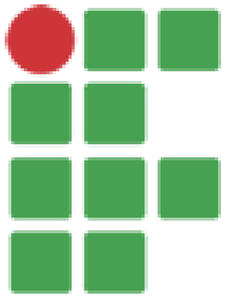


INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Disciplina

Modelagem de Domínio e Conceitos de Orientação a Objetos

PROF. FERNANDO TAMBERLINI ALVES

Paradigmas de Programação

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- ✓ Paradigmas são estilos ou modelos de pensamento que guiam a forma como um programa é estruturado e desenvolvido.

- ✓ Principais Categorias
  - Imperativo vs. Declarativo
  - Estrutural
  - Procedural
  - Funcional
  - Orientado a Objetos
  - Orientado a Serviços

Paradigmas de Programação

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- ✓ Programação Imperativa
  - Foca em como resolver.
  - Usa instruções passo a passo.
  - Ex: C, Python.

- ✓ Programação Declarativa
  - ✓ Foca em o que resolver.
  - ✓ Abstrai o fluxo de controle.
  - ✓ Ex: SQL.

Paradigmas de Programação

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- ✓ Programação Estrutural
  - Usa estruturas de controle (if, for, while).
  - Subdivide o código em blocos
  - Ex: C, Pascal

- ✓ Programação Procedural
  - Baseada em procedimentos ou funções.
  - Controle centralizado com uso extensivo de chamadas de função.
  - Ex: C, Fortran.

Paradigmas de Programação

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- ✓ Programação Funcional
  - Funções puras, imutabilidade, sem efeitos colaterais.
  - Paradigma matemático.
  - Ex: Haskell, Elixir, partes de Python/JavaScript.

- ✓ Programação Orientada a Objetos (POO)
  - Baseada em objetos que encapsulam dados e comportamentos.
  - Conceitos: classe, herança, polimorfismo, encapsulamento.
  - Ex: Java, C#, Python

Paradigmas de Programação

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- ✓ Programação Orientada a Serviços (SOA)
  - Aplicações organizadas como serviços independentes.
  - Comunicação via rede, geralmente com web services (REST, SOAP).
  - Ex: sistemas distribuídos em Java, .NET, Node.js.

- ✓ Programação Orientada a Aspectos (AOP)
  - Separa preocupações transversais (ex: logging, segurança).
  - Ex: AspectJ (Java), Spring AOP

Javascript - Tipo de Dados

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

JavaScript é uma linguagem de programação que é fracamente e dinamicamente tipada, o que significa que o tipo de dados de uma variável é determinado em tempo de execução e sem a necessidade de declarar o seu tipo.

Em JavaScript, os tipos de dados podem ser divididos em duas categorias principais: primitivos e não primitivos (também chamados de objetos).

Primitive Data Types	Reference Data Types
• Boolean • Number • BigInt • String • Undefined • Null • Symbol	• Array • Object • RegEx • Date • ....

Javascript - Tipo de Dados

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Tipos Primitivos:

- **Boolean (Booleano):**
  Representa um valor lógico verdadeiro ou falso.
  Exemplo: let diaEnsolarado = true;

- **Number (Número):**
  Representa tanto números inteiros quanto de ponto flutuante.
  Exemplo: let idade = 25;

- **BigInt:**
  Um tipo especial de número para representar inteiros grandes.
  Exemplo: const valorGrande = 1234567890123456789012345678901234567890n;

- **String (Texto):**
  Sequência de caracteres, normalmente usada para representar texto.
  Exemplo: let nome = "Maria";

Javascript - Tipo de Dados

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Tipos Primitivos:

- **Null (Nulo):**
  Representa a ausência intencional de algum valor.
  Exemplo: let valor = null;

- **Undefined (Indefinido):**
  Representa uma variável que foi declarada, mas ainda não recebeu nenhum valor.
  Exemplo: let sobrenome; // undefined

- **Symbol:**
  Introduzido no ECMAScript 6 (ES6), representa um identificador único.
  Exemplo: const id = Symbol('id');

Javascript – Fracamente Tipada

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

JavaScript é uma linguagem fracamente tipada. Isso significa que você não precisa declarar o tipo de uma variável ao criá-la. O tipo da variável é determinado automaticamente quando o programa é executado, e pode mudar durante a execução do programa.

Em linguagens fortemente tipadas, como Java ou C#, você precisa declarar o tipo de uma variável quando a cria, e essa variável só pode armazenar valores desse tipo específico. Por exemplo, se você declarar uma variável como um número inteiro em Java, ela só pode armazenar inteiros. Em **JavaScript**, por outro lado, você pode ter uma variável que armazena um número em um momento e, em seguida, uma string em outro momento, sem problemas.

```
1   let x = 10;    // x é um número
2   x = "Olá";     // agora x é uma string
3   console.log(typeof(x))
4   x = true;      // agora x é um booleano
5   console.log(typeof(x));
```

Javascript - Declaração de variáveis

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Em JavaScript, temos três palavras-chave para declarar variáveis: **var, let, e const**. Cada uma delas tem um escopo diferente e comportamentos específicos. Vamos explorar as diferenças entre elas:

“var”:

- ✓ var era a única forma de declarar variáveis em versões antigas do JavaScript.
- ✓ Tem um escopo de função ou global. Isso significa que, se uma variável é declarada dentro de uma função usando var, ela é visível em toda a função.
- ✓ Se declarada fora de uma função, torna-se uma variável global.
- ✓ Pode ser redeclarada e atualizada em qualquer lugar do código.
- ✓ Não respeita o escopo de bloco, o que pode levar a bugs difíceis de rastrear.

Javascript - Declaração de variáveis

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

“let:”

- ✓ Introduzido no ECMAScript 6 (ES6), tem um escopo de bloco.
- ✓ Variáveis declaradas com let só são acessíveis dentro do bloco em que são definidas (por exemplo, dentro de loops for, if, while, etc.).
- ✓ Não pode ser redeclarada no mesmo escopo.
- ✓ Pode ser atualizada.

