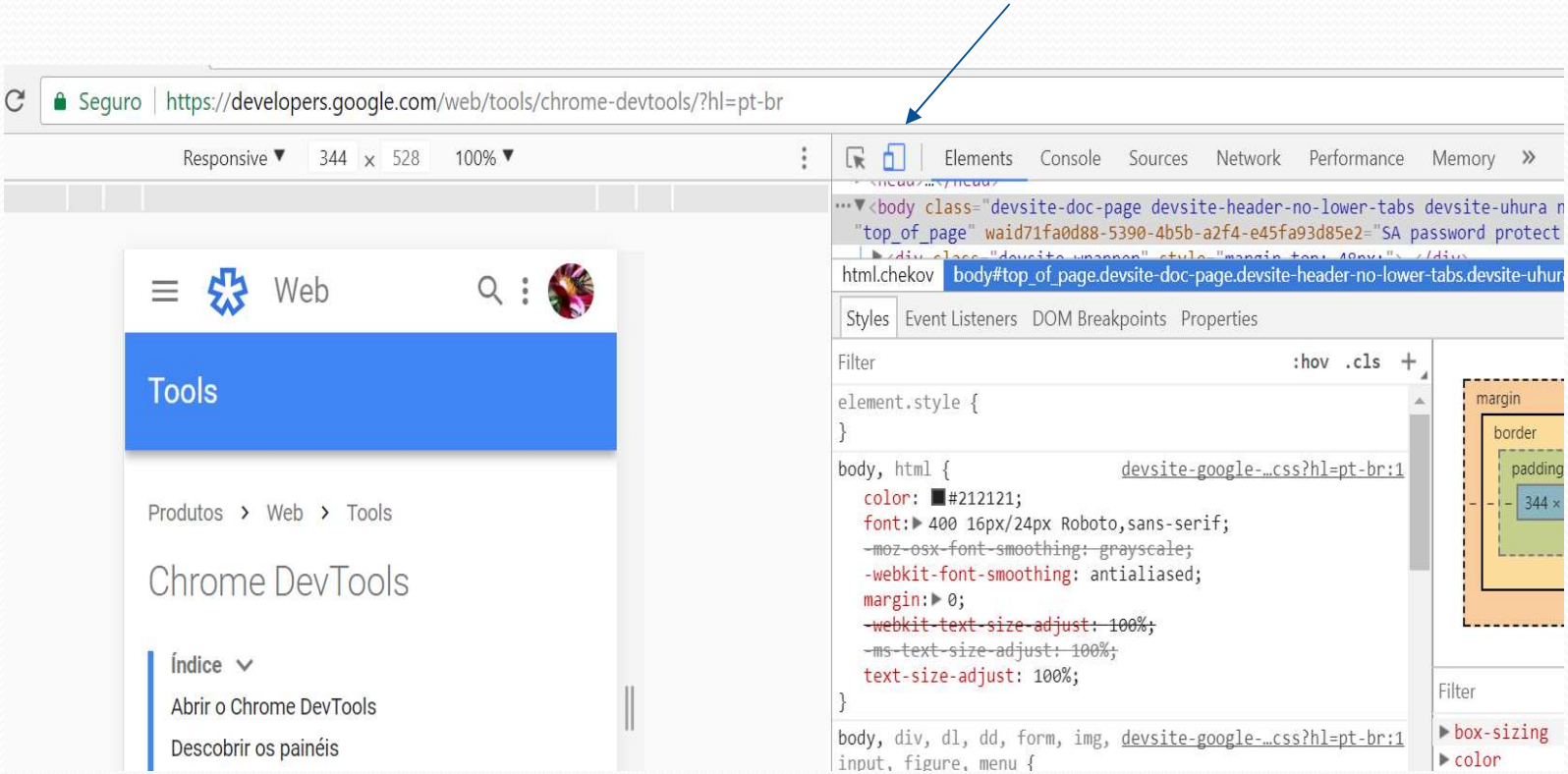
- O WhatIsMyScreenResolution é uma página (http://whatismyscreenresolution.net/multi-screen-test) que permite visualizar como o seu site é visto em diferentes tamanhos de tela. Para utilizá-lo, basta digitar a URL do site que deseja consultar e escolher o tamanho da tela que será verificada.

