INSTITUTO FEDERAL DO PIAUÍ - IFPI

Bruno Serra

Guilherme Rodrigues

Avaliação 01 : Descrição dos itens implementados na construção do site estático.

Relatório técnico apresentado como requisito para uma breve descrição do trabalho avaliativo, realizado para a matéria de Programação Para Internet do Curso de Análise e Desenvolvimento de Sistemas, no Universidade ,Instituto Federal do Piauí.

Professor Ely da Silva Miranda

Resumo

Este Trabalho visa a construção de uma interface web com o uso de HTML5 e CSS3, ambos conteúdos abordados em sala de aula. O objetivo proposto pelo discente foi que ao criarmos a interface web usássemos os recursos oferecidos pela nova versão da linguagem de marcação HTML5 assim como em conjunto com CSS3 para estilizar as páginas criadas. Foi criado um arquivo css para trazer estilo a duas páginas html, ambas páginas aproveitam as novas tags semânticas que a API do HTML5 traz.

Palavras-chave: Relatório, CSS3, HTML5, semântica.

Sumário

1. INTRODUÇÃO

Através da realização desse trabalho, a dupla em questão realizou através de métodos validados do mercado de trabalho na construção de páginas web(Client Side), o aprendizado prático das tecnologias usadas. Todo o processo foi avaliado através do versionamento de código utilizando git com a hospedagem destes códigos no github.

2. DESENVOLVIMENTO

No início do trabalho acadêmico foram listados temas de sites que poderíamos fazer de exemplo, e como escolha tivemos o planejamento de fazer um site para uma “empresa” de estúdio, onde a mesma teria uma seção destinada a seus projetos desenvolvidos e uma área para contato, bem como a criação da página referente ao projeto que a empresa criou.

2.1 - Estrutura das pastas.

Organizamos a estrutura do projeto como:

index.html
project.html
/css
style.css
normalize.css
font-awesome.min.css
/fonts
FontAwesome.otf
/img

2.1.1 - Organização de cada arquivo do projeto.

Tivemos por opção tal separação de pastas, para que tivéssemos o controle melhor sobre as dependências que nossa página teria que obter como o css-reset no normalize, o arquivo de font-awesome minificado para obtermos acesso a várias fontes sem precisar importá-las em nosso css.

3 - METODOLOGIAS DE CÓDIGO.

HTML -

Como citado no início do relatório, procuramos seguir as boas práticas de html semântico. Então durante o desenvolvimento procuramos separar cada elemento a seu destino como exemplo: Elementos de menu; **ul, li ,** Dentro de uma tag nav que seria filha de uma header, dessa forma o navegador interpreta com mais facilidade do que separando tudo em div’s.

Outras tags semânticas usadas foram: Section, Main

CSS -

Durante a definição das propriedades CSS sobre o HTML, foi definido o uso da prática de declarar primeiro valores que respeitem as telas

a partir de 900px, para depois assim declararmos através de Breakpoints com o uso do recurso media query, para termos assim os elementos se adaptando a cada tipo de tela a medida que a resolução mudada. Os breakpoints foram **minimo de 600px, mínimo de 900px, entre 0 (mínimo) até o máximo de 899px .**

Vendo a necessidade de termos um menu personalizado para mobile, foi criado uma classe chamada **sandwich**, dentro do **container** relacionado ao **menu**, para que quando a resolução fosse relacionado a mobile, o menu responsivo então surgisse, junto com um **icon** para abrir e fechar o conteúdo.

Além do que citado acima, foi utilizado o acesso a pseudo classes dos elementos, onde podemos definir comportamentos para os mesmos baseado no estado atual como: **:active**, **:focus**, **:hover**, **:visited**.

Exemplo: **button:hover{}, input:focus, textarea:focus {}**

Flexbox: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Seguindo o guideline citado acima, com o uso de flexbox, foi possível obter recursos de alinhamento e distribuição de espaço entre itens dentro da interface. Tal recurso é atualmente utilizado para as construções de grid, como foi feito a seção de projetos.

JAVASCRIPT -

Para possibilitar o uso de um menu "sanduíche" em resoluções pequenas, essa linguagem de programação foi usada para acesso ao **DOM,** fazendo assim a mudança de elementos através do acesso de seus respectivos IDs (identificadores).

