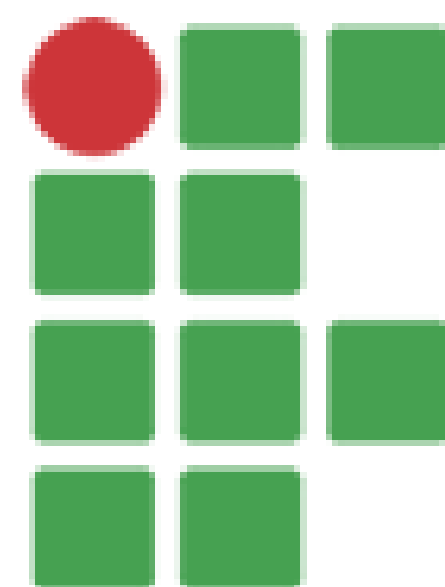




**INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**
Rio de Janeiro

Disciplina

Modelagem de Domínio e
Conceitos de Orientação a Objetos

PROF. FERNANDO TAMBERLINI ALVES

Criando Objetos no Javascript

- Criamos um objeto abrindo e fechando chaves;
- Um objeto pode ser uma variável “let” ou uma constante “const”
- Um objeto parece com um array de chave e valor;
- Podemos iniciar um objeto com vários atributos;
- E os atributos pode ser acessados de duas formas:

```
let carro = {  
  portas: 4  
}
```

```
let carro = {  
  portas: 4,  
  cor: 'verde'  
}  
  
console.log(carro.portas);  
console.log(carro['cor']);
```

Atributos e Tipo de Dados

- Atributos podem ser de qualquer tipo de dados;
- Boolean, number, string, array, objeto...

```
let carro = {  
  portas: 4,  
  cor: "verde",  
  opcionais: [  
    "teto solar", "blindagem", "ar condicionado"  
  ],  
  revisado: true  
}  
  
console.log(carro.portas);  
console.log(carro["cor"]);  
console.log(carro.revisado);  
console.log(carro.opcionais);
```

Atributos e Tipo de Dados

- As propriedades podem ter mais de uma palavra;
- Neste caso precisamos colocá-las entre aspas;
- Não é muito utilizado, opte por camelCase;
- É possível acessar um atributo usando uma variável.

```
let carro = {  
  portas: 4,  
  cor: "verde",  
  "tem blindagem": true  
}  
  
console.log(carro.portas);  
console.log(carro["cor"]);  
console.log(carro["tem blindagem"]);
```

```
const robo = {  
  bracos: 4,  
  pernas: 2,  
  arma: 'metralhadora',  
  armaEspecial: 'foguete'  
}  
  
let a = 'arma';  
  
console.log(robo[a]);
```

Métodos

- Os métodos são as operações objetos;
- Os métodos podem recuperar e/ou alterar os dados dos atributos do objeto;
- Os métodos podem realizar qualquer operação que uma função realiza

```
const robo = {  
  bracos: 4,  
  pernas: 2,  
  arma: 'metralhadora',  
  armaEspecial: 'foguetete',  
  atirar: function() {  
    console.log('pew pew pew');  
  }  
}  
  
robo.atirar();
```

```
let pessoa = {  
  nome: 'Matheus',  
  getNome: function() {  
    console.log("O nome é " + this.nome);  
  },  
  setNome: function(novoNome) {  
    this.nome = novoNome;  
  }  
}  
  
pessoa.getNome();  
pessoa.setNome('Teste');  
pessoa.getNome();
```

Objetos dentro de Objetos

- Como o objeto é também um tipo de dado, podemos ter objetos com objetos dentro;
- Utilizando como um array associativo, por exemplo:
- O objeto não é imutável, ele pode ganhar atributos e métodos ao longo do código;

```
let pessoa = {  
  nome: 'Matheus',  
  características: {  
    olhos: 'verdes',  
    cabelo: 'castanho',  
    brincos: false,  
    olhos: false  
  }  
}  
  
console.log(pessoa.caracteristicas.cabelo);
```

```
let pessoa = {  
  nome: 'Matheus',  
}  
  
pessoa.idade = 29;  
  
pessoa.falar = function() {  
  console.log('Olá');  
}  
  
console.log(pessoa);
```

Trabalhando com Objetos

- Como é possível adicionar, também podemos excluir atributos dos objetos;
- A palavra reservada `this` dentro de um objeto, vai se referir a própria instância;
- Podemos utilizar para resgatar as propriedades em um método;

```
let pessoa = {  
  nome: 'Matheus',  
}  
  
pessoa.idade = 29;  
  
pessoa.falar = function() {  
  console.log('Olá');  
}  
  
delete pessoa.idade;  
delete pessoa.falar;  
  
console.log(pessoa);
```

```
let pessoa = {  
  nome: 'Matheus',  
}  
  
pessoa.idade = 29;  
  
pessoa.falar = function() {  
  console.log('Olá, eu tenho ' + this.idade + ' anos');  
}  
  
pessoa.falar();
```

Trabalhando com objetos

- Como criar um objeto?
- Como acessar um atributo?
- Como executar um método?

```
1  ✓ const nomeObjeto = {  
2      atributo1 : 80,  
3      atributo2 : "blue",  
4  ✓      metodo1 : function() {  
5          console.log("Olá Mundo !");  
6          console.log("Minha cor é " + this.atributo2);  
7      }  
8  }  
9  nomeObjeto.atributo1;  
10 nomeObjeto.metodo1()
```

Trabalhando com Classes

- Como definir uma classe?
- Como instanciar um objeto a partir de uma classe?

```
class NomeClasse{  
    constructor(p1,p2){  
        this.atributo1 = p1;  
        this.atributo2 = p2;  
    }  
    metodo1(){  
        return this.atributo1*this.atributo2/2;  
    }  
}  
  
const nomeObjeto = new NomeClasse(5,8);
```

Exercício Prático

- Modelar e codificar o clássico jogo do celular Nokia
 - Identificar as classes
 - Identificar os atributos
 - Identificar os relacionamentos
 - Elaborar o diagrama de classes
 - Codificar os objetos do jogo no VSCODE
 - Salvar o código no Github

