

Programação Web para Jogos



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



HTML, CSS e JavaScript para Jogos

Responsável pelo Conteúdo:
Prof. Me. Alcides Teixeira Barboza Junior

Revisão Textual:
Prof.^a Esp. Kelciane da Rocha Campos

UNIDADE

HTML, CSS e JavaScript para Jogos



- Contextualização;
- O que é um IDE e Para que Serve?
- Introdução ao HTML;
- Introdução ao CSS;
- Programação em JavaScript – Conceitos Iniciais;
- Desenvolvendo um Jogo em HTML, CSS e JavaScript.



OBJETIVO DE APRENDIZADO

- Demonstrar o uso de uma ferramenta para desenvolvimento de códigos HTML, CSS e JavaScript;
- Apresentar os conceitos fundamentais da linguagem HTML e CSS;
- Apresentar os conceitos iniciais da linguagem JavaScript para a elaboração de um jogo;
- Desenvolver um jogo simples utilizando as tecnologias Web apresentadas.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:

Determine um horário fixo para estudar.

Mantenha o foco! Evite se distrair com as redes sociais.

Procure manter contato com seus colegas e tutores para trocar ideias! Isso amplia a aprendizagem.

Seja original! Nunca plágie trabalhos.

Aproveite as indicações de Material Complementar.

Conserve seu material e local de estudos sempre organizados.

Não se esqueça de se alimentar e de se manter hidratado.

Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Contextualização

Você já jogou diversos jogos por meio do navegador, certo? Alguns mais complexos e outros mais simples para passar o tempo. A questão é: como podemos começar a entender e desenvolver jogos que irão ser jogados via Web por meio de um navegador? Podemos pensar em primeiro lugar em uma engine como Unity, porém existem diferentes maneiras de desenvolver um jogo, principalmente para Web.

Note que nossa disciplina se chama Programação Web para Jogos, logo iremos utilizar a programação para desenvolver os jogos. Pensando em Web, e como foi comentado na Unidade 1, existem três tecnologias básicas, as quais são processadas por um navegador na máquina cliente. Você lembra quais são essas tecnologias?

Se você lembrou, deve ter pensado em HTML, CSS e JavaScript.

Nesta unidade, iremos conhecer os princípios dessas três tecnologias ou linguagens voltadas para o desenvolvimento de jogos para Web; o objetivo principal é desenvolver um jogo simples de corrida utilizando o básico das três tecnologias. Embora seja um jogo considerado simples, em relação ao tamanho do código, ele irá envolver diferentes conceitos de programação.

Vamos começar!

O que é um IDE e Para que Serve?

Quando ingressamos no mundo da programação, além de estudarmos as linguagens existentes e decidirmos qual dessas linguagens escolheremos para trabalharmos, existem dezenas de termos, tecnologias e outros aspectos que precisamos conhecer e estudar.

Considerando que você está decidido(a) a expandir seus conhecimentos para ingressar na carreira de desenvolvimento de jogos, vamos começar entendendo o que é um IDE (Integrated Development Environment).

Um IDE é um ambiente de desenvolvimento integrado, ou seja, é um programa que possui várias ferramentas e recursos integrados que são essenciais para o desenvolvimento de um software. Esses programas facilitam a execução de testes, acesso a banco de dados, geralmente permitem gerar alguns trechos de códigos de forma automática para padrões de determinadas linguagens, executam o compilador de uma determinada linguagem, depuram o código, entre vários outros recursos.

Em geral, todos esses recursos podem variar de acordo com o IDE escolhido, sua configuração e principalmente a linguagem à qual se destina.

Existem muitos IDEs e entre os principais e mais usados estão NetBeans e Eclipse, para trabalhar com desenvolvimento de aplicações em Java, por exemplo; Visual Studio, para trabalhar com tecnologias Microsoft; Android Studio, para desenvolvimento de aplicativos Android; etc.

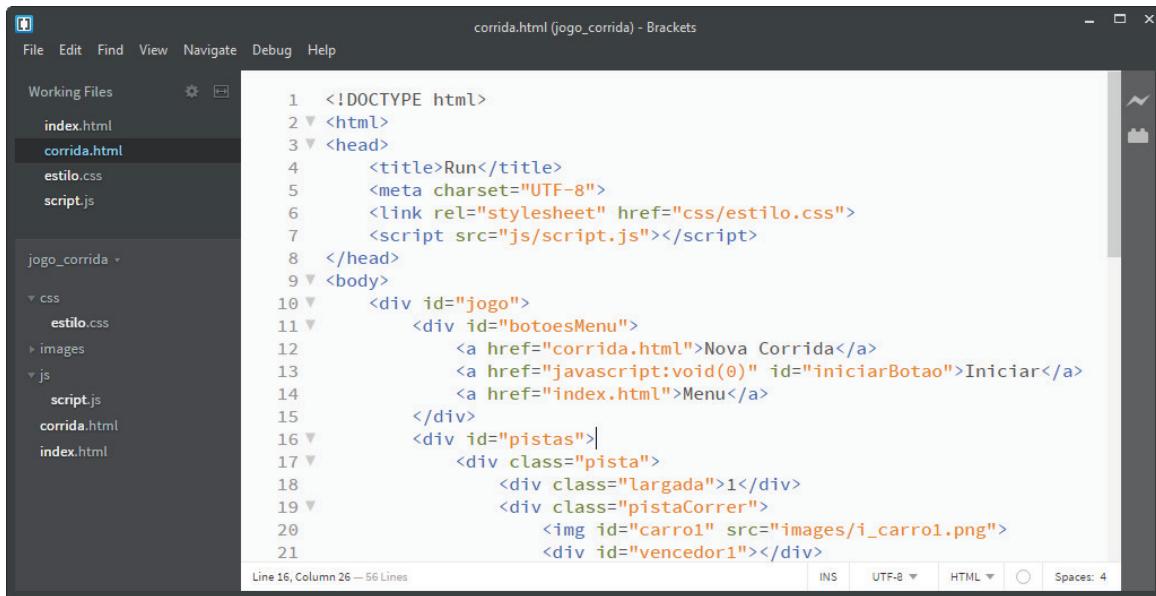
Para o desenvolvimento de aplicações focadas em Web, temos, além dos IDEs já citados, editores de texto puro, que visam à edição do código e fornecem algum suporte para a digitação desses códigos apresentando a sintaxe da linguagem que está sendo utilizada, alguns exemplos seriam: NotePad++, Sublime Text, Brackets, Atom, entre outros.

Nós utilizaremos o Brackets (Figura 1), inclusive esse programa foi criado e é mantido pela Adobe.



Figura 1 – Ícone do programa Brackets

O ambiente de desenvolvimento é simples, porém a ferramenta permite adicionarmos diversos recursos por meio de extensões. A Figura 2 exibe a interface da ferramenta.