“const:”

- ✓ Também introduzido no ECMAScript 6 (ES6), const cria uma variável cujo valor é fixo (constante).
- ✓ Como let, const tem escopo de bloco.
- ✓ Não pode ser reatribuída. Isso significa que uma vez que uma constante recebe um valor, esse valor não pode ser alterado ou redefinido.
- ✓ No entanto, para objetos e arrays, o valor dos elementos do objeto ou array pode ser modificado. A referência ao objeto ou array é constante, mas não os valores dentro dele.

Javascript - Declaração de variáveis

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Escolhendo entre var, let e const:

- ✓ Use const por padrão para declarar variáveis que não precisam ser reatribuídas.
- ✓ Use let para variáveis que precisam ser reatribuídas.
- ✓ Evite var na maioria dos casos, devido ao seu escopo de função global que pode levar a bugs difíceis de depurar.

Javascript - Declaração de variáveis

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Exemplos:

```
1   function exemplo() {
2       if (true) {
3         var x = 10;
4       }
5       console.log(x); // 10
6     }
7
8     console.log(x); // undefined
9
10  function exemplo() {
11      if (true) {
12        let y = 20;
13        console.log(y); // 20
14      }
15      console.log(y); // Uncaught ReferenceError: y is not defined
16  }
17
18    let y = 30;
19    console.log(y); // 30
20
21  const z = 40;
22  z = 50; // TypeError: Assignment to constant variable.
```

Programação Orientada a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

A Orientação a Objetos (OO) é um paradigma de programação que se baseia na ideia de organizar o código em torno de objetos, que são instâncias de classes. É uma abordagem poderosa e amplamente utilizada no desenvolvimento de software, pois ajuda a criar sistemas **mais organizados, modulares e reutilizáveis**.

A Orientação a Objetos promove uma abordagem mais natural e organizada para modelar o mundo real em código, tornando-o mais legível, manutenível e flexível. É amplamente utilizada em linguagens de programação como Java, C++, Python, C#, entre outras, e é aplicável a uma variedade de domínios de desenvolvimento de software, desde aplicações desktop até sistemas distribuídos e aplicativos web.

Programação Orientada a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Aqui estão os principais conceitos do paradigma de Orientação a Objetos:

Objeto: Um objeto é uma instância concreta de uma classe. Ele representa uma entidade do mundo real e possui atributos (propriedades) que descrevem seu estado e métodos que definem seu comportamento. Por exemplo, um objeto "Carro" pode ter atributos como "cor" e "velocidade" e métodos como "ligar" e "desligar".

Classe: Uma classe é um modelo ou uma definição abstrata que descreve as características comuns a um conjunto de objetos. Ela serve como um plano para a criação de objetos. A classe define os atributos e métodos que seus objetos terão em comum. Por exemplo, uma classe "Carro" define os atributos e métodos que todos os carros terão

Paradigma Orientação a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- ✓Abstração
- ✓Encapsulamento
- ✓Herança
- ✓Polimorfismo

Criando Objetos no Javascript

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- 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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

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

```
let carro = {
  portas: 4,
  cor: "verde",
  opicionais: [
    "teto solar", "blindagem", "ar condicionado"
  ],
  revisado: true
}

console.log(carro.portas);
console.log(carro["cor"]);
console.log(carro.revisado);
console.log(carro.opicionais);
```

Atributos e Tipo de Dados

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- 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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- 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: 'foguete',
    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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- 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',
    caracteristicas: {
        olhos: 'verdes',
        cabelo: 'castanho',
        brincos: false,
        oculos: 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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- 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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

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

```
const nomeObjeto = {
    atributo1 : 80,
    atributo2 : "blue",
    metodo1 : function() {
        console.log("Olá Mundo !");
        console.log("Minha cor é " + this.atributo2);
    }
}
nomeObjeto.atributo1;
nomeObjeto.metodo1()
```

