Aula 1 — Revisão de JavaScript

Frameworks Modernos para Desenvolvimento de Sistemas

Prof. Ivan Luiz Pedroso Pires

Objetivo

Consolidar os conceitos essenciais de JavaScript usados intensivamente em Vue.js (frontend) e Express (backend), com foco em sintaxe moderna (ES6+), manipulação de dados e assincronismo.

Roteiro

1. Sintaxe moderna ES6+ (declarações, arrow, destructuring, spread/rest, template literals)
2. Arrays e objetos no dia a dia (map/filter/reduce/find, imutabilidade prática)
3. `this`, funções e classes (noções rápidas)
4. Assincronismo: `Promise`, `async/await`, `fetch`, `Promise.all`
5. Módulos ES (import/export) — incluindo exemplo com `type=module`
6. Sugestão de exercícios para consolidação do aprendizado

1 Sintaxe Moderna ES6+

Declaração de Variáveis e Arrow Functions

```
// Evite var; prefira let/const
const PI = 3.14;
let contador = 0;

const inc = (n = 0) => n + 1;
console.log(inc(contador));
```

Template Literals, Destructuring, Spread/Rest

```
const user = { id: 1, name: "Ivan", role: "professor" };
const { name, role } = user; // destructuring objeto
const coords = [ -11.85, -55.54 ];
const [ lat, lon ] = coords; // destructuring array

console.log(`Ol, ${name} (${role}) @ ${lat}, ${lon}`);

// clone e merge imutveis
```

```
const prefs = { theme: "dark", lang: "pt-BR" };
const userWithPrefs = { ...user, ...prefs }; // spread

// parmetros rest
const soma = (...nums) => nums.reduce((a,b) => a + b, 0);
console.log(soma(1,2,3,4));
```

Optional Chaining e Nullish Coalescing

```
const aluno = { nome: "Ana", contato: { email: "ana@ex.com" } };

// ?. evita erro se propriedade intermediria for undefined/null
const telefone = aluno.contato?.fone ?? "sem telefone cadastrado";
console.log(telefone);
```

2 Arrays e Objetos no Dia a Dia

map/filter/reduce/find/some/every

```
const alunos = [
  { nome: "Ana", nota: 9, ativo: true },
  { nome: "Pedro", nota: 5, ativo: false },
  { nome: "Maria", nota: 7, ativo: true },
  { nome: "Joo", nota: 8, ativo: true },
];

const aprovados = alunos
  .filter(a => a.nota >= 7 && a.ativo)
  .map(a => a.nome);

const media = alunos.reduce((acc, a) => acc + a.nota, 0) / alunos.length;

const temReprovado = alunos.some(a => a.nota < 7);
const todosAtivos = alunos.every(a => a.ativo);

const maria = alunos.find(a => a.nome === "Maria");

console.log({ aprovados, media, temReprovado, todosAtivos, maria });
```

Atualizações Imutáveis em Objetos/Arrays

```
// atualizar nota de Pedro sem mutar o array original
const atualizados = alunos.map(a =>
  a.nome === "Pedro" ? { ...a, nota: 6.5 } : a
);

// inserir novo aluno (cpia)
const comNovo = [ ...alunos, { nome: "Carla", nota: 10, ativo: true } ];

// remover por condio (sem splice mutvel)
const semInativos = alunos.filter(a => a.ativo);
```

3 `this`, Funções e Classes (noções rápidas)

```
const perfil = {
  name: "Ivan",
  greet() { console.log(`Ol, ${this.name}`); }, // this -> objeto
};
perfil.greet();

class Timer {
  constructor() { this.t = 0; }
  tick() { this.t += 1; }
}
const timer = new Timer();
timer.tick();
console.log(timer.t);
```