Testar o site em vários dispositivos

O Google Chrome também oferece uma ferramenta para ajudar a identificar problemas iniciais de desenvolvimento de sites para diferentes telas. Para utilizar esse recurso, vá até o menu do Chrome > "Mais Ferramentas" > “Ferramentas do desenvolvedor” (ou use o atalho “Ctrl + Shift + I” ou F12). Um painel irá aparecer à direita da tela; clique sobre o ícone do celular. Agora, à esquerda, selecione o aparelho que deseja testar em “Device”. Por fim, digite o endereço do site que você quer testar na barra de endereços do navegador. Caso já esteja com ele aberto, basta recarregar as páginas desejadas.

Responsive Web Design

Exemplo do Chrome



Responsive Web Design

Ethan Marcotte criou o termo Responsive Web Design em um artigo escrito no site A List Apart (2010), que então se tornou um livro, que é referência no assunto. A combinação de elementos fluídos, com dimensões relativas ao espaço disponível no aparelho do usuário, e regras específicas para definir estilos específicos para tamanhos de tela diferentes e funcionalidades diferentes. Ele apresenta o arquiteto inglês *Christopher Wren*, que disse certa vez que arquitetura é uma arte que "objetiva a eternidade". Todo arquiteto que se preze quer construir um prédio que seja admirado (e habitado) por séculos... Infelizmente o Design Digital não tem permanência da arquitetura. Na verdade, o que é projetado para internet hoje, logo já estará desatualizado.

Responsive Web Design

Sites responsivos adequam seus elementos e o seu comportamento para cenários, disponibilizando fluxo de navegação que se adequará ao tamanho da tela utilizada ou mudando as interações do usuário caso ele use um dispositivo sensível a toque ou não. Além de precisar identificar o tamanho, capacidade e funcionalidades disponíveis, é importante que a disposição dos elementos seja flexível o bastante para se adequar a qualquer dispositivo.

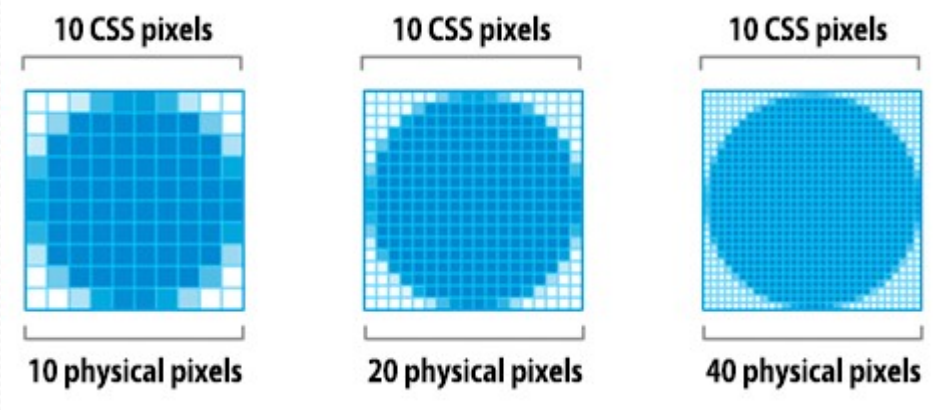
Exemplos: http://www.kadunew.com/blog/web-design/23-exemplos-de-design-responsivo

Responsive Web Design – PPI x DPI

O PPI significa “pixels por polegada” *(pixels per inch)* e o DPI significa “pontos por polegada” *(dots per inch)*. Ambos servem pra denominar a “resolução” de uma imagem, indicando quantos pixels ou pontos existem em uma polegada, mas sem especificar proporções ou dimensões. A verdade é que um está associado à “qualidade” de reprodução de imagens em displays e o outro à “qualidade” de impressão. Além de servir para a reprodução de conteúdo visual na tela do seu computador, o pixel serve para a composição de imagens, as fotografias, por exemplo, são compostas por uma série de pixels. O pixel não tem tamanho fixo, sendo que ele tem suas dimensões alteradas conforme a capacidade do aparelho em questão. Uma tela de 42 polegadas com resolução Full HD exibirá pixels maiores do que um smartphone com display de 5 polegadas e de mesma resolução. Apesar disso, o DPI é dependente dos valores em pixels, já que é necessária uma conversão no computador antes de mandar um arquivo para impressão.