The screenshot shows the Brackets IDE interface. The title bar says "corrida.html (jogo_corrida) - Brackets". The menu bar includes File, Edit, Find, View, Navigate, Debug, and Help. The left sidebar shows "Working Files" with files: index.html, corrida.html, estilos.css, script.js, and a folder "jogo_corrida" containing index.html, estilos.css, images, and script.js. The main code editor window displays the following HTML code:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Run</title>
5      <meta charset="UTF-8">
6      <link rel="stylesheet" href="css/estilo.css">
7      <script src="js/script.js"></script>
8  </head>
9  <body>
10 <div id="jogo">
11     <div id="botoesMenu">
12         <a href="corrida.html">Nova Corrida</a>
13         <a href="javascript:void(0)" id="iniciarBotao">Iniciar</a>
14         <a href="index.html">Menu</a>
15     </div>
16     <div id="pistas">
17         <div class="pista">
18             <div class="largada">1</div>
19             <div class="pistaCorrer">
20                 
21                 <div id="vencedor1"></div>

```

The status bar at the bottom shows "Line 16, Column 26 — 56 Lines" and buttons for INS, UTF-8, HTML, and Spaces: 4.

Figura 2 – Interface do Brackets

Como mencionado, o Brackets é uma ferramenta desenvolvida pela Adobe, que utiliza a metodologia de código orientado no design, ou seja, uma ferramenta focada em programação web. E é por esse motivo que iremos estudar utilizando esse ambiente de desenvolvimento, afinal até o fim desta unidade, você terá aprendido a desenvolver um jogo para web, utilizando as linguagens HTML, CSS e JavaScript.



Para baixar e instalar o Brackets, acesse: <https://goo.gl/tZQpSy>.



O processo de instalação pode ser consultado em: <https://youtu.be/xWT3ezzJFLU>, contudo é um processo simples, basta baixar o programa, executar o arquivo baixado e avançar nas telas de instalação sem que haja a necessidade de alterações.

Sugerimos que você instale algumas extensões no Brackets. Para isso, clique no ícone semelhante a uma peça do brinquedo Lego, geralmente esse ícone está na barra de ferramentas do lado direito. Se preferir, clique em *File >> Extension Manager*.

A tela de instalação possui algumas abas, veja na Figura 3 um exemplo das extensões instaladas.

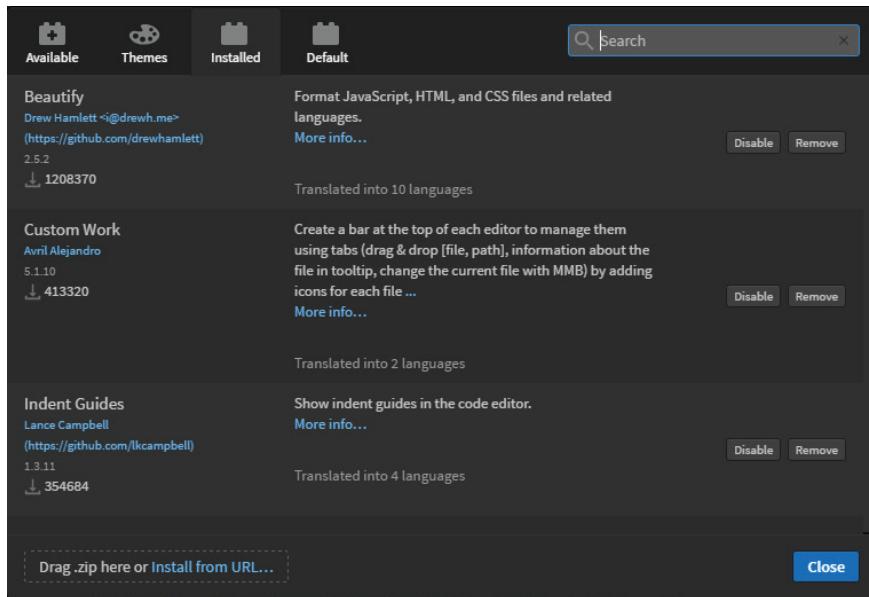


Figura 3 – Sugestão de extensões para o Brackets

A Figura 3 apresenta as extensões que instalamos em nossa ferramenta, fique à vontade para instalar essas ou outras que achar interessante. Para instalar extensões, clique na aba Available, no campo de pesquisa insira o nome da extensão ou do recurso de que gostaria e em seguida clique em Install.

Como você percebeu, na aba Installed, você poderá visualizar o que foi instalado e remover caso necessário.

Visualizando Nossos Arquivos

Um arquivo HTML pode ser aberto para visualização em qualquer navegador, usamos geralmente para os exemplos o Google Chrome ou o Firefox.

Para abrir o arquivo HTML, basta dar um duplo clique no arquivo que possui a extensão .html ou, então, com o botão direito do mouse escolher Abrir Com... e escolher algum navegador que você tenha instalado no seu computador.

Existe também a possibilidade de abrir o arquivo HTML pelo Brackets, para isso você deve clicar no relâmpago perto do símbolo do brinquedo da Lego. Esse relâmpago irá simular um servidor web no seu computador para então abrir o arquivo como se estivesse hospedado em algum servidor. Em alguns casos, esse método será interessante, mas, por enquanto, a primeira forma de abrir o arquivo HTML já irá nos ajudar.

Você perceberá que o HTML é a porta de entrada para o CSS e o JavaScript, por isso sempre executamos o arquivo .html primeiro.

Introdução ao HTML

Vamos relembrar qual foi a definição da linguagem HTML na unidade anterior?

O HTML (Hypertext Markup Language, em português Linguagem de Marcação de Hipertexto), como o nome já diz, é uma linguagem utilizada para estruturar ou marcar todo o conteúdo de uma página HTML, definindo elementos como parágrafos, imagens, listas, links, etc.

A linguagem HTML possui um conjunto de elementos comumente chamados de *tags*. As tags são responsáveis pela marcação dos conteúdos, cada uma possuindo uma funcionalidade.

Uma tag é iniciada com o sinal de “<” seguido pelo nome da tag e encerrada com o sinal de “>”. Por exemplo, para definirmos um parágrafo em HTML, utilizamos a tag <p>.

Após a abertura da tag, vem o conteúdo a ser inserido - por isso o HTML é uma linguagem de marcação - e geralmente, finalizamos a tag com seu fechamento, que segue o mesmo padrão de abertura, porém adicionando uma “/” antes do nome da tag. Por exemplo: </p>.

Existem algumas tags em HTML que não possuem o fechamento padrão descrito anteriormente, como, por exemplo a
 (efetua a quebra de linha), <hr> (insere uma linha horizontal para separação de seções), <meta> (insere alguns metadados), entre outras. Nestes casos, para utilizar a tag, basta colocar sua abertura.

Todo arquivo HTML irá possuir uma estrutura padrão mínima, composta principalmente por duas regiões principais, o cabeçalho com a tag <head> e o corpo da página com a tag <body>, veja na Figura 4 essa estrutura mínima.

```
1  <!DOCTYPE html>
2 ▼ <html>
3 ▼   <head>
4     <meta charset="UTF-8"/>
5     <title>Título da página</title>
6   </head>
7 ▼   <body>
8     <!-- Conteúdo -->
9   </body>
10 </html>
```

Figura 4 – Estrutura mínima do HTML

Na linha 1 inserimos a definição DOCTYPE. Deve ser sempre a primeira a ser digitada em um documento HTML. Ela indica para o navegador qual versão do HTML estamos usando; neste exemplo, estamos seguindo o padrão HTML5.

Nas linhas 2 e 10 inserimos a abertura e fechamento da tag <html>, que delimita o documento todo. Assim, todas as demais tags da página devem estar dentro desse bloco, ou seja, dentro do <html>...</html>.

No cabeçalho (linha 3 a 6), inserimos algumas configurações como o título da página e a codificação de caracteres. As tags inseridas nesse bloco não serão visualizadas diretamente no conteúdo, mas possuem diversas instruções úteis para a página.

Dentro dessa tag, por exemplo, inserimos na 5 a tag <title>, que define o título que aparece na aba do navegador. Na linha 4 inserimos a tag <meta>, que especifica qual conjunto de caracteres será usado para renderizar o texto na página. A definição UTF-8 contém todos os caracteres dos padrões referentes ao nosso idioma, que é o europeu ocidental.

Ressaltamos que dentro do cabeçalho podemos ainda linkar os arquivos de folha de estilo (CSS) e scripts (JavaScript).

No corpo da página (linha 7 a 9), devemos inserir tudo aquilo que servirá de estrutura para o usuário visualizar no seu navegador nosso conteúdo ou jogo. Na linha 8 foi adicionada a sintaxe de como adicionar um comentário nos documentos HTML. Esse comentário não é visualizado no navegador.

Vejamos a seguir um exemplo prático utilizando a estrutura mínima e o uso de uma tag de conteúdo, a tag de parágrafo (Figura 5).

