RPG Game

`call()`, `apply()`, and `bind()`

1. Equip Weapon:

- **Task:** Given this function that equips a new weapon:

```
function equipWeapon(newWeapon) {
 this.weapon = newWeapon.name;
}
```

Use the relevant method (`call()` , `apply()` , or `bind()`) to execute this function with a different context.

- **Inputs:** A player object and a weapon object.
- **Example:**
  - Input: `Player object {name: 'Warrior', weapon: 'Sword'}` , `Weapon object {name: 'Axe', damage: 10}`
  - Output: `Player object {name: 'Warrior', weapon: 'Axe'}`

2. Calculate Player Damage:

- **Task:** Given this function that calculates the potential damage of a player:

```
function calculateDamage() {
 return this.strength + this.weapon.damage;
}
```

Use the relevant method to execute this function with a different context.

- **Inputs:** A player object.
- **Example:**
  - Input:
    `Player object {name: 'Warrior', strength: 10, weapon: {name: 'Axe', damage: 10}}`
  - Output: `20`

3. Bind a Player to a Spell Casting Method:

- **Task:** Given this function that allows a Player object to cast a spell:

```
function castSpell(spell) {
 if (this.mana >= spell.cost) {
   this.mana -= spell.cost;
   return `${this.name} casts ${spell.name}`;
 } else {
   return `${this.name} does not have enough mana`;
 }
}
```

Use the relevant method to permanently bind this function to a player context.

- **Inputs:** A player object and a spell object.
- **Example:**
  - Input: `Player object {name: 'Wizard', mana: 50}`, `Spell object {name: 'Fireball', cost: 10}`
  - Output: `[Function: bound castSpell]`

4. Display Player Spells:

- **Task:** Given this function that displays all the spells known by a player:

```
function displaySpells() {
 return this.spells.map(spell => spell.name);
}
```

Use the relevant method to invoke this function with different contexts.

- **Inputs:** A player object.
- **Example:**
  - Input: `Player object {name: 'Wizard', spells: [{ name: 'Fireball', damage: 10, cost: 10 }, { name: 'Lightning Bolt', damage: 15, cost: 20 }]}`
  - Output: `[ 'Fireball', 'Lightning Bolt' ]`

5. Calculate Average Damage of Player Spells:

- **Task:** Given this function that calculates the average damage of spells known by a player:

```
function calculateAverageSpellDamage() {
 const totalDamage = this.spells.reduce((acc, spell) => acc +
spell.damage, 0);
 return totalDamage / this.spells.length;
}
```

Use the relevant method to execute this function with a different context.

- **Inputs:** A player object.
- **Example:**
  - Input:
    `Player object {name: 'Wizard', spells: [{name: 'Fireball', damage: 10}, {name: 'Lightning Bolt', damage: 15}]}`
  - Output: `12.5`

6. Bind a Player to an Inventory Display Method:

- **Task:** Given this function that allows a Player object to display its inventory:

```
function displayInventory() {
 return this.inventory;
}
```

Use the relevant method to permanently bind this function to a player context.

- **Inputs:** A player object.
- **Example:**
  - Input:
    `Player object {name: 'Warrior', inventory: ['Health Potion', 'Mana Potion']}`
  - Output: `[Function: bound displayInventory]`

7. Update Player Strength:

- **Task:** Given this function for the Player object that updates its strength:

```
function updateStrength(newStrength) {
 this.strength = newStrength;
}
```

Use the relevant method to execute this function with a different context.

- **Inputs:** A player object and new strength value.
- **Example:**
  - Input: `Player object {name: 'Warrior', strength: 10}`, `New strength: 15`
  - Output: `Updated player object {name: 'Warrior', strength: 15}`

8. Filter Player Spells:

- **Task:** Given this function that filters the spells of a player based on a mana cost value:

```
function filterSpells(maxCost) {
 return this.spells.filter(spell => spell.cost <= maxCost);
}
```

Use the relevant method to execute this function with a different context.

- **Inputs:** A player object and a maximum mana cost value.
- **Example:**
  - Input:
    `Player object {name: 'Wizard', spells: [{name: 'Fireball', cost: 10}, {name: 'Teleport', cost: 20}]}`
    , `Maximum cost: [15]`
  - Output: `Filtered spells: [{name: 'Fireball', cost: 10}]`