Responsive Web Design – PPI x DPI

Um único pixel de CSS podem conter vários pixels do dispositivo. Por exemplo, um único pixel de CSS pode corresponder diretamente a um único pixel de dispositivo ou a vários pixels de dispositivo. Quantos mais pixels de dispositivo, maiores os detalhes do conteúdo exibido na tela. Telas com alto DPI tem melhores resultados.



https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/image-optimization?hl=pt-br

Layouts fluidos

O primeiro passo para se começar a trabalhar com designs responsivos é fazer a transição de elementos de dimensões fixas em px - para elementos fluídos - que se baseiam em porcentagens e em. Desta forma, a sua página consegue se expandir ou diminuir de acordo com a largura disponível, em vez de se fixar em algo similar a 960px de largura máxima.

Layouts fluidos

A conversão de tamanhos fixos para fluídos é feita com base em uma fórmula bastante simples: "tamanho fixo"/ "contexto"= "tamanho fluído". Com ela é possível converter dimensões em px para porcentagens (para larguras, alturas, padding, margens) ou em em (para fontes), assim:

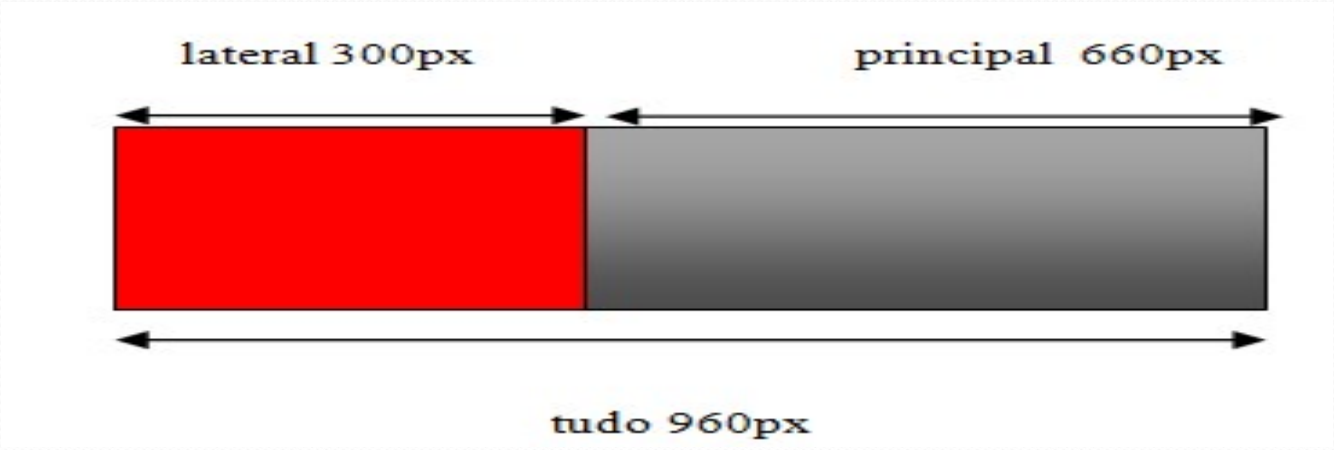
```
/* Utilizando o tamanho de fonte padrão 16px
  se quiser mudar o padrão é só fazer */
html {
  font-size:16px;
}

body {
font: normal 100% Helvetica, Arial, sans-serif;
}

h1 {
font-size: 1.5em; /* 24px / 16px = 1.5em */
}
```

** A princípio, esta definição terá o mesmo efeito que definir o tamanho da fonte do h1 como 24px. Porém, em dispositivos que o tamanho padrão de fonte seja diferente, como em smartphones, o font-size do elemento irá acompanhar esta mudança de tamanho: em um dispositivo que utilize 12px a fonte do elemento terá 18px de altura,e não 24px.

Layouts Fluidos – Exemplo



OBJETO / CONTEXTO = RESULTADO

É necessário calcular o valor em porcentagem da lateral e da área de conteúdo (principal) **em relação** ao valor total do site (tudo) (960px). O "tudo" seria o contexto. Então: 300 ÷ 960 = 0.3125 Segundo autores não é aconselhável arredondar, colocar apenas duas casas para a direita e acrescentar o sinal de porcentagem. Ou seja: 31.25%.

