AsyncValue, AsyncValueX

methods

```
R when<R>({
  bool skipLoadingOnReload = false,
  bool skipLoadingOnRefresh = true,
  bool skipError = false,
  required R Function(T data) data,
  required R Function(Object error, StackTrace stackTrace) error,
  required R Function() loading,
}) {
  if (isLoading) {
    bool skip;
    if (isRefreshing) {
      skip = skipLoadingOnRefresh;
    } else if (isReloading) {
      skip = skipLoadingOnReload;
    } else {
      skip = false;
    }
    if (!skip) return loading();
  }

  if (hasError && (!hasValue || !skipError)) {
    return error(this.error!, stackTrace!);
  }

  return data(requireValue);
}
```

```
R maybeWhen<R>({
  bool skipLoadingOnReload = false,
  bool skipLoadingOnRefresh = true,
  bool skipError = false,
  R Function(T data)? data,
  R Function(Object error, StackTrace stackTrace)? error,
  R Function()? loading,
  required R Function() orElse,
}) {
  return when(
    skipError: skipError,
    skipLoadingOnRefresh: skipLoadingOnRefresh,
    skipLoadingOnReload: skipLoadingOnReload,
    data: data ?? (_) => orElse(),
    error: error ?? (err, stack) => orElse(),
    loading: loading ?? () => orElse(),
  );
}
```

```dart
R? whenOrNull<R>({
  bool skipLoadingOnReload = false,
  bool skipLoadingOnRefresh = true,
  bool skipError = false,
  R? Function(T data)? data,
  R? Function(Object error, StackTrace stackTrace)? error,
  R? Function()? loading,
}) {
  return when(
    skipError: skipError,
    skipLoadingOnRefresh: skipLoadingOnRefresh,
    skipLoadingOnReload: skipLoadingOnReload,
    data: data ?? (_) => null,
    error: error ?? (err, stack) => null,
    loading: loading ?? () => null,
  );
}
```

```
R map<R>({
  required R Function(AsyncData<T> data) data,
  required R Function(AsyncError<T> error) error,
  required R Function(AsyncLoading<T> loading) loading,
});
```

```
R mayBeMap<R>({
  R Function(AsyncData<T> data)? data,
  R Function(AsyncError<T> error)? error,
  R Function(AsyncLoading<T> loading)? loading,
  required R Function() orElse,
}) {}
```

```
R? mapOrNull<R>({
  R? Function(AsyncData<T> data)? data,
  R? Function(AsyncError<T> error)? error,
  R? Function(AsyncLoading<T> loading)? loading,
}) {}
```

Your Dev Edge YDE

```
R when<R>({
  skipLoadingOnReload = false,
  skipLoadingOnRefresh = true,
  skipError = false,
  required R Function(T data) data,
  required R Function(Object error, StackTrace stackTrace) error,
  required R Function() loading,
})
```

```
R map<R>({
  required R Function(AsyncData<T> data) data,
  required R Function(AsyncError<T> error) error,
  required R Function(AsyncLoading<T> loading) loading,
})
```

```dart
@sealed
@immutable
abstract class AsyncValue<T> {

  ...

  AsyncValue<T> copyWithPrevious(
    AsyncValue<T> previous, {
    bool isRefresh = true,
  });

  ...

}
```

```
class AsyncData<T> extends AsyncValue<T> {

  ...

  @override
  AsyncData<T> copyWithPrevious(
    AsyncValue<T> previous, {
    bool isRefresh = true,
  }) {
    return this;
  }

  ...

}
```

```
const AsyncData(T value)
    : this._(
        value,
        isLoading: false,
        error: null,
        stackTrace: null,
       );

const AsyncData._(
  this.value, {
  required this.isLoading,
  required this.error,
  required this.stackTrace,
}) : super._();
```

<constructor>

```
class AsyncLoading<T> extends AsyncValue<T> {
  ...
  @override
  AsyncValue<T> copyWithPrevious(
    AsyncValue<T> previous, {
    bool isRefresh = true,
  }) {
    if (isRefresh) {
      return previous.map();
    } else {
      return previous.map();
    }
  }
  ...
}
```

```
const AsyncLoading()
    : hasValue = false,
      value = null,
      error = null,
      stackTrace = null,
      super._();

const AsyncLoading._({
  required this.hasValue,
  required this.value,
  required this.error,
  required this.stackTrace,
}) : super._();
```

<constructor>

isRefresh: false

```
return previous.map(
  data: (d) => AsyncLoading._(
    hasValue: true,
    value: d.valueOrNull,
    error: d.error,
    stackTrace: d.stackTrace,
  ),
  error: (e) => AsyncLoading._(
    hasValue: e.hasValue,
    value: e.valueOrNull,
    error: e.error,
    stackTrace: e.stackTrace,
  ),
  loading: (e) => e,
);
```

isRefresh: true

```
return previous.map(
  data: (d) => AsyncData._(
    d.value,
    isLoading: true,
    error: d.error,
    stackTrace: d.stackTrace,
  ),
  error: (e) => AsyncError._(
    e.error,
    isLoading: true,
    value: e.valueOrNull,
    stackTrace: e.stackTrace,
    hasValue: e.hasValue,
  ),
  loading: (_) => this,
);
```

```dart
class AsyncError<T> extends AsyncValue<T> {
  ...
  @override
  AsyncError<T> copyWithPrevious(
    AsyncValue<T> previous, {
    bool isRefresh = true,
  }) {
    return AsyncError._(
      error,
      stackTrace: stackTrace,
      isLoading: isLoading,
      value: previous.valueOrNull,
      hasValue: previous.hasValue,
    );
  }
  ...
}
```

```dart
const AsyncError(Object error, StackTrace stackTrace)
    : this._(
        error,
        stackTrace: stackTrace,
        isLoading: false,
        hasValue: false,
        value: null,
       );

const AsyncError._(
  this.error, {
  required this.stackTrace,
  required T? value,
  required this.hasValue,
  required this.isLoading,
}) : _value = value,
     super._();
```

<constructor>

Your Dev Edge

error ▶ dialog

previous data ▶ main UI

opaque loading indicator ▶ semi-transparent loading indicator

previous data ▶ main UI