Ficha de Exercícios

Programação e Desenvolvimento Web

11 de outubro de 2024

Exercício 1: Definição de Funções e Manipulação de Personagens

1. Cria uma função chamada `createPlayer` que aceita `name`, `health`, e `strength` como parâmetros e retorna um objeto `player` com essas propriedades. Adiciona um método `attack` que imprime no console uma mensagem indicando que o jogador atacou, mencionando o `name` e o `strength` do jogador.

   Exemplo:

   ```
   const player1 = createPlayer("Mark", 100, 20);
   player1.attack();
   ```

2. Define uma função `takeDamage` para o objeto `player` que aceita um valor de `damage` e subtrai esse valor da `health` do jogador. Se a `health` for menor ou igual a 0, imprime no console que o jogador foi derrotado.

   Exemplo:

   ```
   player1.takeDamage(30);
   ```

Exercício 2: Inimigos e Objetos no Jogo

1. Cria uma função `createEnemy` que funciona de maneira semelhante à função `createPlayer`. Define `enemyType`, `health`, e `strength` como propriedades do objeto `enemy` e cria um método `attack` semelhante ao do `player`.

   Exemplo:

   ```
   const enemy1 = createEnemy("Goblin", 50, 15);
   enemy1.attack();
   ```

2. Adiciona um método `dropLoot` ao objeto `enemy` que devolve um objeto `loot` com uma `type` (como "moeda", "poção") e um valor `amount`. A função `dropLoot` deve gerar um tipo de loot aleatório com valores entre 1 e 50.

   Exemplo:

   ```
   const loot = enemy1.dropLoot();
   ```

Exercício 3: Arrays e Gestão de Inventário

1. Cria um array chamado `inventory` para armazenar itens de um jogador. Escreve uma função `addItem` que aceita um item como argumento e o adiciona ao `inventory`.

   Exemplo:

   ```
   addItem("Espada");
   ```

2. Define uma função `removeItem` que aceita o nome de um item e o remove do `inventory`. Se o item não estiver no inventário, imprime uma mensagem dizendo que o item não foi encontrado.

   Exemplo:

   ```
   removeItem("Espada");
   ```

Exercício 4: Map - Melhorias de Habilidade

1. Cria um array `skills` contendo objetos com `name` e `level`. Usa o método `map` para criar um novo array `upgradedSkills` que aumenta o `level` de cada habilidade em 1.

   Exemplo:

   ```
   const upgradedSkills = skills.map(increaseSkillLevel);
   ```

2. Define um array de `abilities`, onde cada habilidade tem uma `name` e um valor `power`. Usa o método `map` para criar um novo array `boostedAbilities`, aumentando o `power` de cada habilidade em 20%.

   Exemplo:

   ```
   const boostedAbilities = abilities.map(increaseAbilityPower);
   ```

Exercício 5: Filter - Filtros de Inventário e Desafios

1. Adiciona alguns itens ao `inventory`, incluindo itens de `type` ”moeda”, ”arma”, e ”poção”. Usa o método `filter` para criar um novo array `weapons` que contém apenas os itens que são do tipo ”arma”.

   Exemplo:

   ```
   const weapons = inventory.filter(isWeapon);
   ```

2. Cria um array `challenges` contendo objetos com `name` e `difficulty`. Usa `filter` para criar um novo array `hardChallenges` que inclui apenas os desafios com `difficulty` superior a um valor de 7.

   Exemplo:

   ```
   const hardChallenges = challenges.filter(isHard);
   ```

Exercício 6: Reduce - Cálculo de Pontuação e Saúde Total

1. Define um array `scores` com valores numéricos que representam pontuações de diferentes níveis. Usa `reduce` para calcular a pontuação total do jogador ao somar todos os valores do array.

   Exemplo:

   ```
   const totalScore = scores.reduce(sumScore);
   ```

2. Cria um array `healthItems` onde cada objeto representa um item de cura com uma `name` e um valor `restore`. Usa `reduce` para calcular a quantidade total de saúde que o jogador recuperaria ao utilizar todos os itens de `healthItems`.

   Exemplo:

   ```
   const totalHealthRestore = healthItems.reduce(sumHealthRestore);
   ```

Exercício 7: Criação de Um Pequeno Sistema de Combate

1. Usando as funções `createPlayer` e `createEnemy`, cria um jogador e um inimigo com valores específicos. Implementa uma função `combatRound` que permite ao jogador e ao inimigo atacarem-se mutuamente.

   Exemplo:

   ```
   combatRound(player1, enemy1);
   ```

2. Adiciona uma funcionalidade na função `combatRound` para que, após cada ataque, verifique se o jogador ou o inimigo têm saúde inferior ou igual a 0, declarando o vencedor do combate e terminando a função.

Exercício 8: Eventos Aleatórios e Simulação de Jogo

1. Cria uma função `randomEvent` que retorna um evento aleatório, como "encontrou um inimigo", "achou uma poção", ou "ganhou uma moeda". Usa esta função numa simulação onde o jogador avança e encontra eventos aleatórios.

   Exemplo:

   ```
   const event = randomEvent();
   ```

2. Implementa uma função `simulateGame` que faz uma série de chamadas a `randomEvent`, acumulando loot no inventário do jogador e o número de inimigos derrotados até que o jogador perca toda a sua saúde. No final, imprime um resumo com o número de eventos, loot recolhido e inimigos derrotados.

   Exemplo:

   ```
   simulateGame(player1);
   ```