Ambiente de Desenvolvimento

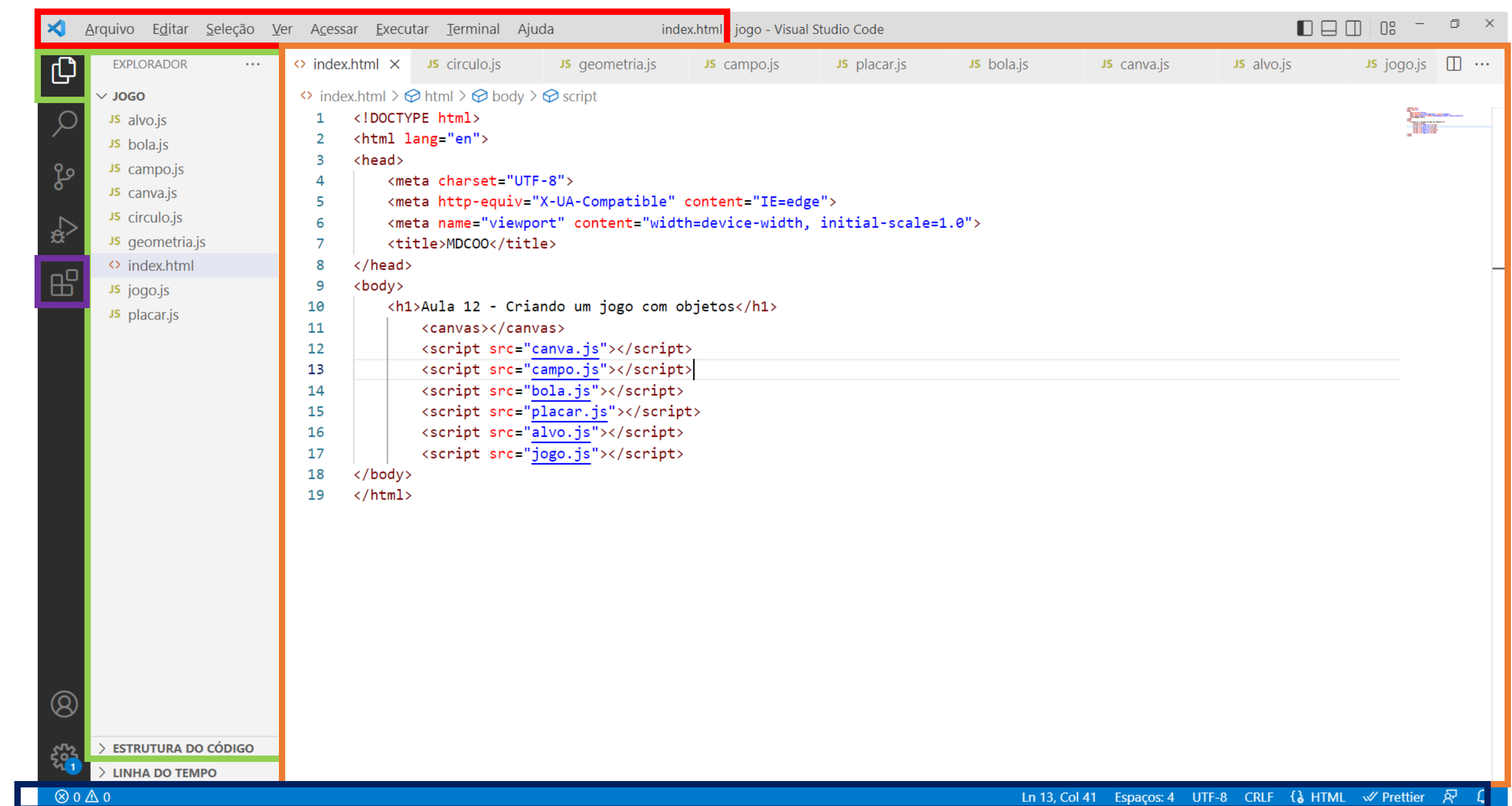
Menu Superior

Explorador

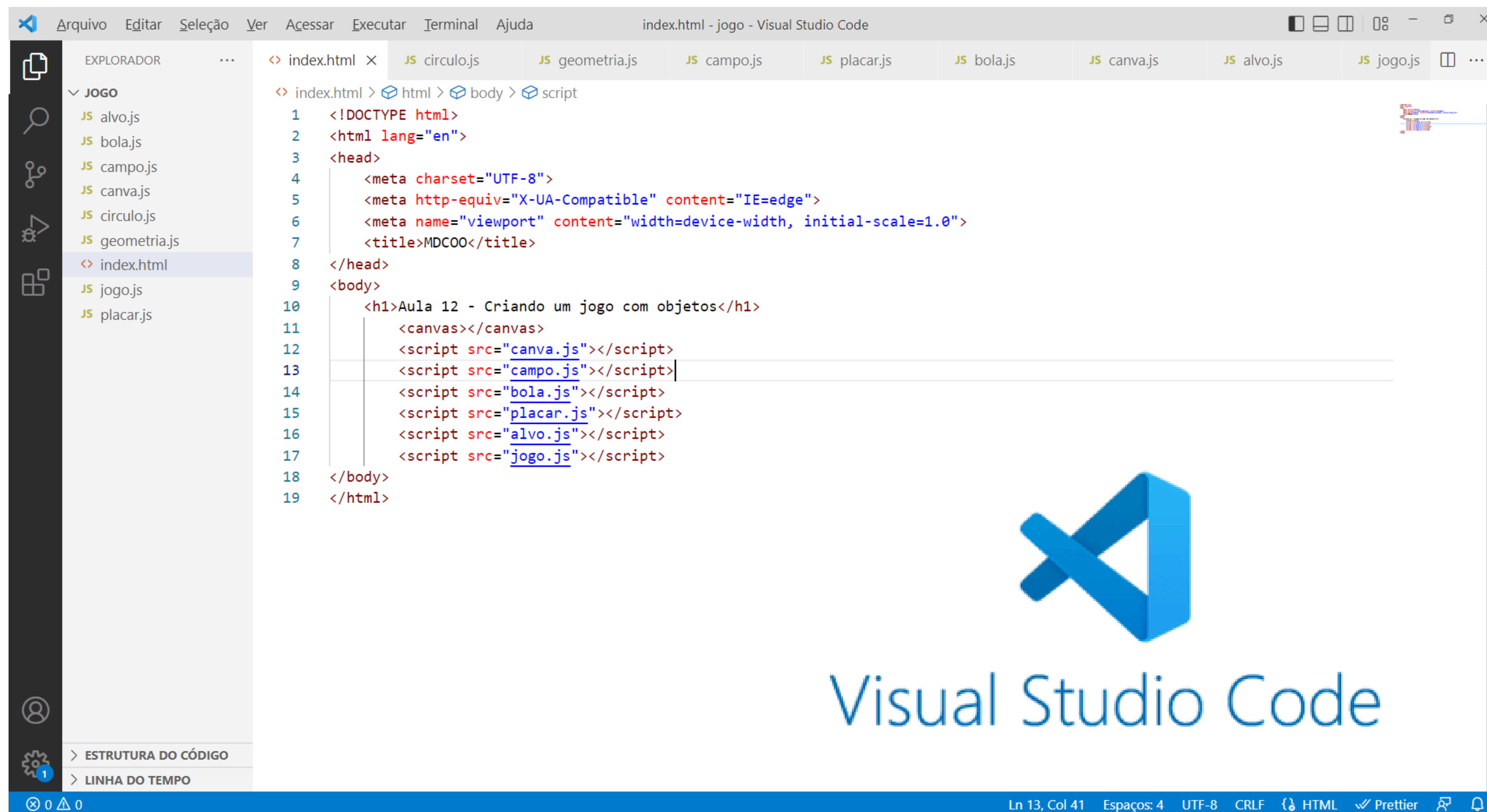
Editor

Extensões

Barra de Status



Ambiente de Desenvolvimento



Visual Studio Code

Exercício Prático

- Modelar clássico jogo do celular Nokia



1º Passo – Criando index.html

- Criar o arquivo index.html de forma incremental.
- Preste atenção nas tags <canvas> e <script>

```
<> index.html >  html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Aula 10 - Jogo da Cobra</title>
7  </head>
8  <body>
9      <canvas></canvas>
10     <script src="canva.js"></script>
11     <script src="cobra.js"></script>
12     <script src="tela.js"></script>
13     <script src="placar.js"></script>
14     <script src="comida.js"></script>
15     <script src="jogo.js"></script>
16 </body>
17 </html>
```

2º Passo – Criando canva.js

- Não se preocupe ainda com o código deste arquivo.

JS canva.js > ...

```
1  const canvasElemento = document.querySelector("canvas");
2  const canvasCtx = canvasElemento.getContext("2d");
3
4  function setup(){
5      canvasElemento.height = canvasCtx.height = screen.height
6      canvasElemento.width = canvasCtx.width = screen.width
7  }
8  setup();
```

O que é o Canvas ?

- **Definição**

O <canvas> é um elemento HTML que pode ser usado para desenhar gráficos em tempo real, como linhas, formas, texto, e imagens. Ele atua como uma área de desenho que pode ser manipulada com JavaScript.

- **Uso comum**