Trabalhando com Classes

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- 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);
```

Ambiente de Desenvolvimento

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

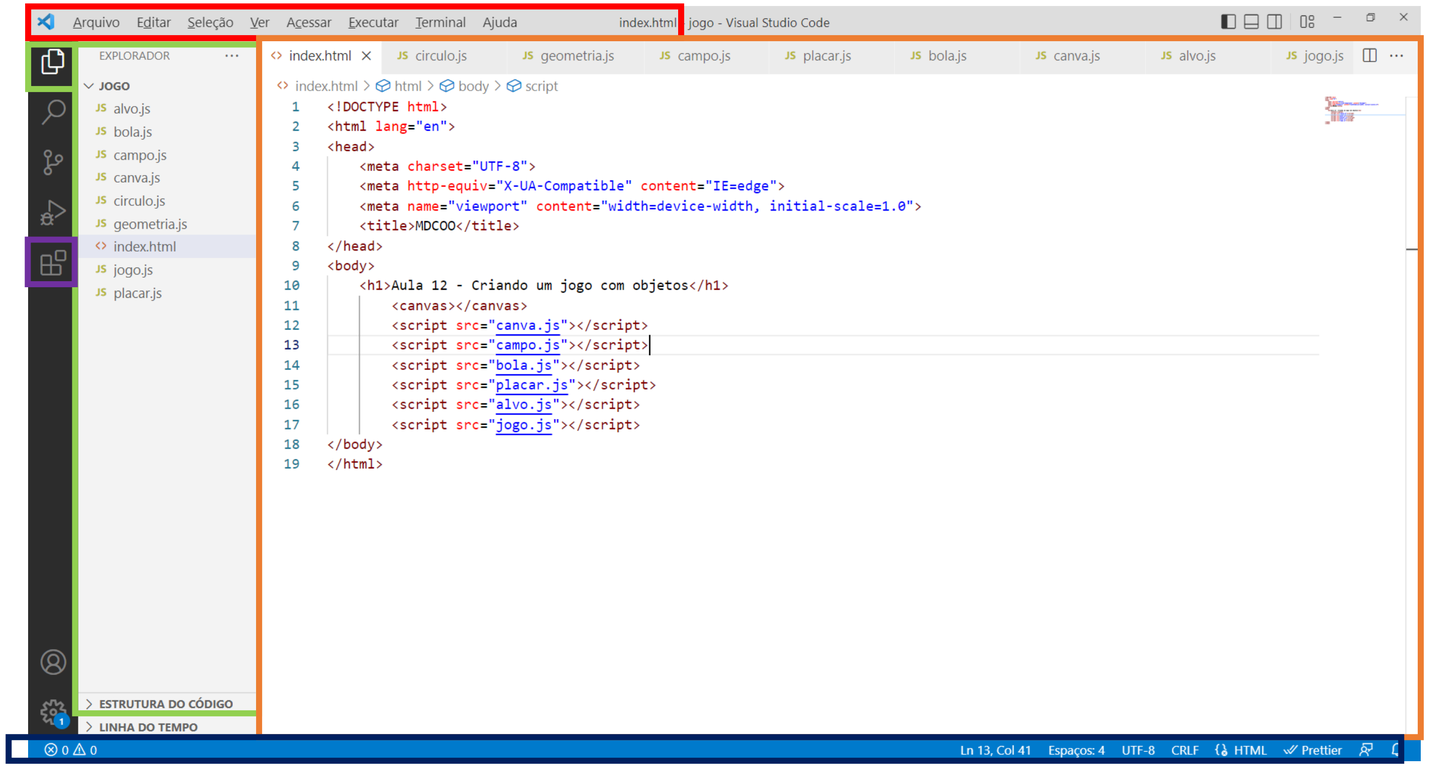
Menu Superior

Explorador

Editor

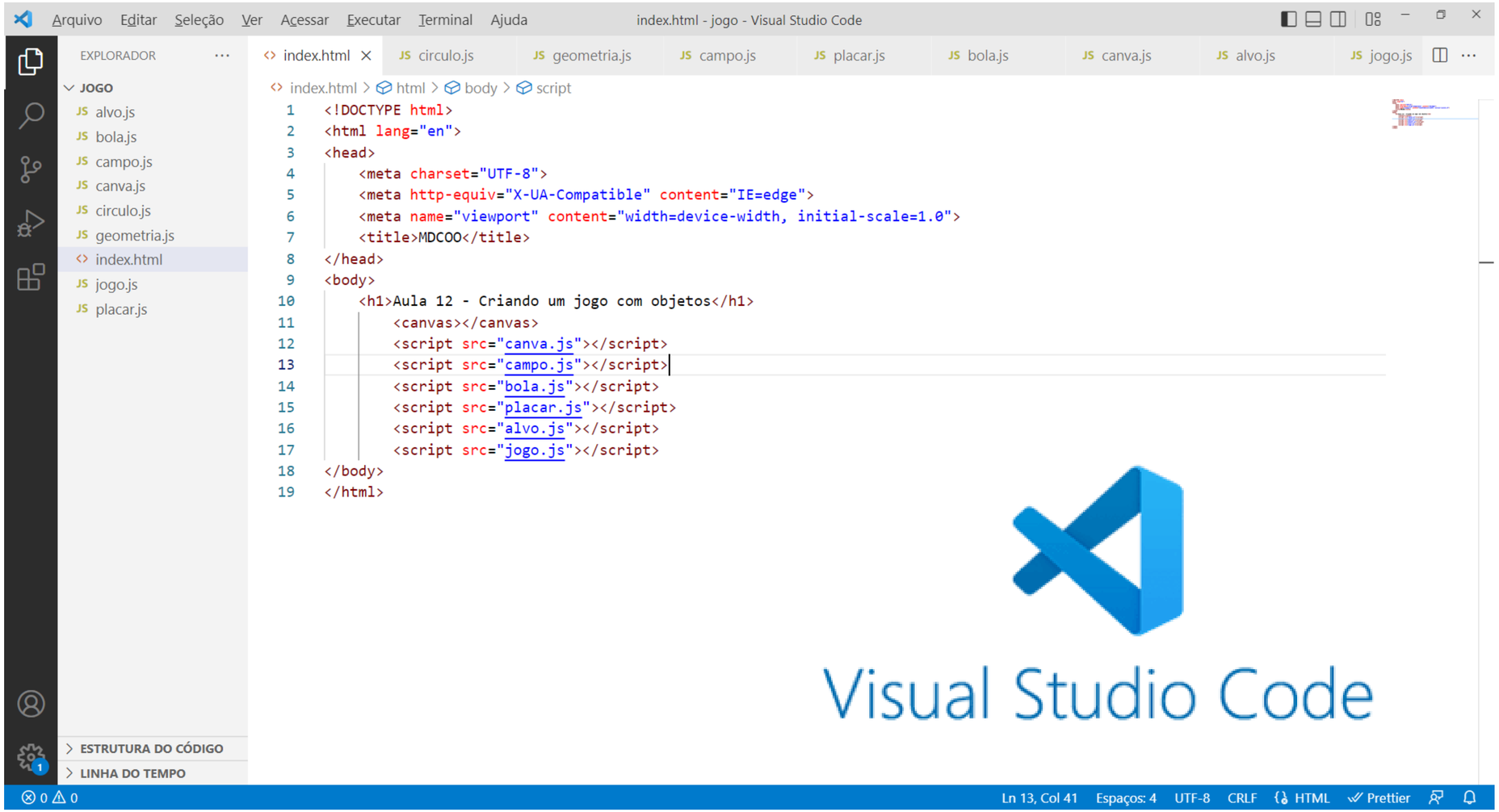
Extensões

Barra de Status



Ambiente de Desenvolvimento

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro



Ambiente de Desenvolvimento

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Teclas de Atalho

Comandos Menu Arquivo

- Abrir Arquivo — Ctrl + O
- Abrir Pasta — Ctrl + K Ctrl + O
- Novo Arquivo Texto — Ctrl + N
- Novo Arquivo — Ctrl + Alt + Windows + N
- Fechar Pasta — Ctrl + K F
- Salvar — Ctrl + S
- Salvar Como — Ctrl + Shift + S

Comandos Menu Editar

- Desfazer — Ctrl + Z
- Refazer — Ctrl + Y
- Comentário Linha — Ctrl + ; ou Ctrl + /
- Comentário Bloco — Shift + Alt + A

Comandos Menu Seleção

- Copiar Linha Cima — Shift + Alt + Seta Para Cima
- Copiar Linha Abaixo — Shift + Alt + Seta Para Baixo
- Mover Linha Cima — Alt + Seta Para Cima
- Mover Linha Abaixo — Alt + Seta Para Baixo
- Cursor Acima — Ctrl + Alt + Seta Para Cima
- Cursor Abaixo — Ctrl + Alt + Seta Para Baixo

