BuilderFns

Builder functions are the "nouns" of your models. They create buildable properties for faketastic, bundling all necessary information to build mock-data for a specified property.

The following builder functions are built into faketastic:

combine

```
combine<T extends object>(values: T, mapFn: T => any, ...attachedFns:
AttachedFn[])
```

Takes an object of values to be evaluated and combines them into a single value.

```
const Names = ['dorah.deen'];
const Domains = ['gmail'];

const $Student = model({
 email: combine(
    {
      name: oneOf(Names),
      number: 42,
      domain: oneOf(Domains),
    },
    values => `${values.name}${values.number}@${values.domain}.de`,
  );
});

const student = build($Student);
// => { email: "dorah.deen42@gmail.de" }
```

Example: Evaluates all the given values and combines them as implemented within a mapping function.

itself

```
itself(endWhen: RecursionController, ...attachedFns: AttachedFn[])
```

Enables models to be recursive by referencing the model it located on.

```
import { /* ... */, RecursionDepth } from 'faketastic';

const $Directory = model({
 name: oneOf(DirNames),
 directories: itself(RecursionDepth(null, 1, 1)),
});

const dir = build($Directory);
// => {
```

```
//   name: 'Documents',
//   directories: {
//      name: '.trash',
//      directories: null,
//   }
// }
```

Example: Builds a recursive model, with a recursion depth that ranges from at least 1 recursion to at most 1 recursion (=> exact one recursion depth) and ends the recursion with the value `null`.

oneOf

```
oneOf(values: any[], ...attachedFns: AttachedFn[])
```

Chooses a random item from a given array of items. The randomly chosen item can be a model again.

```
const $Person = model({ age: range(1, 42) });
oneOf([4711, $Person]);
// => { age: 22 }
```

Example: oneOf chooses a random item, which can be also a model.

someOf

```
someOf(values: any[], options?: SomeOfOpts, ...attachedFns: AttachedFn[])
```

Chooses some random items from a given array. Those items can be models again.

```
const $Person = model({ age: range(1, 42) });
someOf([4711, $Person]);
// => [ { age: 22 }, 4711 ]
```

Example: someOf chooses random items, which can also be models.

ref

```
ref<T = any>(property: keyof T, ...attachedFns: AttachedFn[])
```

References the specified property and resolves its value at "finalizer" build-cycle.

```
const $Person = model({
  age: range(1, 42),
  ageRef: ref('age'),
});
// => { age: 28, ageRef: 28 }
```

> Example: `ageRef` will resolve the value of `age`-property at "finalizer" build cycle.

Related Topics

- AttachedFns
- ValueFns
- FactoryFns
- Overview
- Getting Started