Criar gráficos dinâmicos, animações, jogos, visualizações de dados, e interfaces de usuário personalizadas.

- **Elementos principais**

Elemento HTML <canvas> que define a área onde os gráficos serão renderizados. Fornece métodos e propriedades para desenhar e manipular gráficos dentro do elemento <canvas>

O que é o Canvas ?

- Principais funcionalidades
 - Desenho de formas: Retângulos, círculos, linhas, polígonos, etc.
 - Manipulação de imagens: Carregar e manipular imagens, aplicar filtros, etc.
 - Animações: Atualizar gráficos em uma taxa de quadros para criar movimento.
 - Texto: Renderizar texto com várias fontes e estilos.
- Aplicações do Canvas
 - Jogos: Muitos jogos simples para navegador são desenvolvidos usando o canvas.
 - Visualizações de dados: Gráficos dinâmicos, como gráficos de barras, linhas e dispersão.
 - Ferramentas de desenho: Aplicações que permitem aos usuários desenhar ou anotar imagens.
 - Animações e efeitos visuais: Criação de efeitos visuais complexos, como partículas, sombras, e gradientes.

Exemplos

JS exemplo.js > ...

```
1  const canvasElemento = document.querySelector("canvas");
2  const ctx = canvasElemento.getContext("2d");
3
4  // Desenha um retângulo preenchido
5  ctx.fillStyle = 'blue';
6  ctx.fillRect(50, 50, 200, 100);
7
8  // Desenha um círculo
9  ctx.beginPath();
10 ctx.arc(150, 200, 40, 0, Math.PI * 2, true);
11 ctx.fillStyle = 'red';
12 ctx.fill();
13
14 // Desenha texto
15 ctx.font = '20px Arial';
16 ctx.fillStyle = 'black';
17 ctx.fillText('Olá, Canvas!', 70, 300);
```

Onde aprender mais ?