- Adicionar Próxima — Ctrl + D
- Adicionar Todas — Ctrl + Shift + L

- Modo Seleção Coluna – Ir no Menu Superior

Exercício Prático

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Modelar clássico jogo do celular Nokia

1º Passo – Criando index.html

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- 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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- 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 ?

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- **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 fefine 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 ?

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- 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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

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 ?

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

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

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

3º Passo – Criando tela.js

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

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

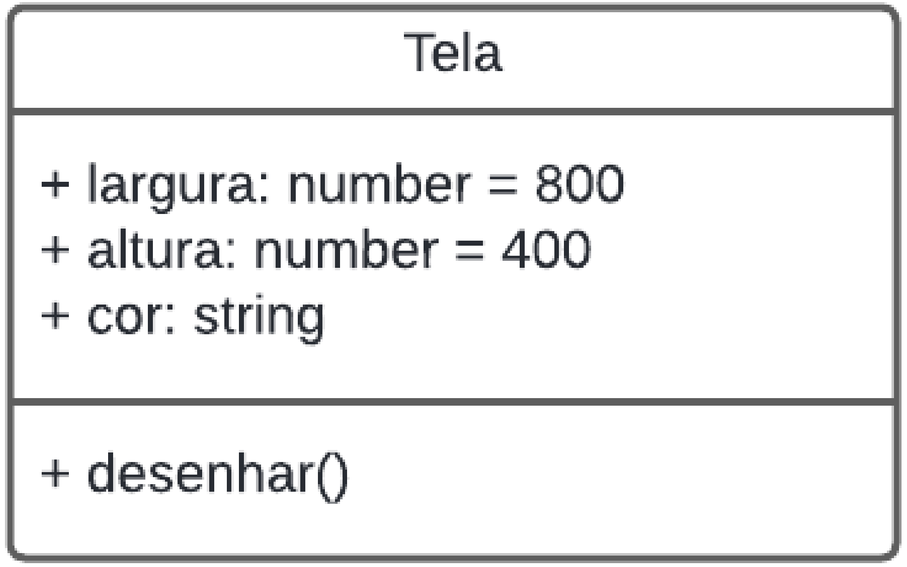
JS tela.js > ...