```

1  <!DOCTYPE html>
2 ▼ <html>
3 ▼ <head>
4    <title>Título</title>
5    <meta charset="UTF-8">
6  </head>
7 ▼ <body>
8    <p>Texto inserido como conteúdo.</p>
9  </body>
10 </html>
```

Figura 5 – Estrutura HTML mínima e uso da tag de parágrafo

Algumas tags, como a utilizada para inserir uma imagem na página , não possuem o fechamento padrão como comentamos anteriormente, nestes casos basta colocar a abertura e suas configurações.

Aproveitando que citamos a tag , é importante destacar que algumas tags podem receber atributos, mais conhecidos como atributos da tag. A Figura 6 demonstra um exemplo do uso da tag que insere uma imagem na página.

```
1  <!DOCTYPE html>
2 ▼ <html>
3 ▼ <head>
4      <title>Exemplo</title>
5      <meta charset="UTF-8">
6  </head>
7 ▼ <body>
8
9      
10
11 </body>
12 </html>
```

Figura 6 – Tag de imagem com atributo src definindo uma imagem

Perceba na Figura 6 que foi adicionado o atributo src, ele serve para indicar qual imagem a tag irá carregar na página.

A partir do momento em que inserimos os conteúdos dentro da tag <body>, como imagens, textos, etc. e abrirmos a página no navegador, percebemos que se quisermos deixar essa página visualmente estilizada ou, em outras palavras, formatar o visual da página, será necessário usarmos uma linguagem separada do HTML. Neste ponto, entra em ação o CSS (Cascading Style Sheets, em português Folhas de Estilo em Cascata).



Um bom site para você encontrar tudo de HTML é o W3School, acesse: <https://goo.gl/USxasu>. Site com materiais de HTML MDN web docs (Mozilla), endereço: <https://goo.gl/DrLHYh>.

Introdução ao CSS

Você já teve uma breve definição do CSS na primeira unidade desta disciplina, mas chegou a hora de conhecer melhor essa linguagem para podermos formatar visualmente nosso futuro jogo, que será desenvolvido mais adiante.

A seguir, iremos conhecer a sintaxe do CSS, seus principais seletores e como inserir o arquivo de extensão .css dentro do nosso documento HTML.

Conforme a Figura 7, quando vamos declarar uma regra de estilo para um elemento, usamos um seletor; neste caso optamos por usar o seletor de elemento que nos permite, neste exemplo, configurar a tag <p>.



Figura 7 – Sintaxe de uma configuração CSS

Cada bloco de seletor é delimitado por chaves, dentro das chaves podemos utilizar diversas propriedades do CSS. No exemplo exibido na Figura 7, utilizamos a propriedade color, que aplica uma cor ao texto. Perceba que separamos a propriedade do valor dela com dois pontos (:). Note também que no final da linha, devemos encerrar a configuração da propriedade com ponto e vírgula (;).



Importante!

Insira sempre um ponto e vírgula no final da linha de cada propriedade.

Assim, podemos definir a sintaxe básica do CSS como:

```
tipo_do_seletor {  

  propriedade1 : valor_da_propriedade1;  

  propriedade2 : valor_da_propriedade2;  

  ...  

}
```

Inserindo o CSS em Uma Página HTML

Existem três maneiras de inserir o código do CSS no nosso documento HTML. Essas três maneiras são:

- CSS externo
- CSS incorporado
- CSS inline

O CSS inline (Figura 8) é umas das maneiras mais básicas de aplicar os estilos utilizando o atributo “style” diretamente nos elementos do HTML. Nossa dica é que você evite utilizar esse formato, pois ele mistura formatação visual com a estrutura, isso não é uma boa prática no desenvolvimento Web.

```
1  <!DOCTYPE html>
2 ▼ <html>
3 ▼ <head>
4      <title>Exemplo</title>
5      <meta charset="UTF-8">
6  </head>
7 ▼ <body>
8
9      <p style="color:blue;">Texto de exemplo</p>
10
11 </body>
12 </html>
```

Figura 8 – Exemplo de CSS inline

Já o CSS incorporado (Figura 9) faz uso da tag `<style>...</style>` inserida obrigatoriamente dentro do bloco de cabeçalho do HTML (`<head>...</head>`). Esse formato de inserir CSS na página é válido e pode ser utilizado em alguns casos em que o CSS externo não seja usual.

```
1  <!DOCTYPE html>
2 ▼ <html>
3 ▼ <head>
4      <title>Exemplo</title>
5      <meta charset="UTF-8">
6 ▼     <style>
7
8 ▼         p{
9             color: red;
10            }
11
12     </style>
13 </head>
14 ▼ <body>
15
16
17 </body>
18 </html>
```

Figura 9 – Exemplo de CSS incorporado

Por fim, temos a maneira mais utilizada e à qual você deve dar preferência nos seus projetos, que seria o CSS externo. Neste formato, devemos criar um arquivo .css com as nossas configurações e posteriormente linkar este arquivo .css com nossos arquivos .html. Esse processo de linkar os arquivos é feito no <head> dos arquivos HTML.

Veja na Figura 10 um exemplo de arquivo .css e como se faz o link deste arquivo no HTML.

estilo.css

```

1 ▼ p{
2     color: yellow;
3 }
```

index.html

```

1  <!DOCTYPE html>
2 ▼ <html>
3 ▼ <head>
4     <title>Exemplo</title>
5     <meta charset="UTF-8">
6
7     <link rel="stylesheet" href="estilo.css">
8
9 </head>
10 ▼ <body>
11
12
13 </body>
14 </html>
```

Figura 10 – Exemplo de CSS externo REFAZER.

No arquivo .css, criamos normalmente as configurações seguindo a sintaxe básica já apresentada. Já no arquivo .html, devemos “linkar” o CSS externo com a tag <link>, preste atenção na linha 7, referente a esse elemento.

Perceba que você tem disponíveis três possibilidades, contudo, como já falamos, dê preferência ao CSS externo, depois ao CSS incorporado e evite sempre o inline.

Em alguns projetos, você também sentirá a necessidade de mesclar o externo com o incorporado, não é errado, você pode fazer isso tomando cuidado com a ideia de cascata do CSS, ou seja, dependendo de como você carrega o arquivo externo e cria o CSS incorporado, uma propriedade pode ser sobreescrita, fique atento(a). A Figura 11 exibe um exemplo de um arquivo HTML que faz uso de um CSS externo e um CSS incorporado.

```
1  <!DOCTYPE html>
2 ▼ <html>
3 ▼ <head>
4      <title>Exemplo</title>
5      <meta charset="UTF-8">
6
7      <link rel="stylesheet" href="estilo.css">
8      <link rel="stylesheet" href="estilo2.css">
9
10 ▼   <style>
11 ▼     div{
12         width: 300px;
13         background-color: yellow;
14     }
15   </style>
16
17 </head>
18 ▼ <body>
19
20
21 </body>
22 </html>
```

Figura 11 – Exemplo de CSS externo e incorporado no HTML

Principais Seletores do CSS

Quando vamos utilizar o CSS, devemos escolher um tipo de seletor que melhor se encaixe às nossas necessidades e fazer as devidas configurações de propriedades. Existem mais de cinco tipos de seletores e diversas combinações que podemos fazer, vamos citar aqui somente o necessário para entendermos os conceitos iniciais, mas consulte as indicações de links para se aprofundar.

Uma primeira opção seria o seletor tipo, este seletor basicamente estipula qual tag (elemento) do HTML queremos usar e permite utilizar praticamente todas as tags. A sintaxe desse seletor é:

```
elemento { propriedade : valor; }
```

Perceba que o seletor tipo configura o CSS para uma tag específica, logo em qualquer ponto do HTML que for utilizada a tag será aplicado o estilo. Você deve se perguntar: não existe uma forma de controlar o uso do estilo e também usar um mesmo estilo em diferentes pontos? A resposta é existe.

Para criarmos estilos CSS que só serão aplicados quando forem chamados e que possam ser aplicados em diferentes tags, temos disponível o seletor classe e o seletor id.

O seletor classe permite criar uma classe genérica que pode ser utilizada em qualquer tag diversas vezes. Já o seletor id pode ser utilizado somente uma vez no HTML, a principal função deste seletor é identificar uma tag para ser chamada no JavaScript.

A sintaxe para criar uma classe no CSS é bem simples, só fique atento(a) para começar a definição com um ponto “.” seguido pelo nome da classe sem espaços e sem caracteres especiais:

```
.nome_da_classe { propriedade : valor; }
```

O seletor id é muito parecido, contudo se inicia a sua criação com o símbolo “#” seguido pelo nome do estilo individual. A sintaxe desse seletor é:

```
#nome_do_id { propriedade : valor; }
```

Como a ideia é criar esses estilos e aplicar onde for necessário, devemos chamar cada um deles de um jeito específico no HTML. Para aplicar uma classe do CSS em uma tag, utilizamos o atributo class. Já para aplicar um estilo id, utilizamos o atributo id.

A Figura 12 apresenta a criação de um estilo individual chamado “pistas” e uma classe chamada “vencedor”.

```

▼ #pistas{
    width: 780px;
    height: 368px;
    background-color: #555;
    border-top: 2px solid #dcd400;
    border-right: 2px solid #dcd400;
    border-left: 2px solid #dcd400;
    margin: 45px 0;
}

▼ .vencedor {
    font-family: Arial;
    font-size: 30px;
    color: #b53f3f;
    text-align: center;
    text-shadow: 0 0 10px #000;
}
  
```

Figura 12 – Exemplo de uso do class e do id

A ideia até aqui foi apresentar os conceitos iniciais do HTML e do CSS, você precisa primeiro entender o seu uso; após esta etapa, basta consultar referências das linguagens para descobrir quais são as diferentes tags do HTML e as diversas propriedades do CSS.

Consulte os materiais complementares para continuar os estudos.



A lista oficial completa de seletores do CSS pode ser encontrada em:
<https://goo.gl/45kcdn> ou <https://goo.gl/ZFCCpD>.

Para consultar o material de CSS da W3Schools, acesse: <https://goo.gl/JZECMA>.

Material de CSS da MDN web docs (Mozilla) está disponível em <https://goo.gl/Lm4zo3>.

Programação em JavaScript – Conceitos Iniciais

Antes de começarmos a desenvolver nosso jogo, precisamos conhecer o JavaScript. O nome pode fazer você confundir essa linguagem script com a linguagem Java, mas não são iguais. Cada uma tem suas aplicações, nosso foco aqui será o JavaScript (JS).

O JS é uma linguagem interpretada por um navegador web como o Chrome, logo é processada na máquina cliente. Essa linguagem recebeu diversas modificações, mas foi depois das mudanças ocasionadas pelo surgimento do HTML5 que ganhou mais o olhar dos desenvolvedores.

Como se trata de uma linguagem de programação, devemos fazer uso da lógica e algoritmos para estruturar ou criar nossos códigos. Nós vamos conhecer alguns dos seus comandos durante a criação do nosso jogo, agora precisamos somente entender como inserir essa programação em uma página HTML.

A programação em JavaScript dará funcionalidades para nosso jogo, tratando os eventos ou ações que o jogador desempenhar, como, por exemplo, apertar um botão para fazer um personagem correr, clicar para disparar um tiro, etc.

As formas de inserir o JS na página são semelhantes às formas do CSS. É possível desenvolver o script incorporado na página ou em um arquivo externo com a extensão .js.

O JavaScript incorporado pode ser inserido no cabeçalho da página (tag head) ou no corpo da página (tag body), depende do seu estilo de programação e/ou necessidades. A Figura 13 apresenta essa forma de inserir o JS.

```
1  <!DOCTYPE html>
2 ▼ <html>
3 ▼ <head>
4      <title>Exemplo</title>
5      <meta charset="UTF-8">
6
7 ▼     <script>
8         alert("Olá, estou no cabeçalho.");
9     </script>
10
11    </head>
12 ▼ <body>
13
14 ▼     <script>
15         alert("Olá, estou no corpo da página.");
16     </script>
17
18    </body>
19 </html>
```

Figura 13 – JavaScript incorporado (head e body)

Perceba que nos dois locais o JS foi inserido com o auxílio da tag <script>...</script>. Embora essa opção de inserir o JS funcione e seja utilizada em diversos tipos de aplicações, o reuso do código se limita somente à página na qual foi inserido, em alguns casos seria interessante inserir um script que pudesse ser utilizado em diferentes páginas, para isso utilizamos o arquivo externo .js.

A Figura 14 exibe a sintaxe de um arquivo .js básico com um comando de exemplo e em seguida como devemos chamar esse arquivo dentro do HTML para que ele seja executado.

funcoes.js

```

1  alert("Olá, estou em um arquivo .js externo.");
2
3

```

index.html

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Exemplo</title>
5      <meta charset="UTF-8">
6
7      <script src="funcoes.js"></script>
8
9  </head>
10 <body>
11
12 </body>
13 </html>

```

Figura 14 – JavaScript externo

Veja que neste exemplo também utilizamos a tag <script>, porém dentro dela foi inserido o atributo src e o valor dele é igual ao nome do arquivo que queremos chamar.



Nunca utilize o mesmo bloco de script para chamar um arquivo externo e inserir seu código incorporado, isso não funciona. Caso precise inserir um arquivo externo e depois colocar um código incorporado, utilize mais de um bloco de script.
Um mesmo arquivo HTML pode fazer uso de diversos arquivos CSS e diversos arquivos JS. Quando você se deparar com essa necessidade, lembre-se de inserir cada arquivo CSS com uma tag <link> e cada arquivo JS com uma tag <script>.

Acesse o material da W3Schools sobre JavaScript para obter mais conhecimento sobre a linguagem.



Outro material interessante de JavaScript para você consultar é o MDN web docs (Mozilla) disponível em: <https://goo.gl/uxHhqA>.

Desenvolvendo um Jogo em HTML, CSS e JavaScript

Chegamos ao momento de juntar as tecnologias apresentadas e com isso desenvolver um jogo para Web.

O jogo que iremos desenvolver servirá para demonstrar o uso do HTML, a formatação com o CSS e as funcionalidades em JavaScript. Vamos desenvolver um jogo de corrida que consiste em mover 4 carros de um ponto a outro da tela com valores aleatórios de descolamento horizontal. Para ficar interessante, iremos utilizar algumas imagens e CSS para formatar o visual. Veja o resultado que iremos obter ao final do projeto na Figura 15.