4 Assincronismo: Promise, async/await, fetch

Promise e async/await básicos

```
const delay = (ms) => new Promise(resolve => setTimeout(resolve, ms));

async function exemplo() {
  console.log("Incio");
  await delay(500);
  console.log("Meio");
  await delay(500);
  console.log("Fim");
}
exemplo();
```

fetch GET e POST + Tratamento de Erros

```
async function getPost(id) {
  try {
    const res = await fetch(`https://jsonplaceholder.typicode.com/posts/${id}`);
    if (!res.ok) throw new Error(`HTTP ${res.status}`);
    return await res.json();
  } catch (err) {
    console.error("Falha ao buscar post:", err.message);
    return null;
  }
}

async function createPost(data) {
  try {
    const res = await fetch("https://jsonplaceholder.typicode.com/posts", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(data)
    });
    if (!res.ok) throw new Error(`HTTP ${res.status}`);
    return await res.json();
```

```
  } catch (err) {
    console.error("Falha ao criar post:", err.message);
    return null;
  }
}
```

Concorrência com Promise.all

```
async function carregarEmParalelo() {
  const [p1, p2, p3] = await Promise.all([
    getPost(1), getPost(2), getPost(3)
  ]);
  console.log(p1?.title, p2?.title, p3?.title);
}
carregarEmParalelo();
```

5 Módulos ES (import/export)

Export/Import — Node (ESM) e Navegador com `type=module`

Módulo `math.js`

```
// exportaes nomeadas e default
export const soma = (a, b) => a + b;
export const media = (...nums) => nums.reduce((a,b)=>a+b,0) / nums.length;
const PI = 3.14159;
export default PI;
```

Node: `index.mjs` (ou `type: "module"` no package.json)

```
import PI, { soma, media } from "./math.js";

console.log(soma(2, 3)); // 5
console.log(media(2, 4, 6)); // 4
console.log(PI);
```

Navegador: `index.html` usando `type=module`

```
<!doctype html>
<html lang="pt-BR">
  <head><meta charset="utf-8"><title>ESM Demo</title></head>
  <body>
    <h1>Exemplo ES Modules</h1>
    <script type="module">
      import PI, { soma, media } from "./math.js";
      console.log(soma(10, 5));
      console.log(media(5, 5, 10));
      console.log(PI);
    </script>
  </body>
</html>
```

Exemplo Express com ESM (Node 18+)

```
// package.json -> { "type": "module" }
import express from "express";

const app = express();
app.get("/", (req, res) => res.json({ ok: true }));
app.listen(3000, () => console.log("Servidor em http://localhost:3000"));
```

Desafio Rápido

1. Dado um array de serviços (`name`, `price`, `active`), filtrar ativos com preço $\geq$ X e calcular a média de preços.
2. Criar um módulo `utils.js` com funções de formatação (ex.: moeda) e importar em um arquivo `main.mjs`.
3. Fazer `fetch` GET e POST (mock API) usando `async/await` e `try/catch`, logando erros de forma amigável.

Exercícios com APIs Públicas (fetch + async/await)

A seguir, sugestões de exercícios práticos para consolidar `fetch`, `async/await`, tratamento de erros e manipulação de dados. Todas as APIs abaixo são públicas e abertas para testes educacionais.

1) RestCountries — Países e Moedas

Endpoint: `https://restcountries.com/v3.1/name/brazil`
Tarefas:

1. Buscar os dados do Brasil e exibir: nome oficial, capital, população, região e moedas (código e nome).
2. Criar uma função que formata a população com separador de milhar.
3. Desafio: montar uma pequena tabela HTML com as informações e estilizar com CSS simples.

```
async function getCountry(name = "brazil") {
  const url = `https://restcountries.com/v3.1/name/${encodeURIComponent(name)}`;
  const res = await fetch(url);
  if (!res.ok) throw new Error(`HTTP ${res.status}`);
  const data = await res.json();
  const br = data[0];
  const moedas = Object.entries(br.currencies || {}).map(([code, cur]) => `${code} - ${cur.name}
      `);
  console.log({
    nomeOficial: br.name?.official,
    capital: br.capital?.[0],
    populacao: br.population.toLocaleString("pt-BR"),
    regiao: br.region,
```

```
    moedas
  });
}
getCountry();
```