```
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   };
```

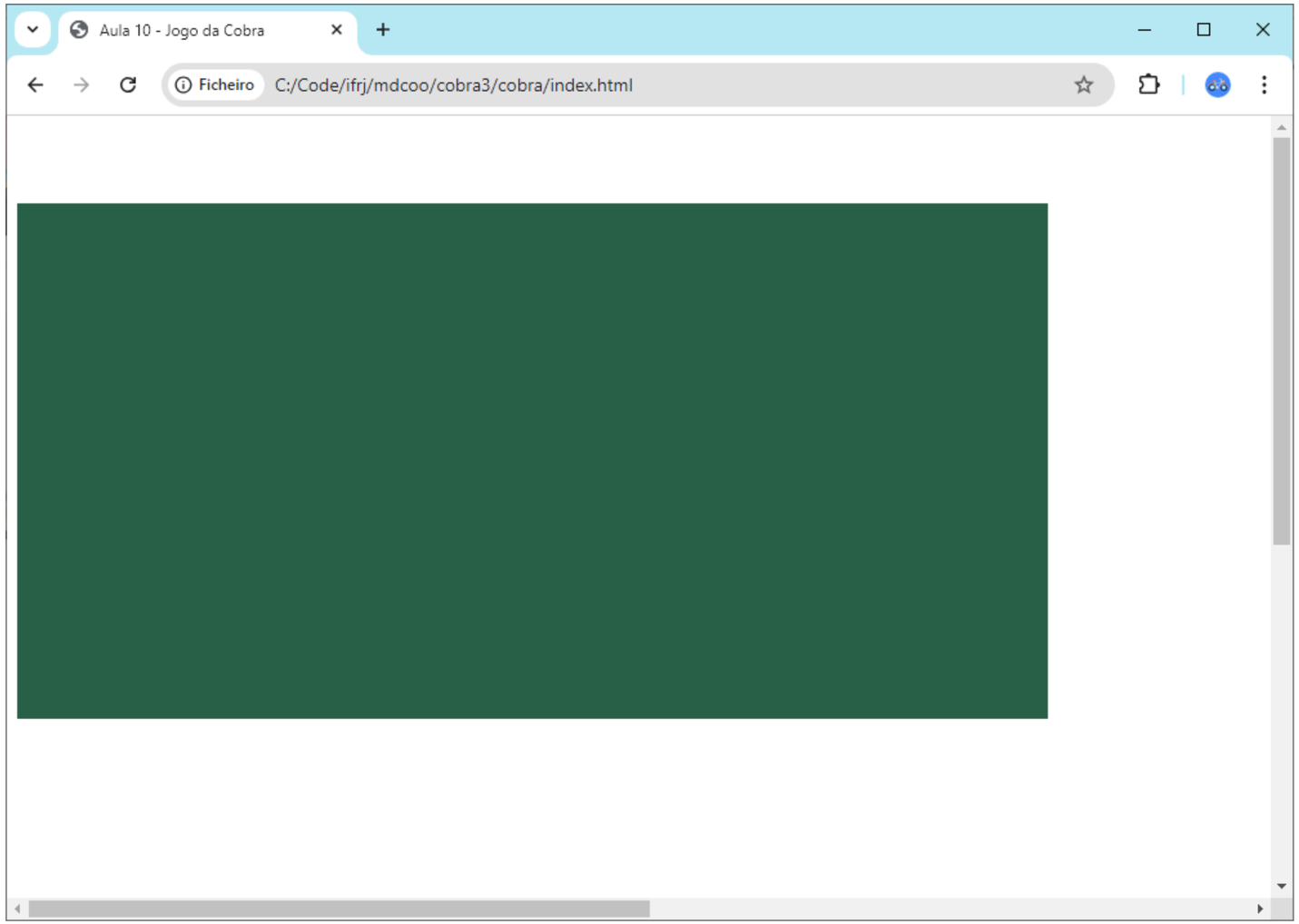
Representação em UML

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Aqui está a representação do nosso objeto em UML



Visualizando o resultado



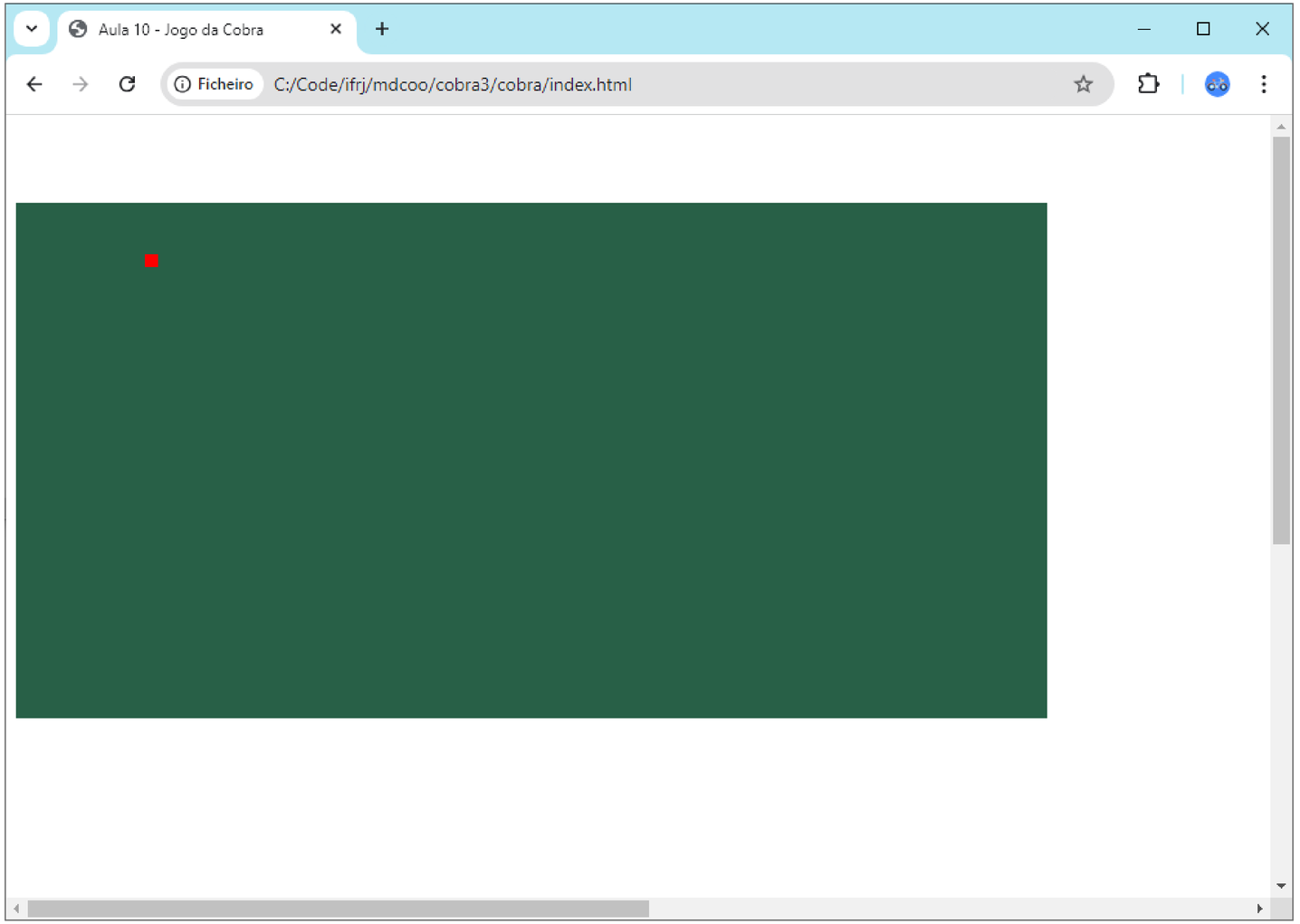
4º Passo – Criando cobra.js

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- 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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- 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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- 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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Criando o nosso objeto Placar

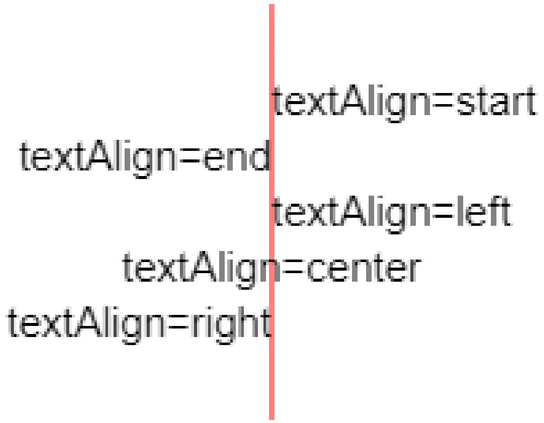
cobra > JS placar.js > ...

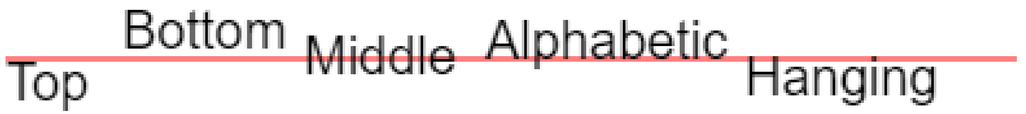
```
1   const placar = {
2       qtdPontos:0,
3       recorde:0,
4       largura:800,
5       altura:60,
6       nomeJogador:"Clique na tela para iniciar...",
7       nomeJogo:"IFRJ COBRA",
8       cor:"#FFFFFF",
9       corFundo:"#01341D",
10      desenhar(){
11          canvasCtx.fillStyle= this.corFundo;
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

