This question evaluates your understanding of closures and first-class functions - functions in JavaScript are variables that can be returned and passed into other functions as arguments.

Clarification questions

- What if we call `sum()` at the start without any arguments?
  - We won't test this case. But we recommend throwing an `Error`.

Solution

The `sum` function returns functions until it is called without any arguments, then it will return the total sum so far. Since we need to retain a running sum of the values added so far, we can use a variable for that. How do we achieve that without classes? The answer is closures.

A closure is the combination of a function bundled together with references to its lexical environment (surrounding state). We can make use of closures to keep a reference to the running total.

We declare an inner function that will be returned for subsequent calls to the function. That function will determine if there's a value being passed in. If there is, we add to the running total and return itself so that subsequent calls can be made, otherwise we return the running total.

We can check for the presence of an argument by using `argument === undefined`. Note that we shouldn't be using `==` here because `null == undefined` is `true`.

JavaScript TypeScript

```javascript
/**
 * @param {number} valueA
 * @return {Function}
 */
export default function sum(valueA) {
  return function (valueB) {
    return valueB === undefined ? valueA : sum(valueA + valueB);
  };
}
```

Alternative approach: One-liner using arrow functions

Using arrow functions, this function can be written in a single line.

```js
const sum = (a) => (b) => b !== undefined ? sum(a + b) : a;
export default sum;
```

Edge cases

Intermediate results should be able to be used again for separate evaluations.

```js
const addTwo = sum(2);
addTwo(3)(); // 5
addTwo(4)(); // 6
addTwo(3)(4)(); // 9
```

Techniques

- Closures
- First-class functions
- `null` vs `undefined`

Notes

- Do not use `==` to check for presence of an argument because `null == undefined` is `true`.