PRINCIPAIS CÓDIGOS CSS:

breakpoints - Media - Query

mínimo de 600px -

```
@media screen and (min-width: 600px) {
        .content h1 {
            font-size: 50px;
            line-height: 60px;
        }
}
```

mínimo de 900px -

```
@media screen and (min-width: 900px) {
        header nav {
           display: inline-block;
        }.content h1 {
           font-size: 65px;
           line-height: 70px;
        }
}
```

mínimo de 0px, máximo de 899px .

```
@media screen and (min-width: 0) and (max-width:
899px) {
  main section p {
     width: 100%;
     text-align: center;
     display: block;
  }
  main section .team {
     top: 0;
     margin: 40px auto;
     float: none;
     box-sizing: border-box;
     width: 342px;
     padding-left: 42px;
  }
  main section h2:before {
     margin: auto;
     position: relative;
     top: -45px;
  }

  .button {
     width: 100%;
     box-sizing: border-box;
     text-align: center;
  }

  h2 {
     width: 100%;
     text-align: center;
  }
  .client {
     top: 0;
     margin: auto;
  }
  .sandwich {
     display: block;
```

```
    position: absolute;
    top: 20px;
    right: 20px;
    z-index: 9999999;
  }

  .portfolio {
    width: 100%;
    margin: 0;
  }
  .portfolio .job {
    width: 100% !important;
  }

  .form-wrapper {
    width: 100%;
  }

  .maps {
    float: none;
    width: 100%;
    margin: 0;
  }

  .gallery {
    top: 0;
  }
  .gallery div {
    width: 100%;
    margin-right: 0;
  }
}
```

Uso da propriedade z-index -

Serve para controlar as camadas de divs. Normalmente o último div da página fica por cima dos outros. Se usarmos z-index podemos alterar esse comportamento. O valor 9999 serve para termos certeza que ele sempre vai estar em cima.

```
header{
  width: 100%;
  display: inline-block;
  position: fixed;
  z-index: 9999;
  box-sizing: border-box;
  padding: 30px 0;
  background: linear-gradient(to bottom, rgba(0, 0, 0, 0.7) -11%);
  filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#000000', endColorstr='#00000000',GradientType=0 );
  text-shadow: 0px 0px 23px rgba(0, 0, 0, 0.3);
}
```

Propriedade Transform - A propriedade transform irá modificar a forma do elemento

Translate

Essa propriedade moverá o elemento no eixo X e Y.

```
.sandwich .icon.opened .bar3 {
  -webkit-transform: rotate(45deg) translate(-6px, -9px);
  transform: rotate(45deg) translate(-5px, -9px);
}
```

rotate -

O rotate rotaciona o elemento levando em consideração seu ângulo, especialmente quando o ângulo é personalizado com o transform-origin.

```
.sandwich .icon.opened .bar1 {
  -webkit-transform: rotate(-45deg) translate(-2px, 6px);
  transform: rotate(-45deg) translate(-2px, 6px);
}
```

Scale

Ao passarmos o mouse em cima das imagens, sua escala aumentará. Para entender o valor de referência da escala: 1 é o tamanho original do elemento. 2 é o dobro deste tamanho.

```
.button:hover {
  transform: scale(1.05);  opacity: 0.7;}
```

Propriedade transition -

Ela permite definir a transição entre dois estados de um elemento, com o auxilio das pseudo-classes.

```
a {
  opacity: 1;
  transition: opacity 0.3s ease-in-out;
}
a:hover{
  opacity: 0.7;
}
```

Uso de pseudo-classes: hover, visited - Palavra chave adicionada aos seletores que especifica o estado atual do elemento selecionado.

```
header a:hover,
header a:visited {
  color: #fff;
  text-decoration: none;
}
```

a:hover - define o estilo do link quando passa-se o mouse sobre ele;

a:visited - define o estilo do link visitado;

a:focus - é aplicada quando um elemento recebe foco, quando o usuário utilizando o mouse sobre o elemento ou o teclado

```
input:focus, textarea:focus {
  outline: 0;
  transform: scale(1.1);
  background-color: white;
}
```

Uso do webkit - um motor de renderização utilizado em navegadores web para renderizar páginas.

```
* {
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
```

Uso dos pseudo-elementos before -

O before(depois) insere uma div antes da div relacionada do objeto pelo pseudo-elemento

```
main section h2:before {
  content: "";
  position: absolute;
  height: 6px;
  width: 50px;
  background-color: #333;
  display: block;
  top: -20px;
  border-radius: 10px;
  opacity: 0.2;
}
```