2) Universidades no Brasil (Hipolabs)

Endpoint: `http://universities.hipolabs.com/search?country=Brazil`
Tarefas:

1. Listar as primeiras 20 universidades (nome e site).
2. Filtrar por estado ou por termo (ex.: "Federal") e exibir uma contagem.
3. Desafio: ordenar alfabeticamente e exportar para JSON (string) para download.

```
async function getUniversities(country = "Brazil") {
  const res = await fetch(`http://universities.hipolabs.com/search?country=${encodeURIComponent(
      country)}`);
  if (!res.ok) throw new Error(`HTTP ${res.status}`);
  const data = await res.json();
  const primeiras20 = data.slice(0, 20).map(u => ({
    nome: u.name,
    site: u.web_pages?.[0] || ""
  }));
  console.table(primeiras20);
}
getUniversities();
```

3) Posição atual da ISS (Open Notify)

Endpoint: `http://api.open-notify.org/iss-now.json`
Tarefas:

1. Buscar a latitude e longitude atuais da Estação Espacial Internacional (ISS).
2. Converter para número (`parseFloat`) e exibir com 4 casas decimais.
3. Desafio: desenhar um marcador em um mapa (ex.: Leaflet) usando as coordenadas.

```
async function getISSNow() {
  const res = await fetch("http://api.open-notify.org/iss-now.json");
  if (!res.ok) throw new Error(`HTTP ${res.status}`);
  const { iss_position } = await res.json();
  const lat = parseFloat(iss_position.latitude).toFixed(4);
  const lon = parseFloat(iss_position.longitude).toFixed(4);
  console.log(`ISS em lat: ${lat}, lon: ${lon}`);
}
getISSNow();
```

4) Astronautas no Espaço (Open Notify)

Endpoint: `http://api.open-notify.org/astros.json`
Tarefas:

1. Exibir quantas pessoas estão no espaço agora e em quais naves.
2. Agrupar os astronautas por nave e mostrar contagem por grupo.
3. Desafio: montar um pequeno relatório em Markdown e imprimir no console.

```
async function getAstros() {
  const res = await fetch("http://api.open-notify.org/astros.json");
  if (!res.ok) throw new Error(`HTTP ${res.status}`);
  const data = await res.json();
  const porNave = data.people.reduce((acc, p) => {
    acc[p.craft] = acc[p.craft] || [];
    acc[p.craft].push(p.name);
    return acc;
  }, {});
  console.log(`Pessoas no espao: ${data.number}`);
  Object.entries(porNave).forEach(([craft, nomes]) => {
    console.log(`- ${craft}: ${nomes.length} -> ${nomes.join(", ")}`);
  });
}
getAstros();
```

5) Poesia (PoetryDB)

Endpoint: `https://poetrydb.org/title/Ozymandias`
Tarefas:

1. Obter o poema "Ozymandias" e exibir título, autor e os primeiros 5 versos.
2. Desafio: contar palavras mais frequentes (ignorando pontuação e maiúsculas/minúsculas).
3. Desafio 2: criar um destaque visual em HTML para os 3 termos mais recorrentes.

```
async function getPoemByTitle(title = "Ozymandias") {
  const url = `https://poetrydb.org/title/${encodeURIComponent(title)}`;
  const res = await fetch(url);
  if (!res.ok) throw new Error(`HTTP ${res.status}`);
  const data = await res.json();
  const poem = data[0];
  console.log(`${poem.title}  ${poem.author}`);
  console.log(poem.lines.slice(0,5).join("\n"));
}
getPoemByTitle();
```

Dicas Gerais:

- Use `try/catch` para tratamento de erros e valide `res.ok`.
- Considere `Promise.all` para buscar múltiplos recursos em paralelo.
- Formate números com `toLocaleString("pt-BR")` quando fizer sentido.
- Caso ocorra erro de CORS no navegador, teste via Node (ESM) ou utilize um proxy/serviço que habilite CORS apenas para fins didáticos.