Layout Fluidos - Exemplo 20UsandoFluidos.html (1)

```
<!DOCTYPE html>
<html lang=“pt-br">
<head>
   <meta charset="UTF-8">
   <title>Teste responsivo</title>
   <link rel="stylesheet" href="20UsandoFluidos.css">
</head>
<body>

  <div class="tudo">

      <div class="lateral">
         <p>  lateral TEXTO 1</p>
           <p>  lateral TEXTO 2</p>
      </div>

      <div class="principal">
         <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer malesuada
augue eu mattis malesuada. Donec quis mollis metus, vel egestas enim. Nulla sodales
turpis vitae iaculis venenatis. Suspendisse tristique orci eget ex faucibus, ut placerat
massa dapibus. Praesent ut est consequat, ultricies nibh sed, pulvinar velit. Aliquam
posuere tellus eu erat lacinia consequat. Integer diam justo, cursus maximus pretium
```

Layout Fluidos - Exemplo 20UsandoFluidos.html (2)

blandit, malesuada quis est. Nunc lobortis ex sed interdum bibendum. Nam cursus velit non velit porttitor, sit amet scelerisque lorem tincidunt. Etiam ut feugiat lacus. Nunc euismod tristique enim, quis accumsan felis convallis sit amet. Aenean euismod ante id iaculis volutpat. Proin gravida imperdiet consequat. Praesent eleifend tortor vel felis luctus feugiat. Fusce sed arcu et ex volutpat placerat. Donec a tortor gravida, commodo nisi a, commodo augue. Vestibulum sed urna porttitor nibh dapibus facilisis. Phasellus rutrum elementum massa, vel bibendum elit molestie id. Maecenas tempus pharetra mi, ut commodo nisl maximus id. Nunc fringilla elementum mi nec maximus. Morbi orci nulla, aliquam et aliquam nec, porta vel dui. Proin iaculis sed purus nec porttitor. Vestibulum sed finibus ante. Nunc rutrum mi vel sapien tempus, eu maximus risus ultrices. Quisque ac metus sed justo maximus pretium. Ut tempus blandit orci, in efficitur tellus gravida a. Vivamus eu nisl in lectus porta lacinia ut sed metus. Suspendisse iaculis imperdiet dapibus. Mauris arcu sem, iaculis in justo ac, scelerisque gravida urna. Integer nec risus scelerisque, consectetur sem quis, mollis elit. Phasellus malesuada vulputate interdum. Nunc ac lorem et nunc aliquet molestie nec at mi. Cras purus tellus, dapibus sed ultricies ut, tempor a purus. Sed eget magna vitae est iaculis mattis. Nulla magna odio, eleifend consequat feugiat ac, pharetra in ligula. Quisque et dui ut mauris mollis sollicitudin eget eu dui. Aenean suscipit placerat posuere. Interdum et malesuada fames ac ante ipsum primis in faucibus.

```
</p>
      </div>
         </div>
  </body>
</html>
```

Layout Fluidos - Exemplo 20UsandoFluidos.css (1)

```
.tudo {
width:960px;
margin: 0 auto; /* top e bottom 0 e left e rigth auto */
    border: 1px solid #000;
        }

/* regra de 3, se 16px é 1  então 26px é x -> x=26/16 */
p {
font-size: 1.6em; /*26 pixels/16*/
line-height: 1.5em; /*24 pixels/16 */
padding: 2.5em; /*40 pixels/16 */
text-align: justify;
}


.lateral {
width: 31.25%; /* 300px / 960px */
float: left;
border: auto solid red;
}
```

Layout Fluidos - Exemplo 20UsandoFluidos.css (2)

```
.principal {
width: 68.75%; /* 660px / 960px */
float: right;
border: auto solid blue;
}
```

Layouts fluidos

Como a maior parte dos monitores eram da resolução 1024×768 foi padronizado 960 pixels como a largura ideal para um site. E isso funcionou bem, por um tempo... Mas os dispositivos estão evoluindo tão rápido. Como fica site de 960 pixels em um iMac de 27"? E o espaço vazio. Estranho? E em uma tv de 50"?

Que bom que existem as Medias Queries ;)

Media Queries

Outra ferramenta do ResponsiveWeb Design são os media queries - uma regra específica para aplicar um bloco de CSS caso a regra seja atendida. Também pode indicar regras associadas ao tamanho, orientação ou a resolução de tela.

```
@media only screen
and (min-device-width : 320px)
and (max-device-width : 480px) {



/*
Nesse exemplo a regra será aplicada em dispositivos com uma largura de 320px a 480px.
*/
}
```

Media Queries

Outra ferramenta do ResponsiveWeb Design são os media queries - uma regra específica para aplicar um bloco de CSS caso a regra seja atendida. Também pode indicar regras associadas ao tamanho, orientação ou a resolução de tela.

```
@media only screen
and (min-device-width : 320px)
and (max-device-width : 480px) {



/*
Nesse exemplo a regra será aplicada em dispositivos com uma largura de 320px a 480px.
*/
}
```

Media Queries – width:100%;

Como o modelo de caixa inclui preenchimento, bordas e margens, definir a largura de um elemento em 100% pode fazer com que o conteúdo transborde o contêiner pai. Por isso é aconselhável que o 100% seja utilizado quando o conteúdo não terá preenchimento, borda ou margem.