Definição do menu responsivo - Menu responsivo quando for aberto e definição da classe sandwich manipulando as divs com classe "bar"

```
.menu-full {
  transform: translate(0, 100%);
  opacity: 0;
  position: fixed;
  width: 100%;
  height: 100%;
  background-color: #4c4c4c;
  z-index: 999999;
  color: #fff;
  left: 0;
  top: 0;
  box-sizing: border-box;
```

```
  padding: 40px;
  -webkit-transition: transform 0s linear 0.2s,
opacity 0.2s ease-in-out;
  -moz-transition: transform 0s linear 0.2s,
opacity 0.2s ease-in-out;
  -ms-transition: transform 0s linear 0.2s,
opacity 0.2s ease-in-out;
  -o-transition: transform 0s linear 0.2s, opacity
0.2s ease-in-out;
  transition: transform 0s linear 0.2s, opacity
0.2s ease-in-out;
}
.menu-full a {
  color: #fff;
  font-size: 19px;
  margin-bottom: 20px;
  display: block;
}
.menu-full.opened {
  transform: translate(0, 0);
  opacity: 1;
  -webkit-transition: opacity 0.2s ease-in-out;
  -moz-transition: opacity 0.2s ease-in-out;
  -ms-transition: opacity 0.2s ease-in-out;
  -o-transition: opacity 0.2s ease-in-out;
  transition: opacity 0.2s ease-in-out;
}

.sandwich {
  display: none;
}
.sandwich .icon {
  color: #fff;
  display: inline-block;
  text-align: right;
  position: relative;
  top: 5px;
}
```

```
.sandwich .icon .bar1, .sandwich .icon .bar2,
.sandwich .icon .bar3 {
  width: 25px;
  height: 2px;
  background-color: rgba(255, 255, 255, 0.9);
  margin: 6px 0;
  transition: 0.3s ease-out;
}
.sandwich .icon.opened .bar1 {
  -webkit-transform: rotate(-45deg)
translate(-2px, 6px);
  transform: rotate(-45deg) translate(-2px,
6px);
}
.sandwich .icon.opened .bar2 {
  opacity: 0;
}
.sandwich .icon.opened .bar3 {
  -webkit-transform: rotate(45deg)
translate(-6px, -9px);
  transform: rotate(45deg) translate(-5px,
-9px);
}
```

Uso do Flexbox -

display: flex -Torna o elemento um flex container automaticamente transformando todos os seus filhos diretos em flex itens.

flex-grow - Define a habilidade de um flex item crescer

flex-wrap -Define se os itens devem quebrar ou não a linha. Por padrão eles não quebram linha, isso faz com que os flex itens sejam compactados além do limite do conteúdo. Essa é geralmente uma propriedade que é quase sempre definida como flex-wrap: wrap; Pois assim quando um dos flex itens atinge o limite do conteúdo, o último item passa para a coluna debaixo e assim por diante.

```
.gallery {
  display: flex;
```

```
  width: 100%;
  position: relative;
  top: -30px;
  flex-wrap: wrap;
}
.gallery div {
  display: inline-block;
  width: 25%;
  flex-grow: 1;
  margin-right: 40px;
  border-radius: 20px;
  overflow: hidden;
  background-size: cover;
  height: 400px;
  margin-bottom: 40px;}
```

Uso da propriedade calc no width -

Recurso utilizado para pegar uma porcentagem que ao ultrapassar a largura de elementos na coluna quebrar o último elemento para a parte de baixo.

```
.portfolio .job {
  width: calc(40% - 40px);
  height: 230px;
  background-color: #777777;
  border-radius: 230px;
  float: left;
  margin-right: 40px;
  margin-top: 40px;
  overflow: hidden;
  background-size: cover;
  position: relative;
  transform: scale(1);
  transition: transform 0.3s ease-in-out;
}

.portfolio {
  display: inline-block;
  position: relative;
  width: calc(100% + 40px);
  margin-bottom: 40px;
```

```
}
.portfolio .job.expand {
  width: calc(60% - 40px);
  margin-right: 40px;
}
```

Uso da propriedade background-size-cover - Necessária para a imagem cobrir todo o container

```
.content {
  background: url("../img/header-3.jpg")
no-repeat right;
  width: 100%;
  height: 110%;
  background-size: cover;
  box-sizing: border-box;
  padding-top: 0;
  padding-bottom: 0;
  display: inline-block;
  position: absolute;
  top: 0;
}
```

Javascript -

Uso de função anônima para criar um listerner que executa a função toda vez que clica no menu responsivo**(sandwich)**. Se tiver a classe *opened* então ela será retirada para ter a animação de fechar. Caso esteja sem a classe então a mesma será adicionada e o css fará a animação.

```
document.getElementById('sandwich').onc
lick = function(){
        var menu =
document.getElementById('menu-full
'),
        sandwich =
document.getElementById('sandwich
');
```

```
        if (menu.className ===
'menu-full opened') {
            menu.className =
'menu-full';
            sandwich.className =
'icon';
        } else {
            menu.className =
'menu-full opened';
            sandwich.className = 'icon
opened';
        }
}
```

7 - RESULTADOS E RECOMENDAÇÕES

Como conclusão do trabalho, foi obtido êxito na construção de um site responsivo utilizando as tecnologias client-side e boas práticas para a construção do layout proposto.

Baseando-se no reuso de classes, para mais de uma página html, e o uso de recomendações atuais para programação para web obtivemos êxito no aprendizado oferecido pela avaliação. Como melhora futura, é previsto ser inserido mais conteúdos dentro da seção de projetos na página principal.