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Alinhamento e Base do Texto





Usando Repositório

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

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

Usando Repositório

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

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.

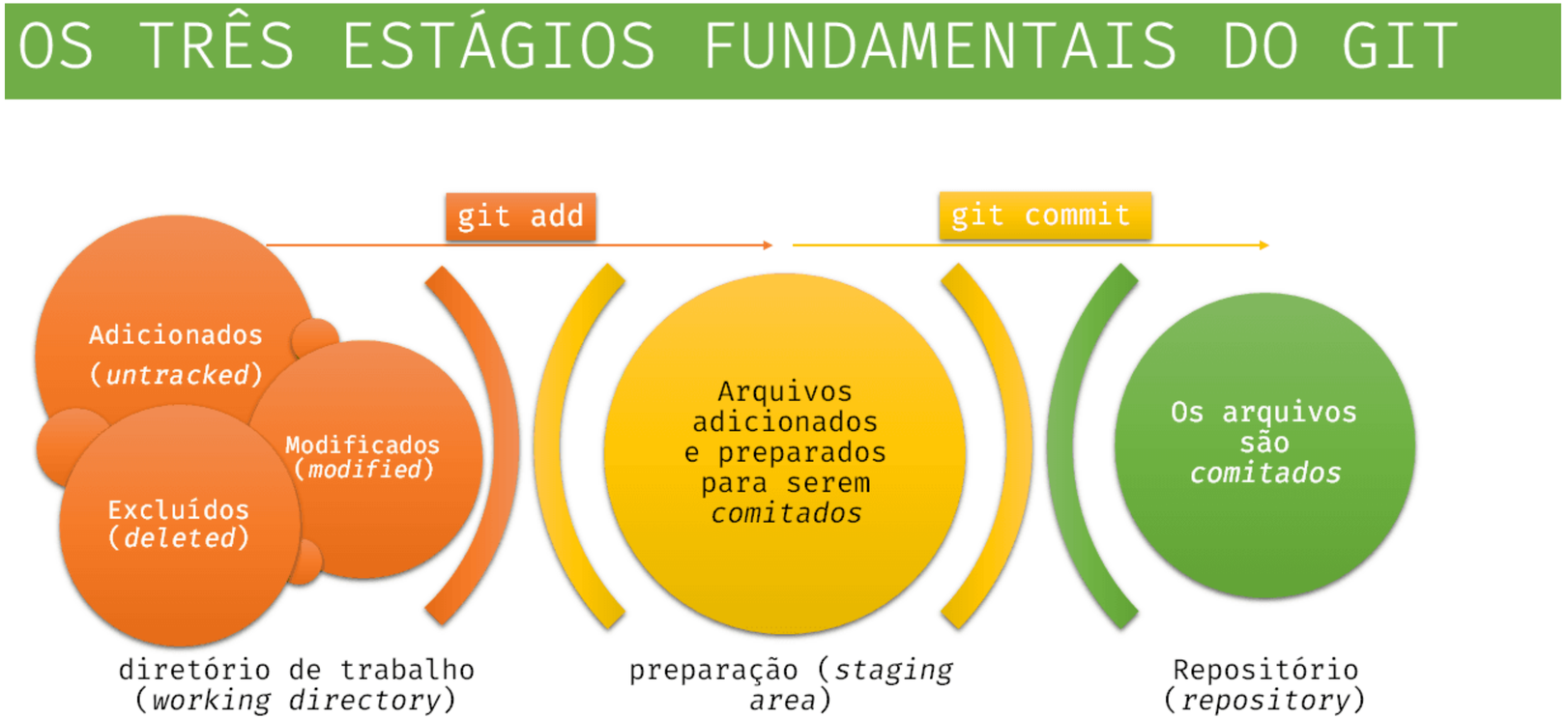
Usando Repositório

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

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.