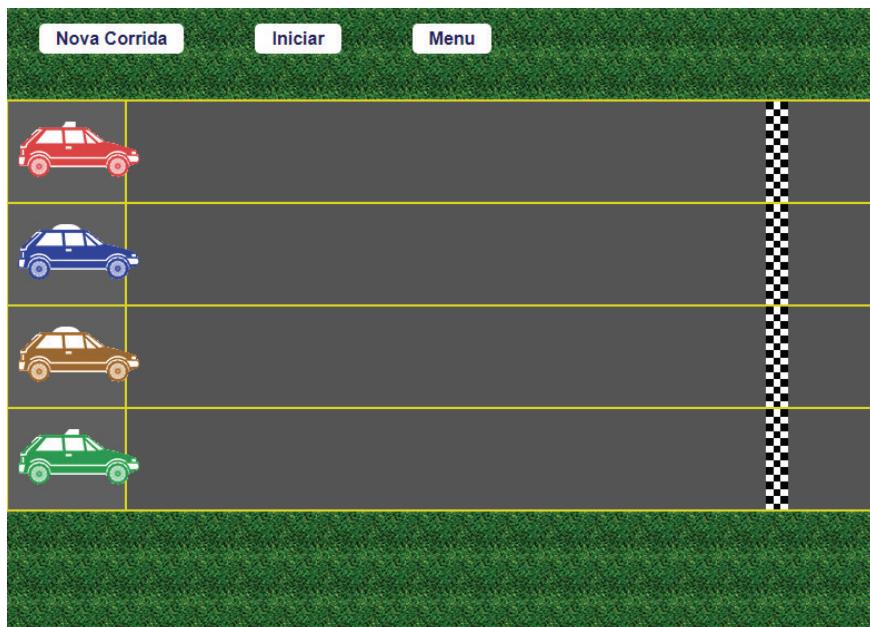


Figura 15 – Jogo de corrida: Run



Importante!

Para você seguir com os passos apresentados nesta seção, terá que fazer o download do material extra disponível no ambiente Blackboard. O material consiste basicamente das imagens utilizadas nos exemplos.

Com os materiais baixados previamente do Blackboard, abra a pasta principal já descompactada no Brackets, para isso clique com o botão direito do mouse e escolha Open as Brackets.

Analise a estrutura de pastas e arquivos fornecida, a Figura 16 apresenta essa estrutura, verifique se a sua está exatamente igual para que possamos dar início ao desenvolvimento.

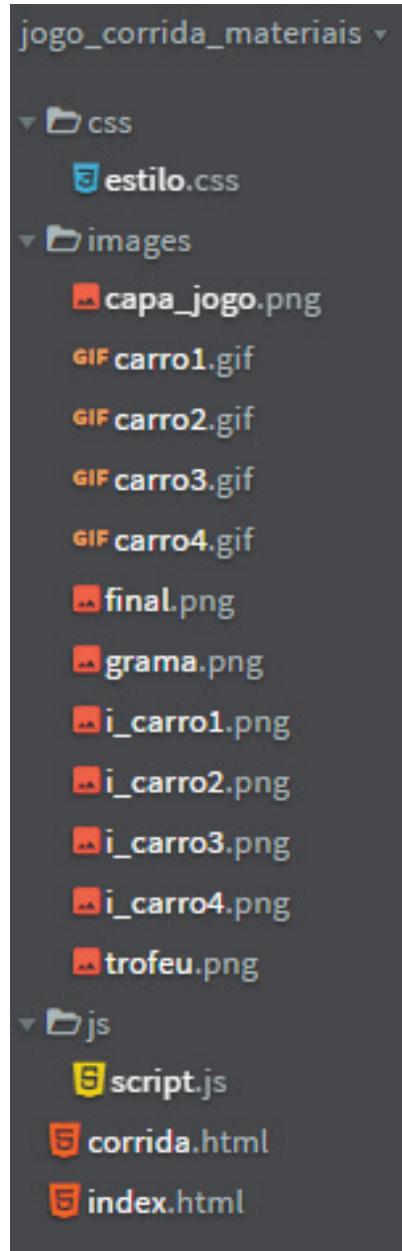


Figura 16 – Estrutura de pastas e arquivos do jogo Run

Você também pode abrir a pasta fornecida clicando em File > Open Folder. Com o nosso projeto aberto no IDE, caso precise criar um arquivo novo, clique em File > New e o Brackets abrirá um arquivo em branco.



Sempre que criar um arquivo novo, salve-o antes de começar a fazer seu código. Salvando com a extensão correta, o IDE irá auxiliar mostrando um help da linguagem conforme a extensão utilizada.

Após realizar alterações nos seus códigos, salve o arquivo pressionando Ctrl+S ou clicando no menu File>Save. Verifique sempre o tipo do arquivo com que irá trabalhar para salvar com a extensão correta, as extensões podem ser .html, .css ou .js..

Como já fornecemos os arquivos criados, com a pasta aberta no Brackets, dê um duplo clique no arquivo index.html para começar a digitar. Perceba que no arquivo já existe a estrutura básica. Vamos inserir dentro da tag body o conteúdo que será responsável por criar a capa do jogo ou tela inicial. A Figura 17 exibe o resultado final.



Figura 17 – Tela inicial do jogo Run

O código do arquivo index.html é exibido na Figura 18, copie esse código no seu arquivo.

A captura de tela mostra o Brackets editor com o arquivo index.html aberto. O código é o seguinte:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="UTF-8">
5     <title>Run</title>
6     <meta charset="UTF-8">
7 </head>
8 <body>
9 <div id="capa">
10    <a href="corrida.html" id="iniciarJogo">Iniciar jogo</a>
11 </div>
12 </body>
13 </html>
```

Figura 18 – Estrutura do documento index.html

Na linha 9, inserimos a tag <div>, que normalmente define um bloco ou caixa no documento HTML. Lembre-se sempre de fechar uma tag <div> que foi aberta, conforme foi feito na linha 11. Perceba que para essa div, nós adicionamos um atributo id com o nome de capa. Este atributo define um nome exclusivo para uma tag, ou seja, o nome que for especificado no atributo id não poderá ser usado em mais nenhum momento neste documento HTML. Lógico que poderemos formatar este id em CSS.

Dentro da div com id capa, inserimos na linha 10 a tag <a>, que define um hiperlink. Essa tag é usada para vincular uma página à outra, ou seja, quando o usuário clicar neste link, irá abrir a página especificada.

Para especificar o arquivo que será aberto ao clicar no link, usamos o atributo href. Assim, quando clicarmos neste hiperlink, seremos direcionados à página com o nome de corrida.html (será a página em que iremos programar o jogo). O nome atribuído ao atributo id foi iniciarJogo. Dentro da tag <a> colocamos o texto “Iniciar Jogo”.



Importante!

Tome cuidado com letras maiúsculas e minúsculas nos nomes de classes, ids e comandos do Javascript, fique atento(a) aos exemplos.

Vamos testar nosso arquivo de capa do jogo abrindo-o em um navegador? Vá no explorador de arquivos e abra a pasta do jogo. Feito isso, encontre o documento index.html e dê um duplo clique ou então clique com o botão direito do mouse, vá em “Abrir com” e peça para abrir este arquivo no navegador Chrome ou em um navegador de sua preferência. Lembre-se de que também podemos abrir o arquivo pelo Brackets clicando no relâmpago.

Sua página deve aparecer conforme a Figura 19.

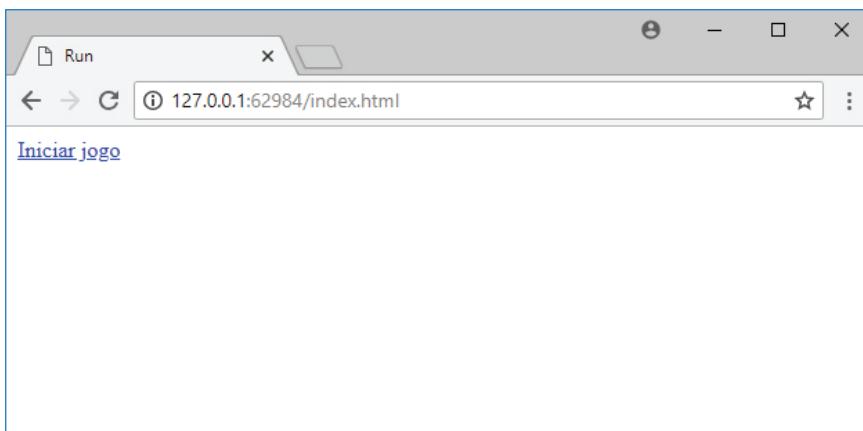


Figura 19 – Documento index.html rodando no navegador

Você deve ter notado que o resultado no navegador não se parece com nossa proposta de capa inicial. A linguagem CSS é utilizada exatamente para dar estilo a um documento HTML e deixar a entrada do nosso jogo mais interessante e conforme a ideia proposta anteriormente.

Para criar o CSS, abra o arquivo estilo.css que está dentro da pasta CSS. Antes de começar a digitar os estilos, lembre-se de que é necessário indicar a ligação do nosso arquivo estilo.css dentro do documento HTML.

Adicione a linha <link rel="stylesheet" href="css/estilo.css"> dentro da nossa tag <head>, essa tag link indica para o HTML que deverá ser carregado o arquivo estilo.css para formatar a página.

Agora que já temos a ligação do arquivo CSS com o arquivo HTML, iremos criar nossos estilos para a capa do jogo.

A Figura 20 apresenta o código necessário para formatar visualmente a tela inicial ou capa do jogo. Vamos digitar esses estilos no nosso arquivo estilo.css para depois entender as propriedades. Uma boa ideia seria fazer um bloco de estilo, salvar o arquivo (Ctrl+S) e visualizar no seu navegador, com isso você verá passo a passo a evolução do que está criando.