Media Queries – %

As porcentagens são comumente usadas para dimensionar recursos das caixas, como width, heigth, borda, padding ou margin de um elemento.

Quando width e heigth são definidos usando porcentagens as dimensões dos elementos filho são calculadas com base nas dimensões do elemento pai.

Quando as porcentagens são usadas para definir padding e margin, no entanto, elas são calculadas com base apenas na largura do elemento pai.

Media Queries – Min-width e Max-width

Embora as medidas relativas forneçam layouts consistentes em dispositivos de diferentes tamanhos de tela, os elementos em um site podem perder sua integridade quando eles se tornam muito pequenos ou grandes. É possível limitar a amplitude de um elemento com as seguintes propriedades:

Min-width - largura mínima para um elemento.
Max-width - largura máxima para um elemento.

```
p {
  min-width: 300px;
  max-width: 600px;
}
```

É possível também limitar a altura mínima e máxima de um elemento.

Min-height - altura mínima para a caixa de um elemento.
Max-height - altura máxima para a caixa de um elemento.

Media Queries – Min-width e Max-width

Quando a altura de uma imagem ou vídeo é definida, sua largura pode ser definida como automática para que a mídia fique proporcional. Inverter essas duas propriedades e valores também alcançará o mesmo resultado.

Uma imagem de plano de fundo de um elemento HTML será dimensionada proporcionalmente quando sua propriedade de tamanho de fundo estiver configurada para cobrir tudo (100%).

Media Queries – EM e REM

REM - é calculado em cima do valor da TAG Body, Html ou Navegador. (elemento root)

EM - já utiliza no cálculo o valor do elemento pai, criando uma hierarquia no cálculo.

Media Queries

```
@media only screen
and (min-device-width : 768px)
and (max-device-width : 1024px)
and (orientation : landscape) {

/*
Além de verificar a largura do dispositivo, é possível conferir a orientação, utilizando
"landscape“  paisagem  ou "portrait“ retrato.
*/

}
@media only screen and (min-width : 1824px) {

/*
Aqui  é possível adicionar estilos para  aproveitar  uma tela maior: aumentando
fontes, ícones, e largura dos seus elementos, tipo Macs */
}
```

** Ver exemplos em: http://mediaqueri.es/

Meta Tag ViewPort

O viewport é a área onde o website aparece no device. É a área branca da janela quando o usuário abre o browser. O viewport sempre vai ter o tamanho da janela. Mas a forma como os elementos são renderizados vai depender bastante do dispositivo. Em máquinas desktop, existem tamanho de tela e resolução média utilizada pelos usuários. Mas quando começa a variar muito o tamanho das telas, a largura do viewport começa a ser uma preocupação porque afeta diretamente a forma como o usuário utiliza seu website. O ponto é que em uma tela pequena, com uma resolução muito grande, por exemplo como as telas dos iPhone e da maioria dos smartphones de hoje, o conteúdo pode aparecer muito pequeno. Por isso é interessante customizar o viewport para que ele se adeque a realidade desses dispositivos. É aí que entra a metatag viewport.

Com essa metatag pode-se customizar a resolução inicial que o browser deve renderizar do viewport do dispositivo. Dessa forma, pode-se preparar websites com resoluções personalizadas para smartphones e outros aparelhos.

Exemplo Utilizando Media Queries

Imagine um layout de página básico, com cabeçalho, conteúdo, barra lateral e rodapé. O cabeçalho tem uma altura fixa de 180px, o conteúdo tem uma largura de 600px, a barra lateral tem 300px de largura.