Usando Repositório



Criando uma conta

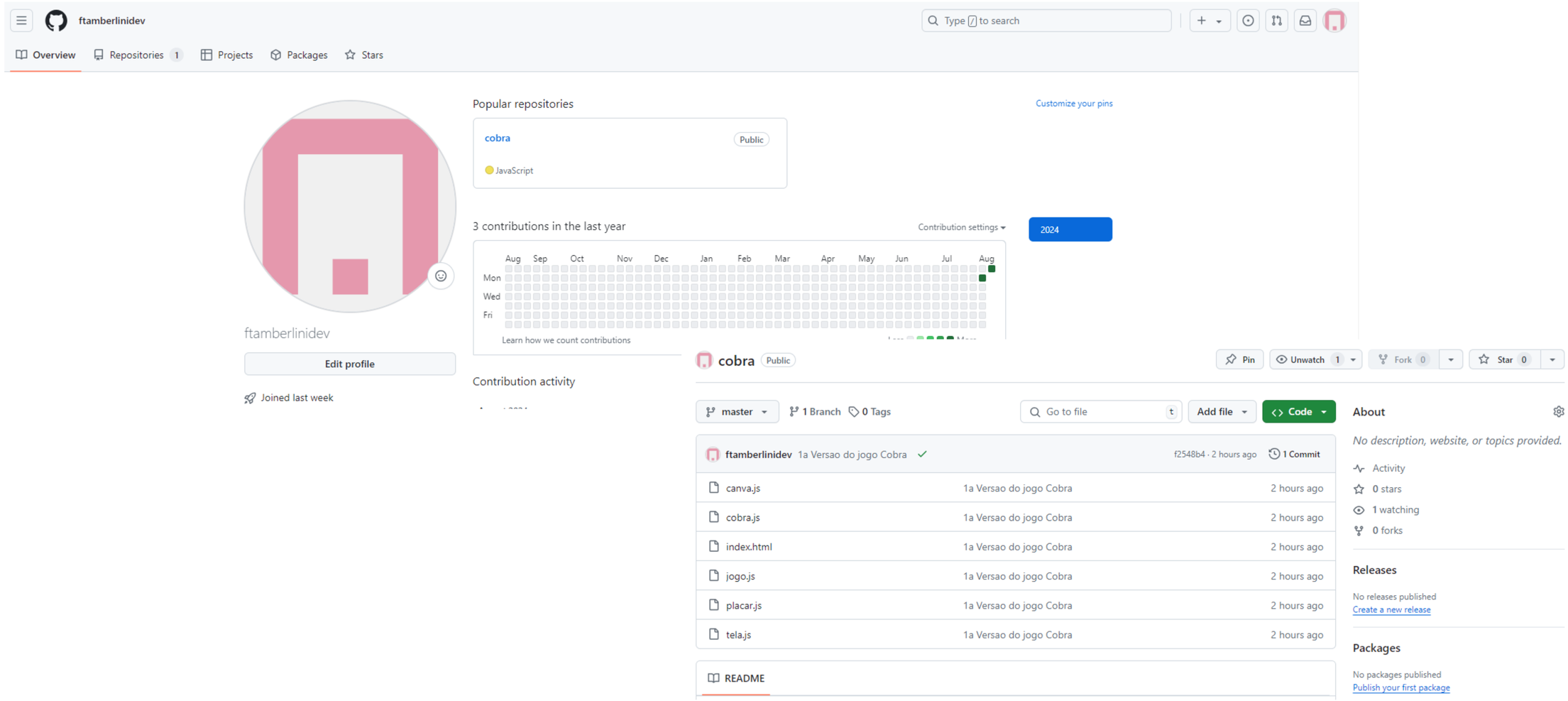
O GitHub é uma plataforma de hospedagem de código-fonte que usa o sistema de controle de versão Git. Ele oferece um conjunto de ferramentas para colaboração no desenvolvimento de software, permitindo que desenvolvedores trabalhem juntos em projetos de qualquer lugar do mundo.

