Travel Planner

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

1. Update Destination:

- **Task:** Given this function that updates the destination:

```
function updateDestination(newDestination) {
 this.destination = newDestination;
}
```

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

- **Inputs:** A TravelPlan object and a new destination.
- **Example:**
  - Input: `TravelPlan object { traveler: 'John', destination: 'Paris' }` , `New destination: 'London'`
  - Output: `Updated TravelPlan object {traveler: 'John', destination: 'London'}`

2. Calculate Total Travel Cost:

- **Task:** Given this function that calculates the total cost of a travel plan:

```
function calculateTotalCost() {
 return this.flights.reduce((total, flight) => total + flight.cost, 0);
}
```

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

- **Inputs:** A TravelPlan object.
- **Example:**
  - Input: `TravelPlan object { traveler: 'John', destination: 'London', flights: [{ departure: 'NYC', arrival: 'London', cost: 500 }, { departure: 'London', arrival: 'NYC', cost: 500 }] }`

- Output: `1000`

3. Bind a Traveler to a Plan Creation Method:

- **Task:** Given this function that that allows a Traveler object to create a travel plan:

```
function createPlan(destination) {
 this.plans.push({ destination });
}
```

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

- **Inputs:** A Traveler object and a destination.
- **Example:**
  - Input: `Traveler object { name: 'John', budget: 2000, plans: [] }`, `Destination: 'London'`
  - Output: `{ name: 'John', budget: 2000, plans: [ { destination: 'London' } ]}`

4. Display Traveler Plans:

- **Task:** Given this method that displays all the plans of a traveler:

```
  function displayPlans() {
return this.plans.map(plan => plan.destination);
  }
```

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

- **Inputs:** A Traveler object.
- **Example:**
  - Input: `Traveler object { name: 'John', budget: 2000, plans: [ { destination: 'London' }, { destination: 'Paris' } ] }`
  - Output: `['London', 'Paris']`

5. Calculate Average Cost of Traveler Plans:

- **Task:** Given this method that calculates the average cost of plans made by a traveler:

```
function calculateAverageCost() {
  let totalCost = this.plans.reduce((total, plan) => total + plan.cost,
0);
  return totalCost / this.plans.length;
}
```

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

- **Inputs:** A Traveler object.
- **Example:**
  - Input:
    `Traveler object {name: 'John', plans: [{destination: 'London', cost: 1000}, {destination: 'Paris', cost: 500}]}`
  - Output: `750`

6. Bind a Traveler to a Plan Deletion Method:

- **Task:** Given this method method that allows a Traveler object to delete a travel plan:

```
function deletePlan(destination) {
 this.plans = this.plans.filter(plan => plan.destination !== destination);
}
```

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

- **Inputs:** A Traveler object.
- **Example:**
  - Input: `Traveler object {name: 'John', plans: ['London', 'Paris']}`
  - Output: `[Function: bound deletePlan]`

7. Update Traveler Budget:

- **Task:** Given this method that updates a budget:

  ```
  function updateBudget(newBudget) {
   this.budget = newBudget;
  }
  ```

  Use the relevant method to execute this function with a different context.
- **Inputs:** A traveler object and a new budget.

- **Example:**
  - Input: `Traveler object {name: 'John', budget: 2000}`, `New budget: 3000`
  - Output: `Traveler object {name: 'John', budget: 3000}`

8. Filter Traveler Plans by Cost:

- **Task:** Given this method that filters the plans of a traveler based on a maximum cost:

```
function filterPlansByCost(maxCost) {
 return this.plans.filter(plan => plan.cost <= maxCost);
}
```

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

- **Inputs:** A traveler object and a maximum cost value.
- **Example:**
  - Input:
    `Traveler object {name: 'John', plans: [{destination: 'London', cost: 1000}, {destination: 'Paris', cost: 1500}]}`
    , `Maximum cost: [1200]`
  - Output: `[{destination: 'London', cost: 1000}]`