https://www.w3s.com/graphics/canvas_intro.asp



https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

3º Passo – Criando tela.js

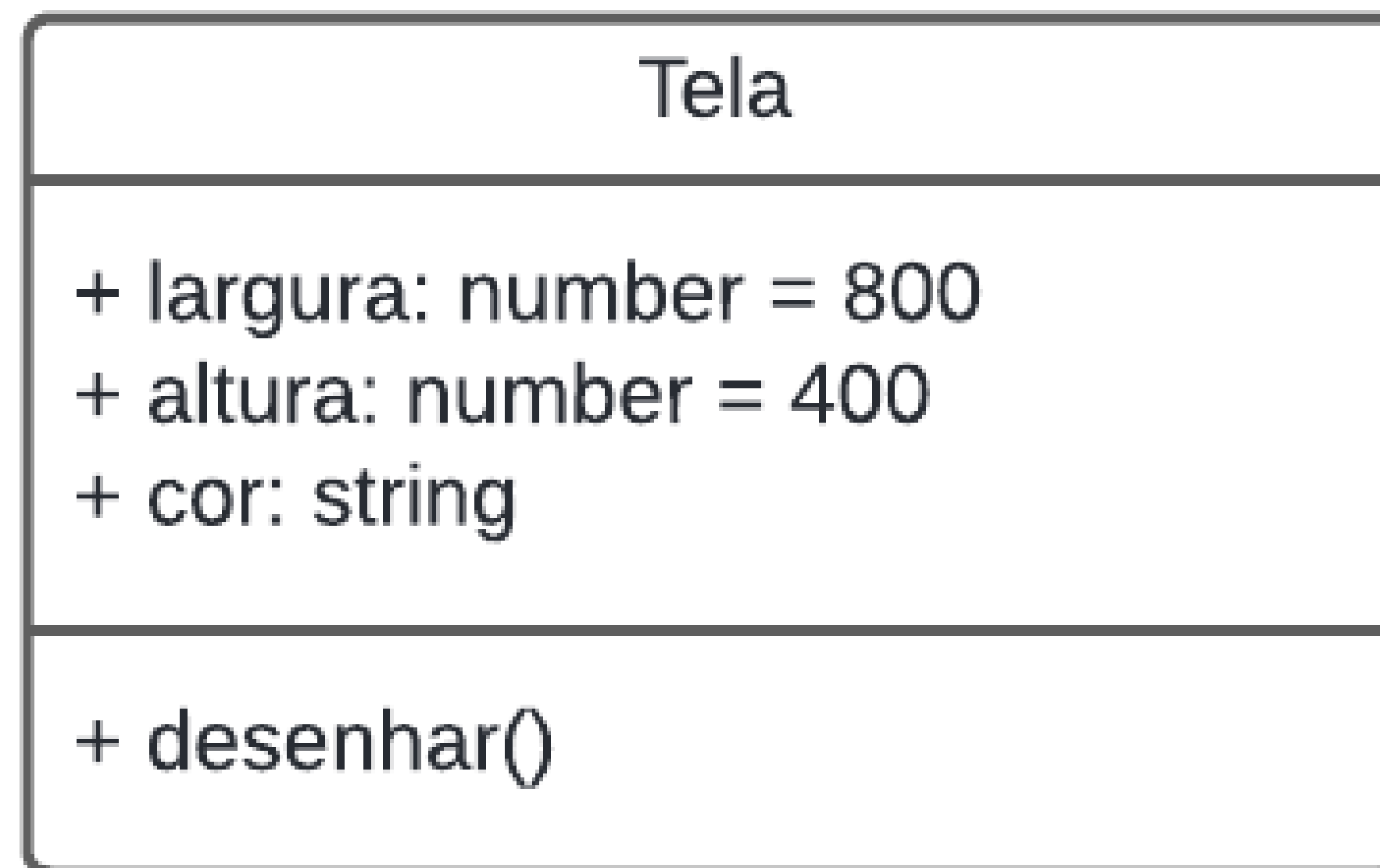
- Aqui já criamos nosso 1º objeto em JS
- Consegue identificar o nome do objeto, seus atributos e seu método?

JS tela.js > ...

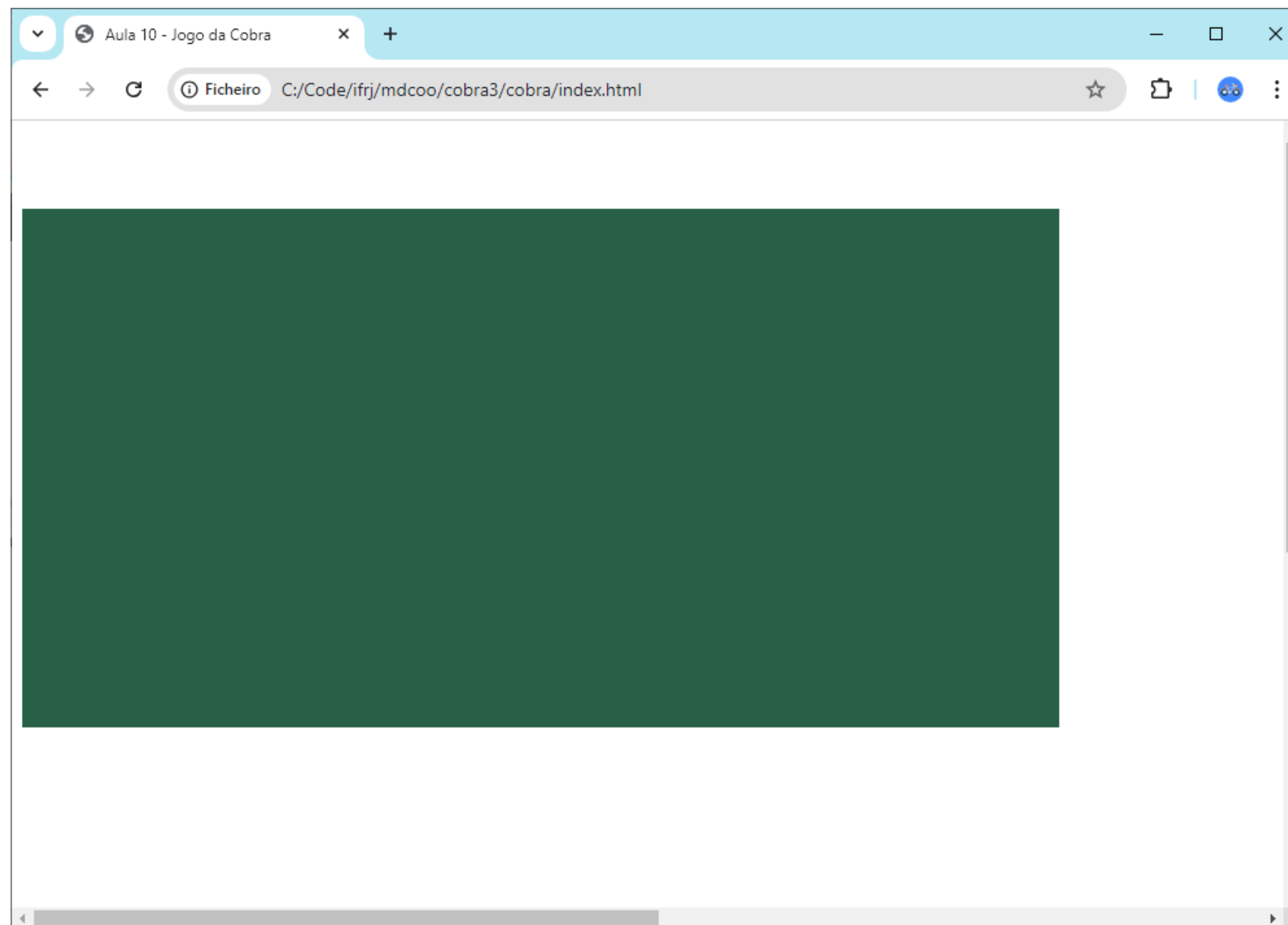
```
1  const tela={
2      largura:800,
3      altura:400,
4      cor:"#286047",
5      desenhar: function(){
6          canvasCtx.fillStyle= this.cor;
7          canvasCtx.fillRect(0,60,this.largura,this.altura);
8      }
9  };
```

