Revisão da Prova I – Desenvolvimento Web II – Prof. Arley

1 – Arrays são úteis para manipular grandes quantidades de dados. O operador spread é muito útil para manusearmos arrays. Marque a alternativa que contém o conteúdo da variável **r**.

```
const v = [4,8,5];
const w = [3.1,4.2];
const r = [...v,...w];
console.log(r);
```

a) [4, 8, 5]
b) [3.1, 4.2]
c) [12,7.3]
d) 19.3
e) [4, 8, 5, 3.1, 4.2]

2 – Array é um objeto global do JS usado na construção de arrays. Ele possui métodos que implementam operações comuns no manuseio de arrays. Marque a alternativa cuja variável possui conteúdo undefined, ou seja, o método correspondente possui retorno void.

```
const v = [4,8,5,9,1];
const a = v.forEach( e => e * 2);
const b = v.map( e => e * 2);
const c = v.reduce( (t,e) => t + e);
const d = v.slice(1);
console.log(a);
console.log(b);
console.log(c);
console.log(d);
```

a) forEach retorna void.
b) map retorna void.
c) reduce retorna void.
d) slice retorna void
e) Todas retornam void.

3 – O operador de spread pode ser utilizado com arrays e objetos na notação JS (JSON). Marque a alternativa que contém o conteúdo da variável **r**.

```
const cliente = {nome:"Ana",idade:21};
const r = {...cliente, peso:60};
console.log(r);
```

a) { nome: 'Ana', idade: 21}
b) { peso: 60 }
c) { peso: 60, nome: 'Ana', idade: 21 }
d) { nome: 'Ana', idade: 21, peso: 60 }
e) { nome: 'Ana', peso: 60 }

4 – A desestruturação é uma operação utilizada em objetos JSON. Marque a alternativa que possui uma desestruturação **inválida**.

```
const cliente = {
    nome:"Ana",
    idade:21,
    logradouro:"R. Faria Lima",
    cidade:"JCR"
};
```

a) const { nome, logradouro } = cliente;
b) const { idade } = cliente;
c) const { name } = cliente;
d) const { nome, logradouro, idade, cidade } = cliente;
e) const { cidade,idade } = cliente;

5 – O oposto da desestruturação é a estruturação de objetos JSON. Marque a alternativa correta considerando o código a seguir.

```
const carro = "Uno";
const marca = "Fiat";
const ano = 2010;
const a = {carro:carro,marca:marca, ano:ano};
const b = {"Uno","Fiat", 2010};
const c = {carro,marca, ano};
const d = {"Uno":carro,"Fiat":marca, 2010:ano};
```

a) Somente a instrução **a** está correta.
b) Somente as instruções **a** e **c** estão corretas.
c) Somente as instruções **b** e **d** estão corretas.
d) Todas as instruções estão corretas.

e) Nenhuma das instruções estão corretas.

6 - A desestruturação de arrays é feita usando colchetes [] no lado esquerdo da atribuição. Analise o trecho de código e marque a alternativa que possui os valores de **a** e **b**.

```
const v = [4,8,5,9,1];
const [a,b] = v;
```

a) [4,8,5,9,1]
b) 8
c) 4
d) [4,8]
e) 4 e 8

7 – Uma função assíncrona retorna um objeto Promise, que representa o compromisso de resultado. Os objetos Promise são manipulados através dos métodos then, catch e finally. Marque a alternativa que possui o uso correta da Promise.

```
const aleatorio = async function(){
    return Math.floor(Math.random() * 10);
}
```

a) aleatorio().then(r => console.log(r));
b) const b = aleatorio(r => console.log(r));
c) const c = aleatorio().then(console.log(r));
d) const d = aleatorio(); d.then(console.log(r));
e) const e = aleatorio.then(r => console.log(r));

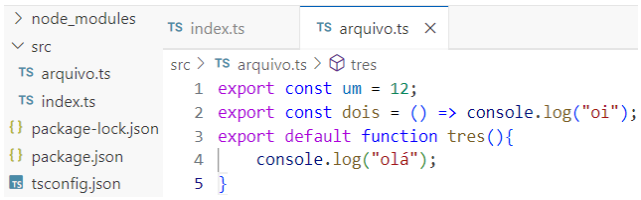
8 – O operador await (aguardar) pode ser usado somente no corpo de funções assíncronas para pausar a execução até que uma Promise seja resolvida. O operador await nos permite escrever código assíncrono de maneira síncrona, facilitando o tratamento de Promises. Marque a alternativa que possui a alteração no código para fazer o uso do operador await corretamente.

```
const aleatorio = async function () {
    return Math.floor(Math.random() * 10);
}

function exibir() {
    const a = aleatorio();
    const b = aleatorio();
    console.log(a + b);
}

exibir();
```

a)
```
async function exibir() {
  const a = aleatorio();
  const b = aleatorio();
  console.log(a + b);
}
await exibir();
```
b)
```
await function exibir() {
  const a = aleatorio();
  const b = aleatorio();
  console.log(a + b);
}
exibir();
```
c)
```
async function exibir() {
  const a = await aleatorio();
  const b = await aleatorio();
  console.log(a + b);
}
exibir();
```
d)
```
await function exibir() {
  const a = aleatorio();
  const b = aleatorio();
  console.log(await a + await b);
}
await exibir();
```
e)
```
await function exibir() {
  const a = async aleatorio();
  const b = async aleatorio();
  console.log(a + b);
}
await exibir();
```