```
1 ▼ body{
2     margin: 10px;
3 }
4
5 ▼ #capa{
6     background-image: url(..../images/capa_jogo.png);
7     width: 774px;
8     height: 447px;
9     margin: auto;
10}
11
12 ▼ #iniciarJogo{
13     background: #f3f3fc;
14     font-family: Arial;
15     font-size: 20px;
16     text-decoration: none;
17     padding: 5px;
18     float: left;
19     margin: 390px 575px 0;
20     width: 150px;
21     display: block;
22     text-align: center;
23     color: #bf1135;
24     border-radius: 110px;
25     font-weight: bold;
26 }
27
28 ▼ #iniciarJogo:hover{
29     background: #bf1135;
30     color:#f3f3fc;
31 }
```

Figura 20 – Estilos definidos para as tags da tela inicial do jogo Run

Na linha 2, definimos uma margem de 10px para a tag <body>, você irá perceber que existe um espaço na parte superior da imagem da capa.

Para nosso id #capa, definimos uma imagem de fundo com o arquivo capa_jogo.png localizado na pasta images (linha 6), usamos a propriedade background-image. Junto à imagem de fundo, será necessário definir uma largura (propriedade width)

e altura (propriedade height) para a caixa que contém a imagem de fundo. Iremos deixar a caixa (tag div) centralizada usando a propriedade margin com o valor auto, esse valor seria automático.

Finalmente, temos o botão de Iniciar Jogo posicionado ao lado inferior direito da div principal #capa. Neste botão, definimos a cor de fundo branca (hexadecimal do branco é #fff); o tipo da fonte para Arial (linha 14); tamanho da fonte de 20 pixels (linha 15); tiramos o estilo padrão que o HTML dá ao hyperlink (linha 16); adicionamos um espaçamento interno do conteúdo para as bordas da caixa do link (linha 17); posicionamos o botão para a esquerda (linhas 18 e 19); colocamos margem, centralizamos o texto no centro da caixa de link (linha 22); definimos uma cor para o texto (linha 23); arredondamos os cantos do botão com a propriedade border-radius (linha 24); e deixamos o texto em negrito (**bold**) utilizando a propriedade font-weight (linha 25).

Agora, verifique o resultado no seu navegador e veja se está parecido com nossa proposta inicial da capa do jogo. Caso não esteja semelhante, revise seu código CSS novamente.

Um dos requisitos de um bom programador é ser curioso. Procure adicionar a pseudo-classe *hover* no seu botão de Iniciar Jogo conforme o que foi inserido nas linhas 28 a 30 e veja o que acontece quando o mouse estiver sobre este botão.

Devemos agora criar o visual do jogo no arquivo corrida.html, conforme Figura 21.

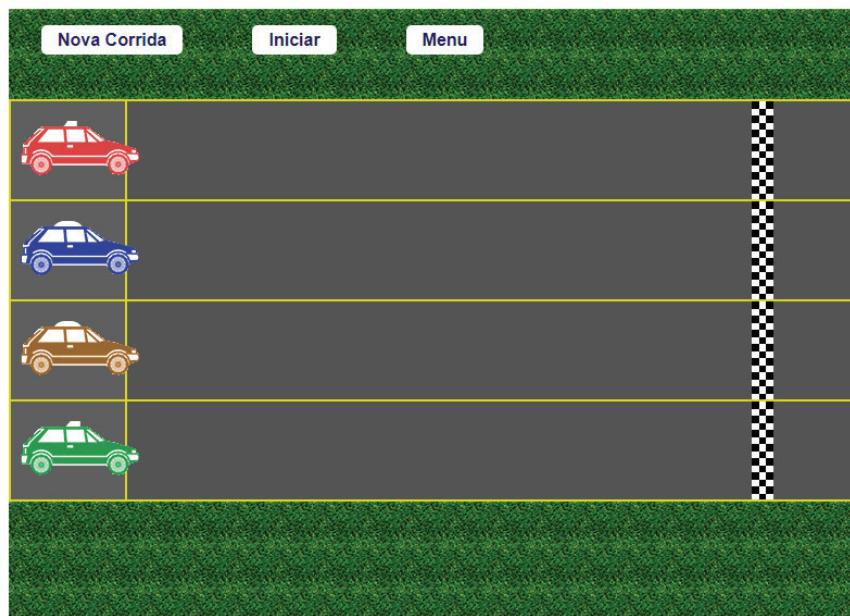


Figura 21 – Tela principal do jogo Run

Abra o arquivo corrida.html para inserir o código exibido na Figura 22.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Run</title>
5      <meta charset="UTF-8">
6      <link rel="stylesheet" href="css/estilo.css">
7  </head>
8
9  <body>
10 <div id="jogo">
11 <div id="botoesMenu">
12     <a href="corrida.html">Nova Corrida</a>
13     <a href="javascript:void(0)" id="iniciarBotao">Iniciar</a>
14     <a href="index.html">Menu</a>
15 </div>
16 <div id="pistas">
17 <div class="pista">
18     <div class="largada">1</div>
19 <div class="pistaCorrer">
20     
21     <div id="vencedor1"></div>
22 </div>
23     <div class="final"></div>
24 </div>
25
26 <div class="pista">
27     <div class="largada">2</div>
28 <div class="pistaCorrer">
29     
30     <div id="vencedor2"></div>
31 </div>
32     <div class="final"></div>
33 </div>
34
35 <div class="pista">
36     <div class="largada">3</div>
37 <div class="pistaCorrer">
38     
39     <div id="vencedor3"></div>
40 </div>
41     <div class="final"></div>
42 </div>
43
44 <div class="pista">
45     <div class="largada">4</div>
46 <div class="pistaCorrer">
47     
48     <div id="vencedor4"></div>
49 </div>
50     <div class="final"></div>
51 </div>
52 </div>
53
54 </div>
55 </body>
56 </html>
```

Figura 22 – Código HTML do arquivo corrida.html

Este código HTML é responsável por fornecer a estrutura para o JavaScript. Nas linhas 11 a 15 foram inseridos os botões:

- Nova Corrida: função de reiniciar o jogo na posição inicial;
- Iniciar: função de iniciar o jogo e fazer os carros se moverem;
- Menu: link para a tela inicial.

O bloco de div com início na linha 16 e término na linha 51 define a região das pistas para os carros. Dentro deste bloco, inserimos 4 pistas, uma para cada carro se deslocar individualmente. Cada bloco div com o id pista possui uma caixa para a região da largada; uma caixa para o carro se deslocar; uma caixa de mensagem para ser exibida caso o carro ganhe; e por fim, uma caixa que delimita a chegada. Note que os ids carroN e vencedor possuem números diferentes, isso porque essas tags serão manipuladas com o JavaScript, e lembre-se de que não podemos ter ids repetidos no HTML.