Representação em UML

- Aqui está a representação do nosso objeto em UML



Visualizando o resultado

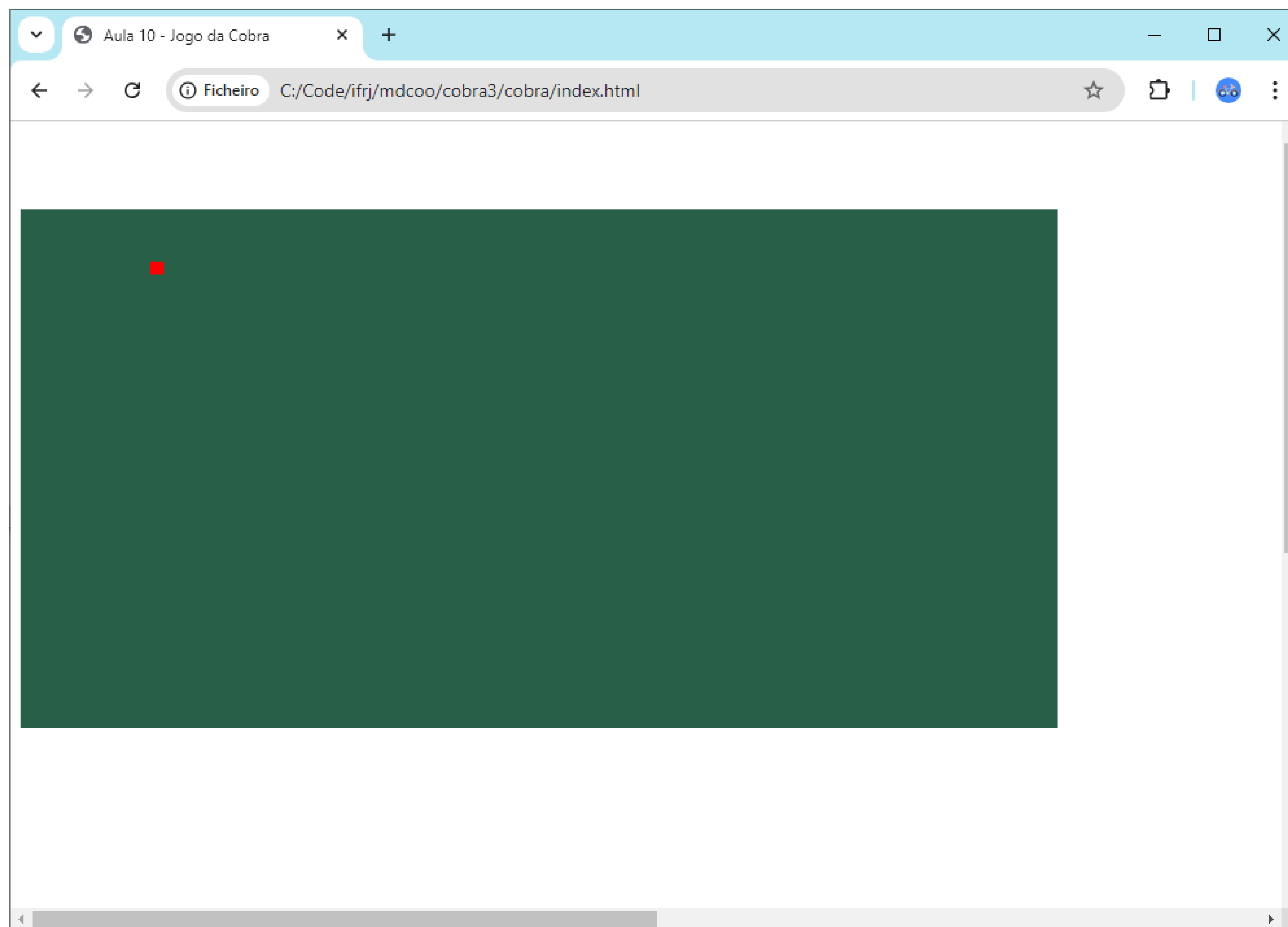


4º Passo – Criando cobra.js

- Aqui criamos a nossa 1ª versão do objeto cobra

```
1  ∨ const cobra = {  
2      x:100,  
3      y:100,  
4      tamanho:10,  
5      cor:"red",  
6  ∨  desenhar(){  
7      |      canvasCtx.fillStyle= this.cor;  
8      |      canvasCtx.fillRect(this.x,this.y,this.tamanho,this.tamanho);  
9      |      }  
10 };  
11 cobra.desenhar();
```

Visualizando o resultado



Exercício 1

Modifique o código do objeto cobra procedendo as seguintes alterações:

- Altere a cor da cobra
- Altere a posição da cobra
- Altere o tamanho da cobra

* Lembrando que você poderá criar novos atributos ou métodos.

5º Passo – Alterando cobra.js

- Criando mais um método no objeto cobra
- Para onde a cobra vai?

```
1  const cobra = {
2      x:100,
3      y:100,
4      tamanho:10,
5      cor:"red",
6      desenhar(){
7          canvasCtx.fillStyle= this.cor;
8          canvasCtx.fillRect(this.x,this.y,this.tamanho,this.tamanho);
9      },
10     mover(){
11         this.x++
12     }
13 };
```

6º Passo – Criando jogo.js

- Criando a função jogo para executar os métodos de tela e cobra para desenhar o jogo na tela?

```
cobra > JS jogo.js > ...  
1   function jogar () {  
2       tela.desenhar()  
3       cobra.desenhar()  
4       cobra.mover()  
5       requestAnimationFrame(jogar)  
6   }  
7   requestAnimationFrame(jogar)
```

Exercício 2

Modifique o código do objeto cobra procedendo as seguintes alterações:

- Mover a cobra para esquerda, para cima e para baixo;
- Limitar o movimento da cobra apenas dentro da tela do jogo;
- Criar movimentos não retilíneos;

* Lembrando que você poderá criar novos atributos ou métodos.

