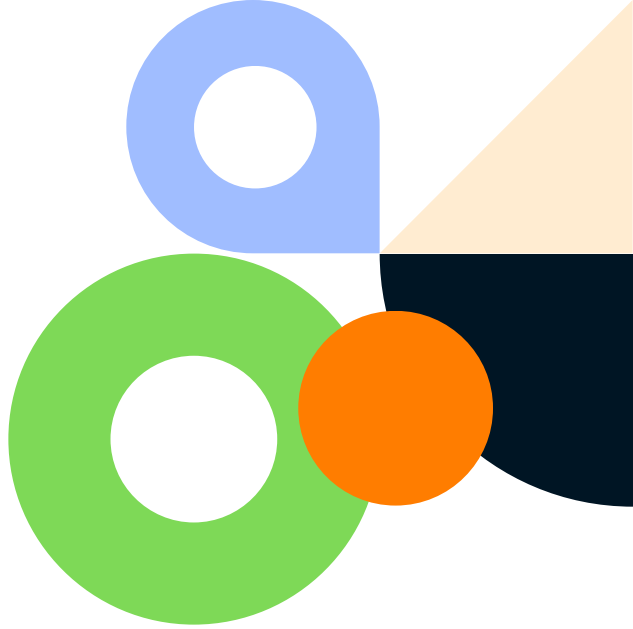
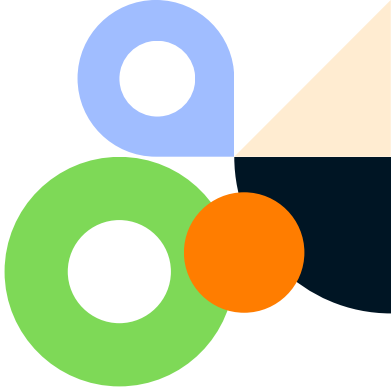


Use Cases for

Proxies

@munib-ehman

Watching for Changes



watching-for-changes.ts

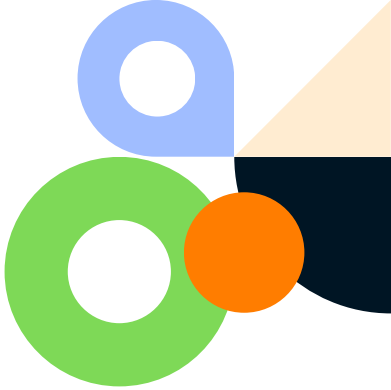
```
function onChange(obj, onChange) {
  const handler = {
    set: (target, property, value, receiver) => {
      onChange(`Property ${String(property)} changed to ${value}`);
      return Reflect.set(target, property, value, receiver);
    },
  };
  return new Proxy(obj, handler);
}

const person = { name: "John", age: 30 };
const watchedPerson = onChange(person, console.log);

watchedPerson.age = 31; // Console: Property age changed to 31
```

@muneeb-ehman

Smart Caching



smart-caching.ts

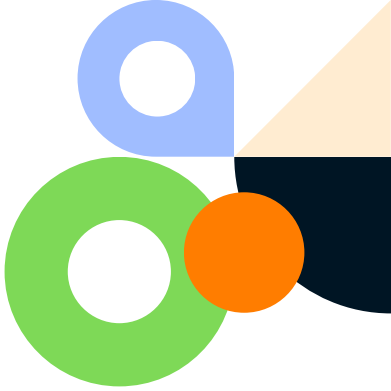
```ts
function smartCache<T extends object>(
  obj: T,
  fetcher: (key: keyof T) => any
): T {
  const cache: Partial<T> = {};
  return new Proxy(obj, {
    get: (target, property: keyof T) => {
      if (!cache[property]) {
        cache[property] = fetcher(property);
      }
      return cache[property];
    },
  });
}

const userData = smartCache({ userId: 1 }, (prop) => {
  console.log(`Fetching data for ${String(prop)}`);
  return { name: "Bob" }; // Simulated fetch
});

console.log(userData.userId);
// Output: Fetching data for userId, then returns { name: "Bob" }
```

@muneeb-ehman

Auto-populating Properties



auto-populating-properties.ts

```ts
type LazyProfile = {
  firstName: string;
  lastName: string;
  fullName?: string;
};

let lazyProfileHandler = {
  get: (target: LazyProfile, property: keyof LazyProfile) => {
    if (property === "fullName" && !target[property]) {
      target[property] = `${target.firstName} ${target.lastName}`;
    }
    return target[property];
  },
};

let profile: LazyProfile = new Proxy(
  { firstName: "John", lastName: "Doe" },
  lazyProfileHandler
);
console.log(profile.fullName); // Output: John Doe
```

@muneeb-ehman

Operation Counting

operation-counting.ts

```ts
type Counter = {
  [key: string]: any;
  _getCount: number;
};

let countHandler = {
  get: (target: Counter, property: keyof Counter) => {
    if (property === "_getCount") {
      return target[property];
    }
    target._getCount++;
    return target[property];
  },
};

let counter: Counter = new Proxy({ a: 1, b: 2, _getCount: 0
 }, countHandler);
counter.a;
counter.b;
console.log(counter._getCount); // Output: 2v
```

@muneeb-ehman

Dynamic Property Validation

dynamic-property-validation.ts

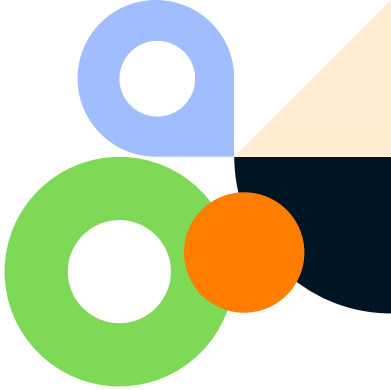
```
let user = {
  age: 25,
};

let validator = {
  set: (obj, prop, value) => {
    if (prop === "age" && (typeof value !== "number" || value < 18)) {
      throw new Error("User must be at least 18 years old.");
    }
    obj[prop] = value;
    return true; // Indicate success
  },
};

let userProxy = new Proxy(user, validator);
userProxy.age = 30; // Works fine
console.log(userProxy.age); // Output: 30
// userProxy.age = 'thirty';  // Throws error
// userProxy.age = 17;  // Throws error
```

@muneeb-ehman

Immutable Objects



immutable-objects.ts

```ts
function createImmutable<T extends object>(obj: T): T {
  return new Proxy(obj, {
    set: () => {
      throw new Error("This object is immutable");
    },
  });
}

const immutableObject = createImmutable({ name: "Jane", age: 25 });
// immutableObject.age = 26;  // Throws error
```

@muneeb-ehman