Para encerrarmos este arquivo, faça a referência ao arquivo script.js que está dentro da pasta js. Essa referência é feita com a tag <script src="js/script.js"></script> logo após a tag link do CSS. Pronto, finalizamos o HTML, porém você deve ter notado que seu arquivo não está com a mesma aparência do nosso, o que está faltando?

Se você pensou em CSS está correto. Vamos agora inserir o código CSS responsável por formatar a tela principal do jogo, esse código é exibido na Figura 23.

```

▼ #jogo{
    background-image: url(..../images/grama.png);
    width: 784px;
    height: 540px;
    margin: auto;
    padding-top: 20px;
    position: absolute;
    left: 0;
    right: 0;
}

▼ #botoesMenu a {
    text-decoration: none;
    color: #010d69;
    font-family: Arial;
    font-weight: bold;
    background-color: #fff;
    padding: 5px 15px;
    margin: 40px 30px;
    border-radius: 5px;
}

▼ #botoesMenu #iniciarBotao.ativado{
    pointer-events: none;
    opacity: .7;
}
  
```

Figura 23 – Código CSS para a tela principal do jogo Run

O CSS que formata as pistas é exibido na Figura 24.

```
▼ #pistas{
    width: 780px;
    height: 368px;
    background-color: #555;
    border-top: 2px solid #dcd400;
    border-right: 2px solid #dcd400;
    border-left: 2px solid #dcd400;
    margin: 45px 0;
}

▼ #pistas .largada {
    width: 105px;
    height: 90px;
    background-color: #5f5f5f;
    border-right: 2px solid #dcd400;
    border-bottom: 2px solid #dcd400;
    text-align: center;
    font-family: "Arial Black";
    font-size: 50px;
    color: #ffffff;
    float: left;
}

▼ #pistas .pistaCorrer{
    width: 575px;
    height: 90px;
    float: left;
    border-bottom: 2px solid #dcd400;
}

▼ #pistas .final{
    width: 98px;
    height: 90px;
    float: left;
    background-image: url(..../images/final.png);
    background-repeat: no-repeat;
    border-bottom: 2px solid #dcd400;
}
```

Figura 24 – Código CSS para formatar as pistas do jogo Run

Por fim, temos o código CSS responsável por formatar os carros e as caixas de vencedor, Figura 25.

```

▼ #carro1,#carro2,#carro3,#carro4{
    position:absolute;
    left:0px;
}

▼ .vencedor {
    font-family: Arial;
    font-size: 30px;
    color: #b53f3f;
    text-align: center;
    text-shadow: 0 0 10px #000;
}

▼ .vencedor img, .vencedor p{
    display: inline-block;
    vertical-align: middle;
}
  
```

Figura 25 – Código CSS para formatar os carros

Agora sim, se você testar, seu arquivo corrida.html deve ter um resultado igual à nossa proposta inicial. Porém, se você clicar nos botões para dar início ao jogo, nada irá funcionar, correto?

Você percebe agora que o HTML necessita do JavaScript para que o jogo funcione, logo todo o núcleo do jogo será programado em JavaScript.

Abra o arquivo script.js para começarmos a desenvolver o código, mas antes lembre-se de verificar se você inseriu a linha `<script src="js/script.js"></script>` no arquivo corrida.html na região do cabeçalho logo após o CSS.

Vamos iniciar nosso código criando uma variável chamada game, que será responsável por armazenar um temporizador que fará os carros se moverem conforme um valor passado em milissegundos. Uma variável é um espaço em memória que reservamos para armazenar valores, esse espaço necessita ter um nome no nosso código.

Para criar uma variável, utilizamos a palavra “var” em JavaScript.

Precisamos, ainda, armazenar na memória os 4 carros do jogo; você poderia pensar em criar 4 variáveis distintas, contudo seria mais viável utilizarmos o conceito de vetor ou comumente chamado de Array em programação. Um vetor é semelhante a uma variável, ou seja, é um espaço em memória para guardar valores, contudo um vetor permite armazenar vários valores, diferentemente de uma variável, que permite um único valor por vez.

Com isso, o início do código deve ser como o exibido na Figura 26.

```
1 var carroCorrida = [];
2 var game;
```

Figura 26 – Declaração de variáveis e vetor em JS

Agora precisamos trabalhar com os botões da tela do jogo. Sempre que o usuário clicar neles, devemos fazer algo, certo? Para realizar alguma ação após clicarmos nos botões, devemos tratar os eventos na página, neste caso precisamos tratar o evento de clicar. Porém, antes de tratar os eventos do usuário, devemos verificar se a página do jogo foi totalmente carregada, garantindo que o JavaScript consiga manipular todos os elementos do HTML.

Iremos utilizar um evento chamado onload para verificar se a página foi carregada totalmente; após o carregamento total da página, esse evento irá executar uma função para realizar uma determinada ação.

A Figura 27 exibe o código que verifica o carregamento total da página e o evento de clicar no botão de iniciar a corrida. Os demais botões não serão tratados em JavaScript, visto que são somente links para as páginas já existentes.

```
4 window.onload = function () {
5     setCarro();
6     var buttonIniciar = document.getElementById("iniciarBotao");
7     buttonIniciar.addEventListener("click", function() {
8         this.className = "ativado";
9         mudarImagem();
10        game = setInterval(moveImage, 100);
11    });
12
13};
```

Figura 27 – Evento para verificar se a página foi carregada e evento de clicar

Note que primeiro precisamos associar a tag do HTML com o id iniciarBotao a uma variável (linha 6) para então adicionar o tratamento de evento clique neste botão (linha 7).

A lógica é simples, quando o usuário clicar no botão iniciar jogo, o jogo deve começar a funcionar e os carros a se mover.

Ao clicar no botão, são executados alguns passos dentro da função do evento. Uma função é um trecho de código especializado em fazer alguma tarefa, por exemplo na linha 9 temos a função mudarImagem(). A tarefa dessa função é basicamente alterar os arquivos de imagens dos carros, que inicialmente são do tipo “png”, para os arquivos do tipo “gif”, que permitem uma animação simples.

A função mudarImagem() é exibida na Figura 28, insira essa função logo após a linha 13 do código anterior.

```

15 ▼ function mudarImagem() {
16 ▼   for (i = 0; i < 4; i++) {
17     document.getElementById("carro" + (i + 1)).src = 'images/' + carroCorrida[i].imagem + '.gif';
18     document.getElementById("carro" + (i + 1)).style.left = "0px";
19   }
20 }
  
```

Figura 28 – Função mudarImagem()

Como nós temos 4 carros, devemos alterar a imagem para cada um deles, logo, neste caso, seriam somente 4 linhas de código. Agora imagine que temos 20 carros, seriam várias linhas para fazer a mesma coisa. Assim, quando precisamos repetir uma instrução no código várias vezes, em vez de copiar e colar o código, utilizamos uma instrução de repetição. Na função mudarImagem() temos a instrução for(início; condição; incremento).

A instrução dentro da função irá repetir de 0 a 3 as duas instruções apresentadas nas linhas 17 e 18 para cada carro do jogo.

Analise nosso código do evento onload novamente; percebeu que temos uma função na linha 5? Essa função deve ser capaz de criar na memória os carros do nosso jogo dentro do vetor carroCorrida criado anteriormente. Cada carro é composto por nome (aqui seria o id de cada carro inserido no HTML), imagem e velocidade.