7º Passo – Criando Placar.js

- Criando o nosso objeto Placar

```
cobra > JS placar.js > ...
1  const placar = {
2      qtdPontos:0,
3      recorde:0,
4      largura:800,
5      altura:60,
6      nomeJogador:"Clique na tela para iniciar...",
7      nomeJogo:"IFRJ COBRA",
8      cor:"#FFFFFF",
9      corFundo:"#01341D",
10     desenhar(){
11         canvasCtx.fillStyle= this.corFundo;
12         canvasCtx.fillRect(0,0,placar.largura,placar.altura);
13         canvasCtx.fillStyle= this.cor;
14         canvasCtx.font= "14px Fantasy"
15         canvasCtx.textBaseline = "top"
16         canvasCtx.textAlign = "left"
17         canvasCtx.fillText(this.nomeJogador, 10,10);
18         canvasCtx.fillText(this.recorde + " Recorde", 10,36);
19         canvasCtx.textBaseline = "top"
20         canvasCtx.textAlign = "right"
21         canvasCtx.fillText(cobra.vida + " vida(s)", 780,10);
22         canvasCtx.fillText(this.qtdPontos + " pontos",780,36);
23         canvasCtx.font= "20px Fantasy"
24         canvasCtx.textBaseline = "middle"
25         canvasCtx.textAlign = "center"
26         canvasCtx.fillText(this.nomeJogo, 400,30);
27     }
28 };
```

Canvas – textAlign - textBaseline

- Alinhamento e Base do Texto

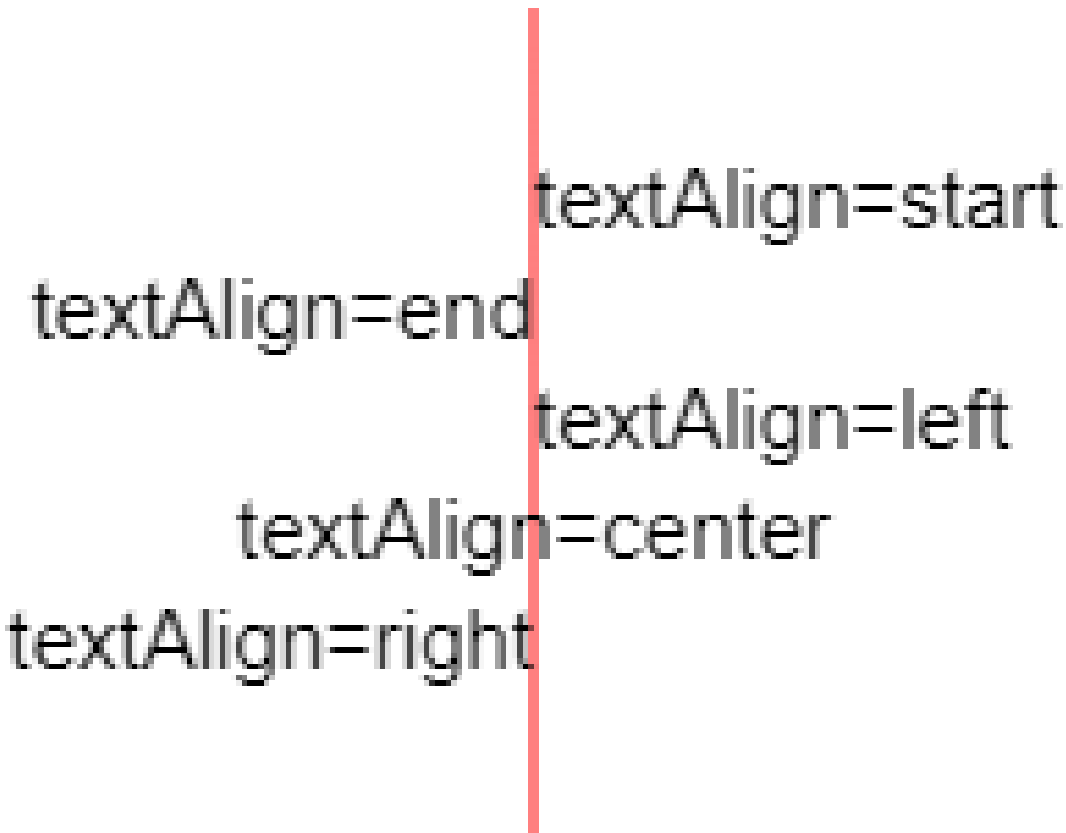


Diagram illustrating text alignment relative to a vertical red line:

- textAlign=start
- textAlign=end
- textAlign=left
- textAlign=center
- textAlign=right



Diagram illustrating text baseline relative to a horizontal red line:

- Top
- Bottom
- Middle
- Alphabetic
- Hanging

Usando Repositório



O Git é um sistema de controle de versão distribuído amplamente utilizado no desenvolvimento de software.



O GitHub é uma plataforma de hospedagem de código-fonte que usa o sistema de controle de versão Git