Media Queries: 21UsandoMediaQueries.html(1)

```
<!doctype html>
<html lang="pt-br">
<head>
<meta charset="utf-8">


<!-- viewport meta to reset iPhone inital scale -->
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Media Queries: 21UsandoMediaQueries.html(2)

```
<title>Design Responsivo</title>

  <link rel="stylesheet" href="UsandoResponsivo.css">

   </head>

<body>

<div id="pagina">
        <div id="cabecalho">
                <h1>Cabeçalho</h1>
                <p>área do cabeçalho</p>
        </div>

        <div id="conteudo">
                <h2>Conteúdo</h2>
                <p>linha 1 do conteúdo</p>
                <p>linha 2 do conteúdo</p>
                <p>linha 3 do conteúdo</p>
```

Media Queries: 21UsandoMediaQueries.html (3)

```
<p>linha 4 do conteúdo</p>
                    <p>linha 5 do conteúdo</p>
                    <p>linha 6 do conteúdo</p>
                    <p>linha 7 do conteúdo</p>
                    <p>linha 7 do conteúdo</p>
                    <p>linha 9 do conteúdo</p>
                    <p>linha 10 do conteúdo</p>

                    </div>
          <div id="lateral">
                    <h3>Barra Lateral</h3>
                    <p>texto1 da barra lateral </p>
                    <p>texto2 da barra lateral </p>
          </div>
                    <div id="rodape">
                    <h4>Rodapé da Página</h4>
          </div>
</div>
</body>
</html>
```

Media Queries: 21UsandoMediaQueries.css(1)

```
body {
        font:  150% Arial, Helvetica, sans-serif;
}
/* 150% do tamanho original ou 1,5 vezes maior*/
h1 {
        font: bold  100% Arial, Helvetica, sans-serif;
}

#pagina {
        padding: 5px;
        width: 960px; /* tamanho mais comum usado */
        margin: 20px auto;
}

#cabecalho {
        height: 180px;
}
```

Media Queries: 21UsandoMediaQueries.css(2)

```
#conteudo {
        width: 600px;
        float: left;
   background: #f8f8f8;
}

#lateral {
        width: 300px;
        float: right;
   background: #foefef;
}

#rodape {
        clear: both;
}

#cabecalho, #conteudo, #lateral {
        margin-bottom: 5px;
}
```

Media Queries: 21UsandoMediaQueries.css(3)

```
#pagina, #cabecalho, #conteudo, #lateral, #rodape {
        border: solid 1px #ccc;
}

/* Media queries/*

/*  980px ou menor */
@media screen and (max-width: 980px) {

        #pagina {
                width: 94%;
        }
        #conteudo {
                width: 65%;
        }
        #lateral {
                width: 30%;
        }

}
```

Media Queries: 21UsandoMediaQueries.css(4)

```
/* 700px ou menor –  removendo divs e colocando auto e none vai ocupar 100% tela*/
@media screen and (max-width: 700px) {

        #conteudo {
                width: auto;
                float: none;
        }
        #lateral {
                width: auto;
                float: none;
        }

}
```

Media Queries: 21UsandoMediaQueries.css(5)

```
/*  480px or menor  -  redefinir a altura da <div> “header” deixando-a automática,
mudar o tamanho da tag <h1> para 24px e esconder a <div> “sidebar”. */
@media screen and (max-width: 480px) {

        #cabecalho {
                height: auto;
        }
        h1 {
                font-size: 24px;
        }
        #lateral { /* nao mostra a lateral */
                display: none;
        }

}
```

Utilizando Media Queries(vários arquivos css)

É possível utilizar vários arquivos para diferentes tipos de dispositivos.
Ex.:

```
<link rel="stylesheet" media="screen and (min-width: 500px)" href="yes.css">
```

Meta Tag ViewPort

A sintaxe é muito simples e deve ser colocada na tag **head**:
<meta name="viewport" content="">

Os valores de content são os que seguem abaixo:

Valor	Descrição
width	Define uma largura para o viewport. Os valores podem ser em PX ou "device-width", que determina automaticamente um valor igual a largura da tela do dispositivo.
height	Define uma altura para o viewport. Os valores podem ser em PX ou "device-height", que determina automaticamente um valor igual a altura da tela do dispositivo.
initial-scale	Define a escala inicial do viewport.
user-scalable	Define a possibilidade de o usuário fazer "zoom" em um determinado lugar da tela. É ativado quando o usuário bate duas vezes com o dedo em um lugar da tela.

Relação entre pixels do dispositivo e do css

```
<!-- reset inital scale -->
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Exercício Responsivo

- Testar os templates de páginas responsivas em vários tamanhos de dispositivos.