Vamos então criar a função setCarro() e a função que define um carro que vamos chamar de carro, Figura 29. Insira essas funções logo após a mudarImagem().

```

22 ▼ function carro(nome, imagem, velocidade) {
23   this.nome = nome;
24   this.imagem = imagem;
25   this.velocidade = velocidade;
26 }
27
28 ▼ function setCarro() {
29 ▼   for (i = 0; i < 4; i++) {
30     carroCorrida[i] = new carro("carro" + (i+1), "carro" + (i+1), randomize(10, 20));
31   }
32 }
  
```

Figura 29 – Função setCarro e carro

A função carro recebe parâmetros que são nome, imagem e velocidade. Os valores que são passados para esses parâmetros são atribuídos às propriedades do carro que será criado (linhas 23, 24 e 25).

Como temos que criar 4 carros, devemos utilizar uma repetição na função setCarro(). Você se pergunta: por que começar no zero? A resposta é simples: como estamos usando um vetor, devemos acessar cada posição dele, cada posição é representada por um índice numérico que se inicia no valor 0. Para cada carro das posições 0, 1, 2 e 3 iremos criar um novo carro com a instrução “new carro” e passar alguns valores para eles.

Os valores passados para a função carro nesta repetição são: id, representado por “carro1”, “carro2”, “carro3” e “carro4”; o nome da imagem que segue essa mesma nomenclatura e a velocidade, que deve ser um valor entre 10 e 20.

Veja que para passarmos a velocidade, estamos utilizando outra função chamada randomize(min,max), sua tarefa é sortear um valor aleatório entre dois valores, essa Figura é exibida na Figura 30. Insira o código logo após a função setCarro().

```
34 ▼ function randomize(min, max) {
35     return min + Math.round(Math.random() * (max - min));
36 }
```

Figura 30 – Função randomize(min,max)

Estamos quase acabando. Na linha 10 do evento onload, existe a instrução game = setInterval(moveImage,100). A função setInterval tem como tarefa chamar uma função nossa de tempos em tempos conforme o valor em milissegundos que informamos. Na instrução passada será chamada a função moveImage a cada 100 milissegundos.

A função moveImage() é exibida na Figura 31, insira logo após a função randomize.

```
38 ▼ function moveImage() {
39 ▼     for (i = 0; i < 4; i++) {
40         var carro = document.getElementById("carro" + (i + 1));
41         carro.style.left = parseInt(carro.style.left.substr(0, carro.style.left.indexOf("px")))
42             + carroCorrida[i].velocidade + "px";
43 ▼         if (parseInt(carro.style.left.substr(0, carro.style.left.indexOf("px"))) >= 660) {
44             fim(i);
45         }
46     }
47 }
```

Figura 31 – Função moveImage()

A tarefa da função moveImage() é fazer com que os carros se desloquem da esquerda para a direita conforme sua velocidade previamente definida. Aqui novamente estamos usando uma repetição, visto que temos um vetor de carros.

Para cada carro na repetição é armazenado seu nome (id) na linha 40. Em seguida, utilizamos CSS em JavaScript para configurar a propriedade left do elemento (linha 41). A linha 41 possui a função parseInt, que converte um texto para um valor inteiro, nesta linha estamos basicamente pegando o valor left do carro (por exemplo 100px), removemos o “px” do valor para então somar com a velocidade do carro a cada rodada.

Sempre que essa função for executada, verificamos na linha 43 uma condição, aqui com a instrução “if”. Se a posição left do carro for maior ou igual a 660, executamos a função fim(x). O valor 660 representa a chegada, conforme nosso layout proposto para o jogo.

O carro (lado esquerdo da imagem) deve ser maior ou igual a esse valor de 660 para que seja o vencedor; quando um dos carros ganhar, a função fim é acionada com o parâmetro que representa a posição do carro no vetor, ou seja, seu índice.

A função fim é exibida na Figura 32. Insira logo após a função moveImagem.

```

50 ▼ function fim(vencedor) {
51     clearInterval(game);
52 ▼   for (i = 0; i < 4; i++) {
53       document.getElementById("carro" + (i + 1)).src = 'images/i_' + carroCorrida[i].imagem+'.png';
54 ▼     if (i == vencedor) {
55       document.getElementById("vencedor" + (i + 1)).innerHTML = '<div class="vencedor">' +
56         ' <p>Vencedor!</p>';
57     }
58   }
59 }
```

Figura 32 – Função fim(x)

A tarefa dessa função é parar a execução do jogo (linha 51), limpando o nosso temporizador criado com o setInterval. Alterar novamente as imagens de todos os carros para o formato “png” e apresentar o troféu ao vencedor.

O troféu é a caixa que criamos no HTML com o id de “vencedorN”, assim, quando terminar o jogo, a tela deve ser como a apresentada na Figura 33.

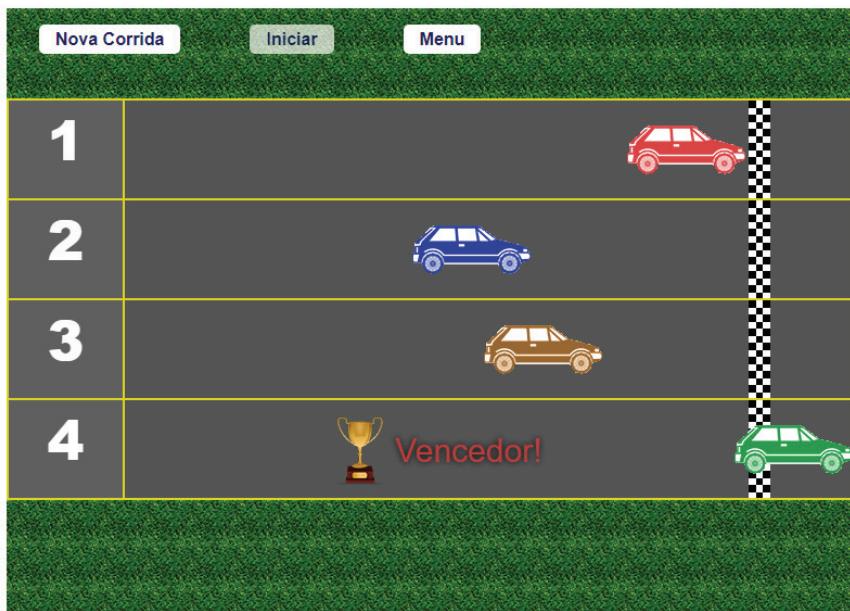


Figura 33 – Fim do jogo

Chegamos ao final da unidade, sabemos que são várias linhas de código, mas temos que praticar. Caso seu jogo apresente algum erro, revise o código. Uma dica é ativar o painel de desenvolver no Chrome quando estiver testando seu jogo. Para ativar o painel, pressione a tecla F12, clique na aba Console e fique atento(a) aos erros apresentados nessa tela, ela indica o erro e a linha do erro, com isso você conseguirá arrumar os possíveis bugs do código.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:



Sites

W3SCHOOLS HTML: the language for building pages

<https://goo.gl/qqckmj>

Solearn – Everyone can code

<https://goo.gl/fx2BVw>

Codecademy

<https://goo.gl/HUi0E1>

Tutoriais MDN Mozilla

<https://goo.gl/9jq6Ma>

Phaser Framework

<https://goo.gl/cd7VnV>

Referências

BUCHARD, E. *The Web Game Developer's Cookbook*: using JavaScript and HTML5 to develop games. Addison-Wesley, 2013.

RAMTAL, Dev; DOBRE, Adrian. *Physics for JavaScript games, animation, and simulations*: with HTML5 canvas. New York: Apress, 2014.

SILVA, Mauricio Samy. **Html 5**: a linguagem de marcação que revolucionou a web. São Paulo: Novatec, 2011.

WORLD WIDE WEB CONSORTIUM W3C. Disponível em: <<https://www.w3.org/html/>>. Acesso em: 09 jun. 2018.



Cruzeiro do Sul
Educacional