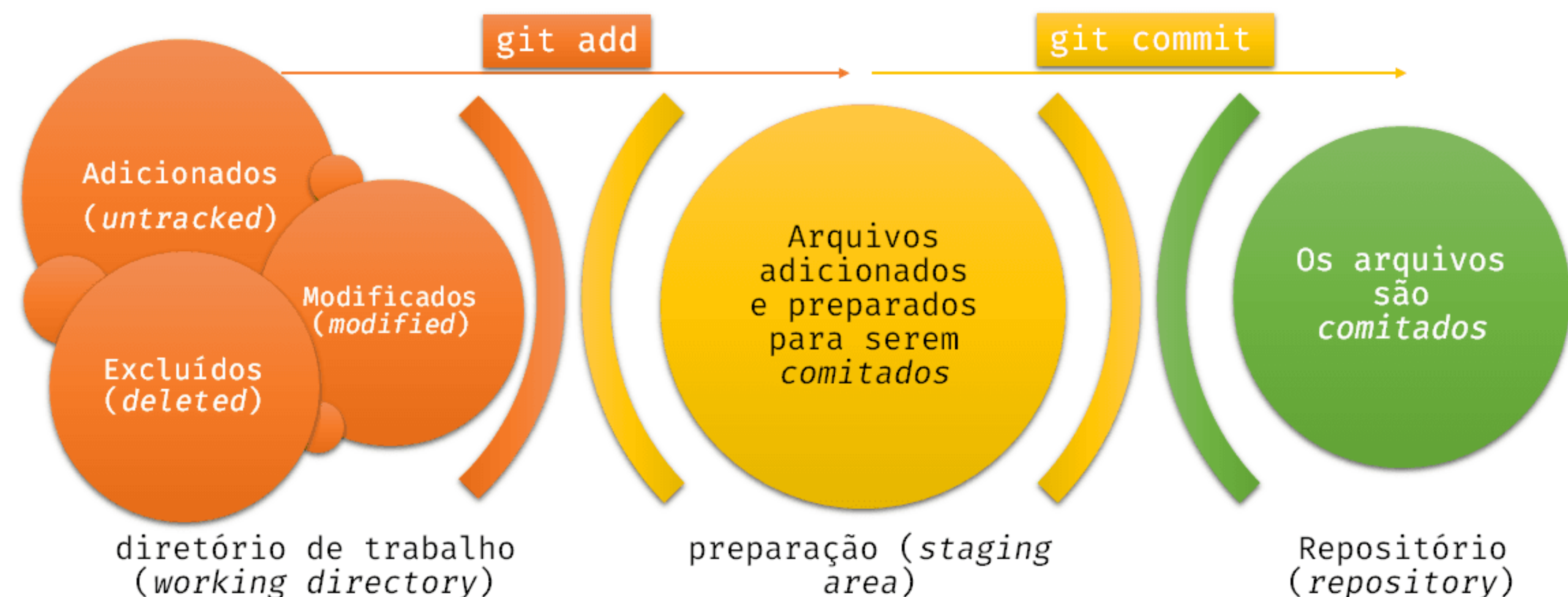
O Git é um sistema de controle de versão distribuído amplamente utilizado no desenvolvimento de software. Ele permite que múltiplos desenvolvedores trabalhem no mesmo projeto simultaneamente, rastreando todas as mudanças feitas no código-fonte.

- **Controle de Versão:** O Git mantém um histórico completo de todas as alterações feitas em um projeto, permitindo que os desenvolvedores voltem a versões anteriores do código se necessário.
- **Distribuído:** Cada desenvolvedor tem uma cópia completa do repositório em seu próprio computador, o que significa que eles podem trabalhar de forma independente e offline. As mudanças podem ser compartilhadas e sincronizadas posteriormente.
- **Branching e Merging:** O Git facilita a criação de ramificações (branches), permitindo que os desenvolvedores trabalhem em funcionalidades ou correções de bugs separadamente. As ramificações podem ser mescladas (merged) de volta ao repositório principal (master) quando estiverem prontas.
- **Colaboração:** Ferramentas de hospedagem de repositórios Git, como GitHub, GitLab e Bitbucket, facilitam a colaboração entre desenvolvedores, fornecendo funcionalidades adicionais como revisões de código, rastreamento de problemas e integração contínua.
- **Eficiência e Velocidade:** O Git foi projetado para ser rápido e eficiente, mesmo com grandes projetos de software.

Principais comandos:

- **git init**: Inicializa um novo repositório Git.
- **git clone <url>**: Clona um repositório remoto para o seu computador local.
- **git add <arquivo>**: Adiciona um arquivo ao próximo commit.
- **git commit -m "mensagem"**: Salva as alterações adicionadas com uma mensagem descritiva.
- **git push**: Envia os commits locais para um repositório remoto.
- **git pull**: Atualiza o repositório local com as mudanças do repositório remoto.

OS TRÊS ESTÁGIOS FUNDAMENTAIS DO GIT



Criando uma conta



O GitHub é uma plataforma de hospedagem de código-fonte que usa o sistema de controle de versão Git. Ele oferece um conjunto de ferramentas para colaboração no desenvolvimento de software, permitindo que desenvolvedores trabalhem juntos em projetos de qualquer lugar do mundo.

- **Repositórios:** No GitHub, os projetos são armazenados em repositórios (repos), que contêm todos os arquivos do projeto, bem como o histórico completo de alterações. Os repositórios podem ser públicos (acessíveis a qualquer pessoa) ou privados (acessíveis apenas a colaboradores específicos).
- **Forks e Pull Requests:**
 - **Fork:** Um fork é uma cópia de um repositório que você pode modificar sem afetar o original. Isso é útil para colaborar em projetos open-source, permitindo que você faça alterações e experimente sem interferir no repositório principal.
 - **Pull Request:** Um pull request é uma solicitação para que suas alterações sejam revisadas e, possivelmente, mescladas no repositório original. É uma maneira central de colaborar e revisar o código.
- **Issues:** O GitHub oferece um sistema de rastreamento de problemas (issues), onde os usuários podem relatar bugs, solicitar novas funcionalidades e discutir aspectos do projeto. Isso ajuda a organizar e priorizar o trabalho no projeto.

Criando uma conta



ftamberlinidev

OverviewRepositories1ProjectsPackagesStars

ftamberlinidev

Joined last week

Edit profile

Popular repositories

cobra

JavaScript

Public

3 contributions in the last year

AugSepOctNovDecJanFebMarAprMayJunJulAug

Mon

Wed

Fri

Learn how we count contributions

Contribution settings

2024

cobra

Public

PinUnwatch1Fork0Star0

master1 Branch0 Tags

Go to file

Add file

Code

About

ftamberlinidev1a Versao do jogo Cobra

f2548b4 · 2 hours ago1 Commit

canva.js1a Versao do jogo Cobra2 hours ago

cobra.js1a Versao do jogo Cobra2 hours ago

index.html1a Versao do jogo Cobra2 hours ago

jogo.js1a Versao do jogo Cobra2 hours ago

placar.js1a Versao do jogo Cobra2 hours ago

tela.js1a Versao do jogo Cobra2 hours ago

README

No description, website, or topics provided.