9 – A exportação e importação é um recurso importante na modularização e reutilização de código. O código pode ser organizado em pastas e módulos e consumidos em diferentes partes do programa. Marque a alternativa que possui a importação dos recursos exportados no arquivo.ts.

Revisão da Prova I – Desenvolvimento Web II – Prof. Arley



a) import um, dois, tres from "./arquivo";

b) import tres, {um, dois} from "./arquivo";

c) import {um, dois, tres} from "./arquivo";

d) import um, dois, {tres} from "./arquivo";

e) import um, dois from "./arquivo";
import {tres} from "./arquivo";

10 – Um servidor web é um programa que responde numa porta às requisições HTTP. Marque a alternativa que possui a instrução para subir o servidor na porta 3100 considerando o código a seguir.

```
import express from "express";
const app = express();
app.listen(3100);
```

a) express.get(3100);

b) app.get(3100);

c) express.listen(3100)

d) app.use(3100);

e) app.listen(3100);

11 – Uma rota define um caminho específico no servidor para o qual as solicitações dos clientes são direcionadas e como essas solicitações são tratadas. definição de uma rota é formada pela combinação da URL com o método HTTP. Marque a alternativa que possui a rota para a URL a seguir.

```
GET http://localhost:3000/um/dois
```

a) app.get("/um/dois", (req,res) => {
const {um,dois} = req.params;
res.send("oi")
});

b) app.get("/{um}/{dois}", (req,res) => {
const {um,dois} = req.body;
res.send("oi")
});

c) app.get("/{um}/{dois}", (req,res) => {
const {um,dois} = req.params;
res.send("oi")
});

d) app.get("/um/dois", (_,res) => res.send("oi"));

e) app.get("/", (_,res) => res.send("oi"));

12 – Parâmetros podem ser enviados na requisição através do corpo ou na URL. Marque a alternativa que possui a instrução para receber no servidor os seguintes valores enviados pelo corpo da requisição.

```
{
  "x":1,
  "y":2
}
```

a) app.get("/", (req,res) => {
const {x,y} = req.params;
res.send("oi");
});

b) app.post("/", (req,res) => {
const {x,y} = req.body;
res.send("oi");
});

c) app.put("/", (req,res) => {
const {x,y} = req.json;
res.send("oi");
});

d) app.delete("/", (req,res) => {
const {x,y} = req.params;
res.send("oi");
});

e) app.get("/", (req,res) => {
const {x,y} = req;
res.send("oi");
});

13 – A ordem de definição das rotas influencia na escolha. Marque a alternativa que possui a rota a ser processado ao submeter a seguinte solicitação `GET http://localhost:3000/um/dois`

```
app.get("/", (req,res) => {
    res.send("a");
} );
app.post("/", (req,res) => {
    res.send("b");
} );
app.put("/", (req,res) => {
    res.send("c");
} );
app.delete("/", (req,res) => {
    res.send("d");
} );
app.use((req,res) => {
    res.send("e");
} );
```

a) a
b) b
c) c
d) d
e) e

14 – Um arquivo estático é aquele arquivo que será enviado para o cliente da forma como ele se encontra, isto é, ele não será executado no servidor. Marque a alternativa que possui a rota para servir qualquer arquivo disponível na pasta public definida na raiz do projeto.

a) app.use("/", public);
b) app.use("/", express.static('public'));
c) app.get("/", public);
d) app.get("/public", public);
e) app.get("/public", (req,res) => {});

15 – Em aplicações web complexas, a hierarquia de rotas é uma abordagem que permite organizar e estruturar as rotas de forma mais eficiente e modular. Podemos dividir o roteamento em diferentes arquivos e diretórios, tornando o código mais organizado, fácil de manter e escalável. Considerando o código a seguir, marque a alternativa que possui a requisição que retorna o texto <u>três</u>.

```
import express, { Router } from "express";
const app = express();
app.listen(3100);

const a = Router();
a.get("/", (req,res) => res.send("um"));
a.post("/", (req,res) => res.send("dois"));

const b = Router();
b.get("/", (req,res) => res.send("três"));
b.post("/", (req,res) => res.send("quatro"));

app.use("/a", a);
app.use("/b", b);
```

a) GET http://localhost:3100/
b) GET http://localhost:3100/a
c) POST http://localhost:3100/a
d) GET http://localhost:3100/b
e) POST http://localhost:3100/b