- Ver arquivo:
- 16_5ExemploResponsivoUsandoFlexBox.html

- Ver pasta: Exemplo da empresa https://templated.co. (inicie pelo index.html)

CONCLUINDO: Seletores e Propriedades

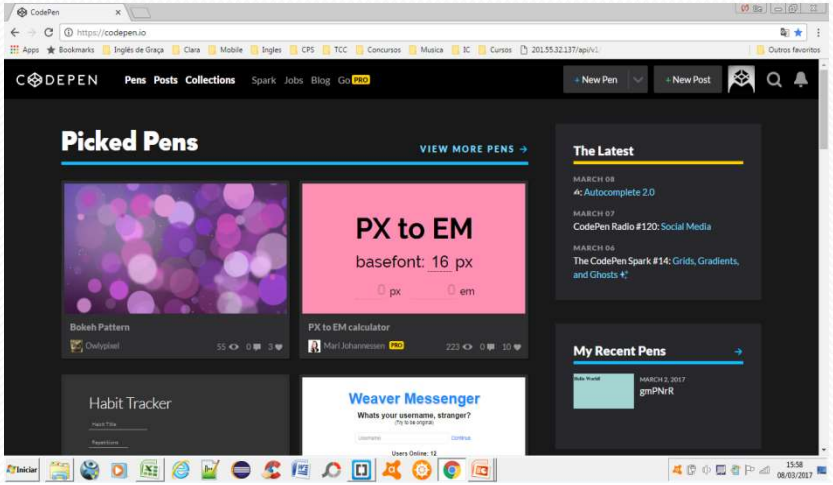
As unidades de medida relativas são um primeiro passo para a incorporação de design responsivo em um site. Quando combinado com técnicas de resposta mais avançadas, é possível criar uma experiência de usuário perfeita, independentemente do tamanho de tela de um dispositivo.

Não é necessário saber todas as propriedades, mesmo porque sempre estão sendo criadas novas propriedades.
Uma boa prática é conhecer os seletores, pois sempre a partir dele que será realizada a formatação e boa parte da inteligência consiste em utilizá-los de maneira eficaz, escalável e inteligente. (CSS2, 2016)

seletor { propriedade: valor; }

Para estudar mais ...

No CodePen é possível criar, compartilhar e pesquisar exemplos de códigos. Os itens, ou "pens", são basicamente divididos em dois painéis: um referente ao código e outro com os elementos renderizados ao vivo. Isto significa que é possível editar e copiar snippets (pedaços de códigos fontes reutilizáveis) e verificar as mudanças em tempo real. É possível ainda criar *forks (*softwares que originam de projetos já existentes dando origem a outros da mesma ramificação). para as suas alterações / melhorias e compartilhar novamente com a comunidade. É possível encontrar nos arquivos do CodePen todo tipo de coisa: animações, elementos de user interface e até pequenos jogos. O editor do CodePen suporta elementos como LASS, SASS e etc. (https://codepen.io/)



Dicas de Software para compressão de Imagens

- 1. PNGGauntlet
- 2. Compressor.io
- 3. Caesium
- 4. Kraken.io
- 5. Optimizilla
- 6. TinyPNG
- Mass Image Compressor
- RIOT
- FileOptmizer

Referências:

https://www.e-konomista.pt/programas-para-comprimir-imagens/

https://www.topfreewares.com.br/top-5-programas-compactar-fotos-windows/

Frameworks para CSS - Bulma

Como o título diz, o Bulma é um framework CSS baseado na tecnologia flexbox, que já tem uma grande compatibilidade entre os navegadores. O pacote contém todos os elementos mais comuns como botões, formulários, menus, tabelas, títulos, notificações, barras de progresso e um simples sistema de grid (basta adicionar uma coluna, o resize das colunas é automático).

9 Best CSS Frameworks in 2021
https://athemes.com/collections/best-css-frameworks/

Slide 49

DV1 DENILCE VELOSO; 10/03/2021

Frameworks para CSS – Pure.css

É um Framework CSS responsivo, que pode ser utilizado em qualquer projeto web. O Framework é baseado no projeto Normalize.css, que é uma biblioteca que ajuda o navegador a renderizar elementos com maior consistência, está de acordo com os padrões modernos de desenvolvimento e funciona em navegadores antigos.

Frameworks para CSS – Kube

É um framework minimalista para desenvolvedores e designers profissionais. Este framework fornece total liberdade de escolha, facilitando apenas o básico na produção de elementos, assim como outros frameworks da lista, deixando o design para terceiros.