- **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

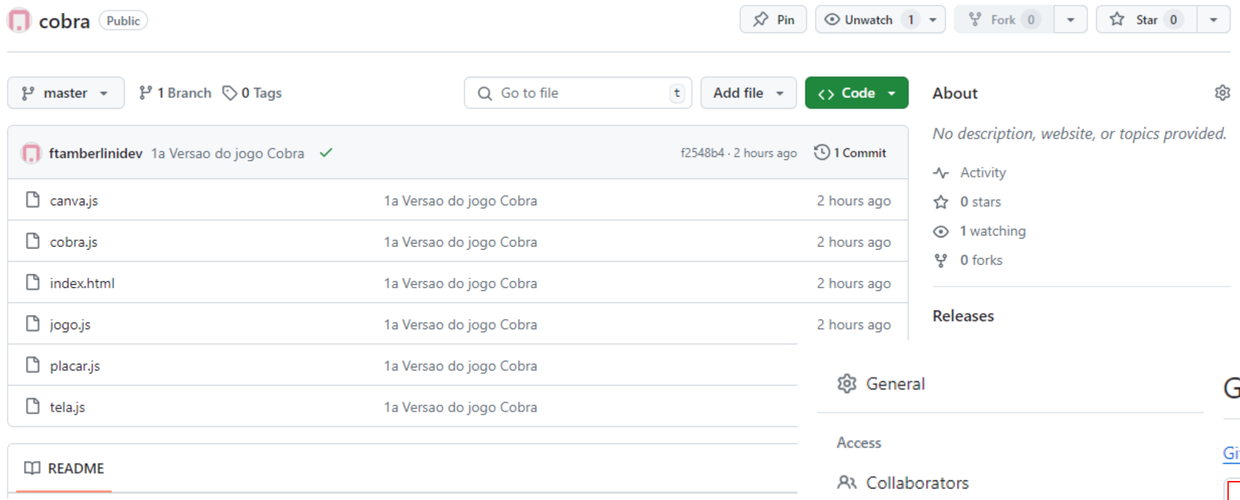


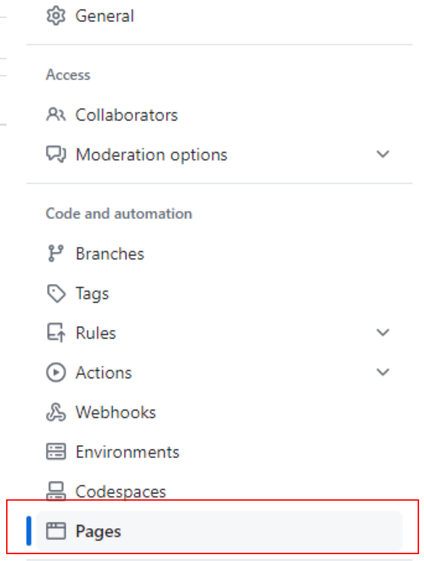
Usando o git

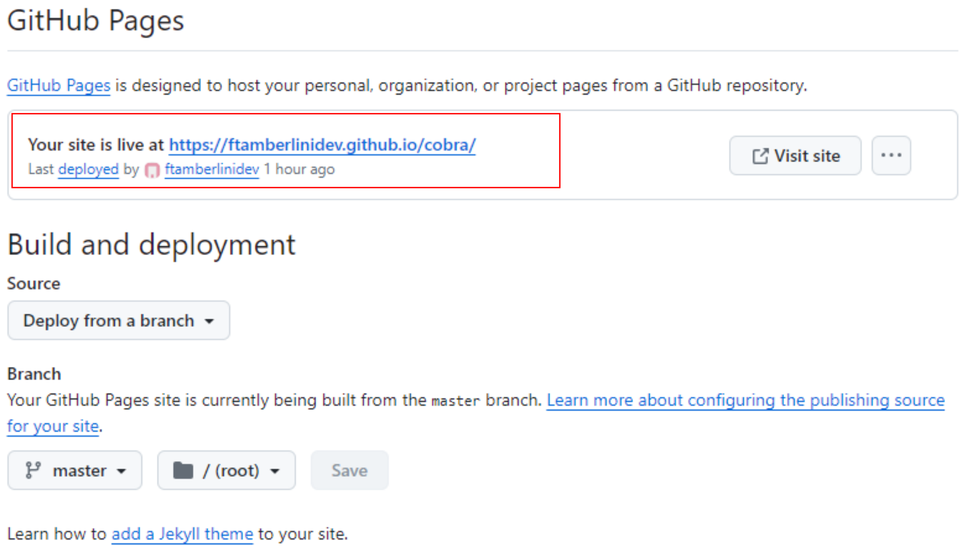
Depois de ter criado uma conta e um repositório, executar os seguintes comandos no git bash

- 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 user.name "SEU USUARIO NO GITHUB"***
  - ***git config user.email "SEU EMAIL NO GITHUB"***
  - ***git config –global init.defaultBranch main***
  - ***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***

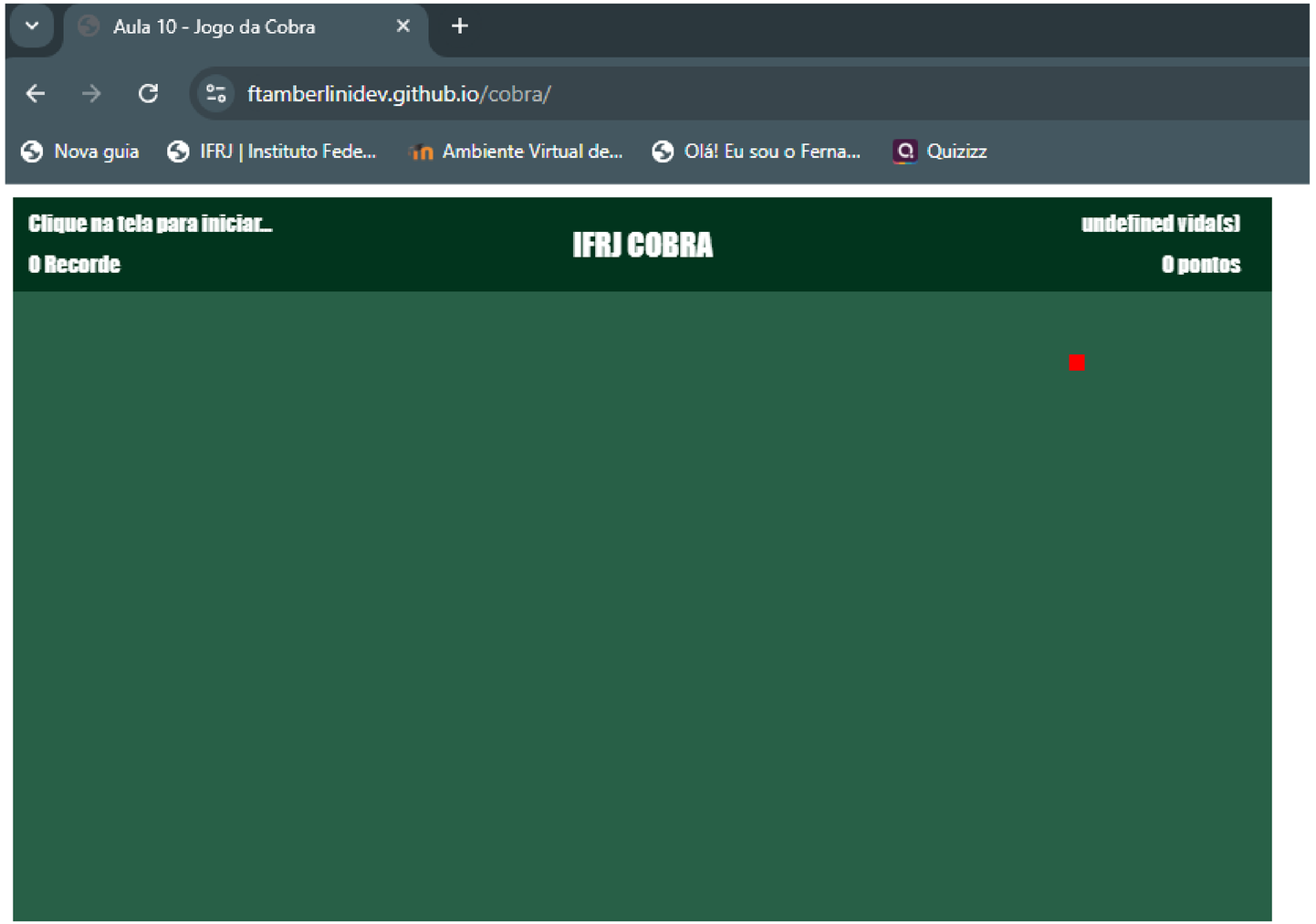
Usando o github







Hospedando no Github



Dúvidas?

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Dúvidas?

fernando.alves@ifrj.edu.br

ftamberlini.dev.br