Here’s a **complete, runnable TypeScript example** that compares two JSON objects and prints their differences in a clear way — without requiring external libraries.
It works recursively, handles nested objects/arrays, and shows added, removed, and changed values.

```
// diff-json.ts
type DiffResult = {
    added?: Record<string, any>;
    removed?: Record<string, any>;
    changed?: Record<string, { oldValue: any; newValue: any }>;
};

/**
 * Recursively finds differences between two JSON objects.
 * @param obj1 First JSON object
 * @param obj2 Second JSON object
 * @returns Object containing added, removed, and changed keys
 */
function diffJSON(obj1: any, obj2: any): DiffResult {
    const result: DiffResult = { added: {}, removed: {}, changed: {}
};

    // Keys present in obj1 or obj2
    const allKeys = new Set([...Object.keys(obj1 || {}),
...Object.keys(obj2 || {})]);

    for (const key of allKeys) {
        const val1 = obj1?.[key];
        const val2 = obj2?.[key];

        if (!(key in obj1)) {
            // Key added
            result.added![key] = val2;
        } else if (!(key in obj2)) {
            // Key removed
            result.removed![key] = val1;
        } else if (isObject(val1) && isObject(val2)) {
            // Nested object → recurse
            const nestedDiff = diffJSON(val1, val2);
            if (hasChanges(nestedDiff)) {
                result.changed![key] = { oldValue: val1, newValue:
val2 };
            }
        } else if (Array.isArray(val1) && Array.isArray(val2)) {
            if (JSON.stringify(val1) !== JSON.stringify(val2)) {
                result.changed![key] = { oldValue: val1, newValue:
val2 };
            }
        } else if (val1 !== val2) {
            // Value changed
            result.changed![key] = { oldValue: val1, newValue: val2
};
        }
    }

    // Clean up empty sections
    if (!Object.keys(result.added!).length) delete result.added;
    if (!Object.keys(result.removed!).length) delete result.removed;
    if (!Object.keys(result.changed!).length) delete result.changed;

    return result;
}

function isObject(value: any): boolean {
    return value && typeof value === "object" && !
Array.isArray(value);
}

function hasChanges(diff: DiffResult): boolean {
    return !!(diff.added || diff.removed || diff.changed);
}
```

```
// Example usage
const json1 = {
    name: "Alice",
    age: 25,
    address: { city: "Rome", zip: "00100" },
    hobbies: ["reading", "traveling"]
};

const json2 = {
    name: "Alice",
    age: 26,
    address: { city: "Milan", zip: "20100" },
    hobbies: ["reading", "sports"],
    email: "alice@example.com"
};

console.log("Differences:", JSON.stringify(diffJSON(json1, json2),
null, 2));
```

How it works

- **Added keys** → Present in **`obj2`** but not in **`obj1`**.
- **Removed keys** → Present in **`obj1`** but not in **`obj2`**.
- **Changed keys** → Present in both but with different values (including nested objects and arrays).
- Uses recursion for deep comparison.

Example Output

For the example above:

```
{
  "added": {
    "email": "alice@example.com"
  },
  "changed": {
    "age": {
      "oldValue": 25,
      "newValue": 26
    },
    "address": {
      "oldValue": { "city": "Rome", "zip": "00100" },
      "newValue": { "city": "Milan", "zip": "20100" }
    },
    "hobbies": {
      "oldValue": ["reading", "traveling"],
      "newValue": ["reading", "sports"]
    }
  }
}
```

If you prefer, I can also give you a **version using the `json-diff-ts` npm package** for a more compact solution.
Do you want me to provide that?