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Prof. Fernando Tamberlini Alves | Modelagem de Domínio e Conceitos de Orientação a Objetos | Aula 08 - Codificação

36

Usando o git



Depois de ter criado uma conta e um repositório, executar os seguintes comandos no git bash

- Criar o diretório do projeto via shell do SO
- Dentro do diretório do projeto, inicializar um novo repositório Git através do comando ***git init***
- Configurar os usuário e ambiente remoto no Git através dos comandos abaixo:
 - ***git config --local user.name "SEU USUARIO NO GITHUB"***
 - ***git config --local user.email "SEU EMAIL NO GITHUB"***
 - ***git remote add origin "URL COMPLETA DO GITHUB COM REPOSITÓRIO"***
- Incluir todos os arquivos do projeto no área "Preparatória" via comando ***git add <arquivo>***
- Incluir todos os arquivos no repositório local via comando ***git commit -m "mensagem"***
- Exportar para o repositório na nuvem do Github via comando ***git push -u origin master***

Usando o github



ftamberlinidev

1a Versao do jogo Cobra

✓

f2548b4 · 2 hours ago

1 Commit

canva.js	1a Versao do jogo Cobra	2 hours ago
cobra.js	1a Versao do jogo Cobra	2 hours ago
index.html	1a Versao do jogo Cobra	2 hours ago
jogo.js	1a Versao do jogo Cobra	2 hours ago
placar.js	1a Versao do jogo Cobra	
tela.js	1a Versao do jogo Cobra	

README

Pin

Unwatch 1

Fork 0

Star 0

master

1 Branch

0 Tags

Go to file

Add file

<> Code

About

No description, website, or topics provided.

Activity

0 stars

1 watching

0 forks

Releases

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at <https://ftamberlinidev.github.io/cobra/>

Last [deployed](#) by [ftamberlinidev](#) 1 hour ago

Visit site

Build and deployment

Source

Deploy from a branch

Branch

Your GitHub Pages site is currently being built from the master branch. [Learn more about configuring the publishing source for your site.](#)

master

/ (root)

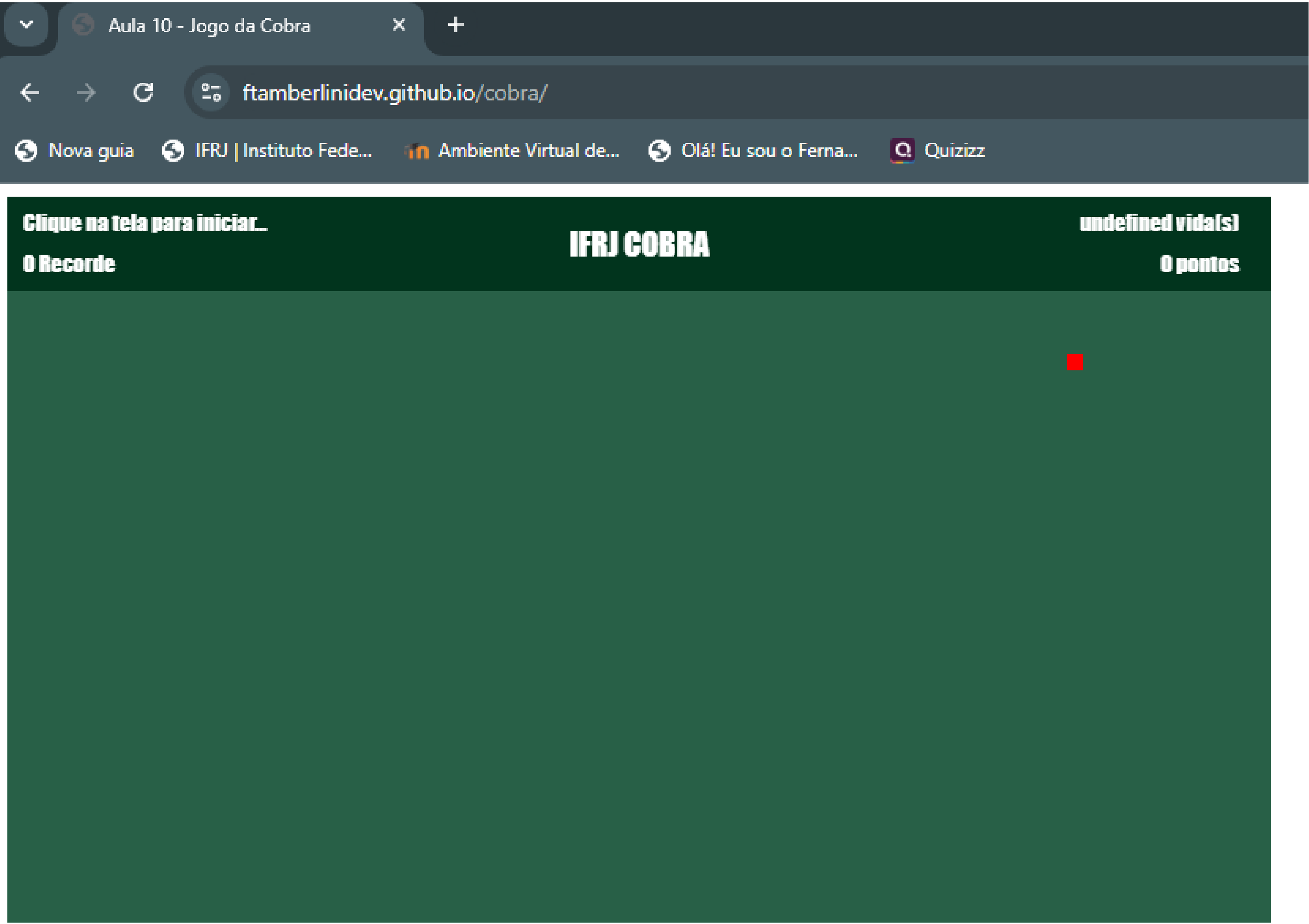
Save

Learn how to [add a Jekyll theme](#) to your site.

Prof. Fernando Tamberlini Alves | Modelagem de Domínio e Conceitos de Orientação a Objetos | Aula 08 - Codificação

38

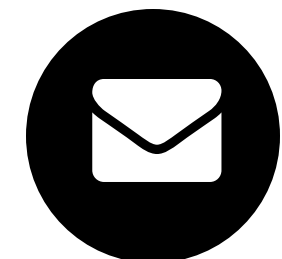
Hospedando no Github



Dúvidas?



Dúvidas?



fernando.alves@ifrj.edu.br



ftamberlini.dev.br