Frameworks para CSS - Materialize

É um Framework desenvolvido para facilitar e agilizar o desenvolvimento quando o assunto é Material Design. Os arquivos são bem distribuídos e não diferem dos demais frameworks.

Frameworks para CSS - Foundation

É um framework indicado para o desenvolvimento responsivo de forma rápida priorizando o valor semântico dos elementos, mantendo uma estrutura HTML limpa e de carregamento rápido. Utiliza o conceito mobile first e melhoria progressiva (Progressive Enhancement) para criação de design responsivo.

Frameworks para CSS - Bootstrap

É um framework que facilita a vida dos desenvolvedores web a criar sites com tecnologia mobile (responsivo) sem ter que digitar uma linha de CSS.

Além disso, o Bootstrap possui uma **diversidade de componentes (plugins) em JavaScript (jQuery)**que auxiliam o designer a implementar: tootlip, menu-dropdown, modal, carousel, slideshow, entre outros sem a menor dificuldade, apenas acrescentando algumas configurações no código, sem a necessidade de criar scripts e mais scripts.

Pré processadores SASS e LESS

- Os pré-processadores são programas que recebem como entrada textos e efetuam conversões léxicas nele, ou seja, entra com uma sintaxe, essa sintaxe é pré-processada para gerar outra sintaxe. As conversões podem incluir substituição de palavras, inclusão condicional e inclusão de outros arquivos.

Exemplos de pré-processadores :

- **HTML:** HAML, Markdown, Slim;
- **CSS:** Sass, LESS, Stylus;
- **JS:** CoffeeScript;

Mais artigos sobre CSS

- https://desenvolvimentoparaweb.com/categoria/css/

Referências (1)

ACESSIBILIDADELEGAL, Disponível em: <http://www.acessibilidadelegal.com/23-webstandards.php> Acesso em: Jan. 2015.

BOOTSTRAP. http://www.tutorialwebdesign.com.br/o-que-e-bootstrap/ Acesso em: Fev.2016.

CAELUM. Apostilas Cursos Gratuitas https://www.caelum.com.br/apostilas/. Acesso em: Jan. 2015.

CURSOCSS. CSS CURSO W3C ESCRITÓRIO BRASIL. http://www.w3c.br/Materiais/ Acesso em dez.2016

CODEACADEMY. Cursos Gratuitos. https://www.codecademy.com/pt Acesso em: Jan. 2015.

CSS. https://css-tricks.com/ Acesso em: Fev.2016.

FRAMEWORKS(1). Disponível em: <https://www.kinghost.com.br/blog/2017/06/5-frameworks-front-end-populares-em-2017/> Acesso: 01.JUL.2017.

FRAMEWORKS(2). Disponível em: https://three29.com/best-css-frameworks-2017/ Acesso:01.JUL.2017.

FONTSIZW. http://kyleschaeffer.com/development/css-font-size-em-vs-px-vs-pt-vs/ Acesso em: Fev.2016

JOBSTRAIBIZER, Flávia. Criação de Sites com Css - Desenvolva páginas Web mais leves e dinâmicas em menos tempo. 1ª Ed. São Paulo: Digerati Books.,2009.

MARCOTTE, Ethan. Responsive Web Design. New York: A Book Apart, 2011.

MAZZA, Lucas. HTML5 e CSS3. Domine a Web do Futuro. Casa do Código, 2012.

Referências (2)

RESPONSIVO. http://wpmidia.com.br/desenvolvimento-web/design-responsivo-em-3-passos/comment-page-1/#comments Acesso em: Fev.2016

RESPONSIVO2. http://blog.popupdesign.com.br/design-responsivo-i-o-que-e-e-por-que-usar/ Acesso em: Fev.2016

UDACITY. Exemplo de Fontes. https://github.com/jeffhill76225/Responsive-Web-Design Acesso em:01.AGO.2017.

W3C. Site da W3C oficial. Disponível em: <http://www.w3.org> Acesso em: Mai.2015.

WILTON, Paul, McPeak, Jeremy. Beginning JavaScript. 3rd Edition. Indianapolis (EUA): Wiley Publishing, Inc., 2007.

WROBLEWSKI, Luke. *Mobile First*. New York: A Book Apart, 2011.

ZEMEL, Tárcio. Web Design Responsivo: Páginas adaptáveis para todos os dispositivos. 1ª Ed. São Paulo: Casa do Código, 2012.