Stdlib	Containers	Batteries	Base
	CCResult.(<$>) : ('a -> 'b) -> ('a, 'err) result -> ('b, 'err) result		
	CCResult.(<*>) : ('a -> 'b, 'err) result -> ('a, 'err) result -> ('b, 'err) result		
	CCResult.(>>=) : ('a, 'err) result -> ('a -> ('b, 'err) result) -> ('b, 'err) result		Base.Result.(>>=) : ('a, 'e) result-> ('a -> ('b, 'e) result) -> ('b, 'e) result
	CCResult.(>|=) : ('a, 'err) result -> ('a -> 'b) -> ('b, 'err) result		Base.Result.(>>|) : ('a, 'e) result-> ('a -> 'b) -> ('b, 'e) result
	CCResult.(and*) : ('a, 'e) result -> ('b, 'e) result -> ('a * 'b, 'e) result		
	CCResult.(and+) : ('a, 'e) result -> ('b, 'e) result -> ('a * 'b, 'e) result		
	CCResult.(let*) : ('a, 'e) result -> ('a -> ('b, 'e) result) -> ('b, 'e) result		
	CCResult.(let+) : ('a, 'e) result -> ('a -> 'b) -> ('b, 'e) result		
	CCResult.add_ctx : string -> ('a, string) result -> ('a, string) result		
	CCResult.add_ctxf : ('a, Format.formatter, unit, ('b, string) result -> ('b, string) result) format4 -> 'a		
			Base.Result.all : ('a, 'e) result list -> ('a list, 'e) result
			Base.Result.all_unit : (unit, 'e) result list -> (unit, 'e) result
Result.bind : ('a, 'e) result -> ('a -> ('b, 'e) result) -> ('b, 'e) result		BatResult.bind : ('a, 'e) result -> ('a -> ('b, 'e) result) -> ('b, 'e) result	Base.Result.bind : ('a, 'e) result-> f:('a -> ('b, 'e) result) -> ('b, 'e) result
	CCResult.both : ('a, 'err) result -> ('b, 'err) result -> ('a * 'b, 'err) result		
	CCResult.catch : ('a, 'err) result -> ok:('a -> 'b) -> err:('err -> 'b) -> 'b		
		BatResult.catch : ('a -> 'e) -> 'a -> ('e, exn) result	
		BatResult.catch2 : ('a -> 'b -> 'c) -> 'a -> 'b -> ('c, exn) result	
		BatResult.catch3 : ('a -> 'b -> 'c -> 'd) -> 'a -> 'b -> 'c -> ('d, exn) result	
	CCResult.choose : ('a, 'err) result list -> ('a, 'err list) result		
			Base.Result.combine : ('ok1, 'err) result-> ('ok2, 'err) result-> ok:('ok1 -> 'ok2 -> 'ok3) -> err:('err -> 'err -> 'err) -> ('ok3, 'err) result
			Base.Result.combine_errors : ('ok, 'err) result list -> ('ok list, 'err list) result
			Base.Result.combine_errors_unit : (unit, 'err) result list -> (unit, 'err list) result
Result.compare : ok:('a -> 'a -> int) -> error:('e -> 'e -> int) -> ('a, 'e) result -> ('a, 'e) result -> int	CCResult.compare : err:'err CCResult.ord -> 'a CCResult.ord -> ('a, 'err) result CCResult.ord	BatResult.compare : ok:('a -> 'a -> int) -> error:('e -> 'e -> int) -> ('a, 'e) result -> ('a, 'e) result -> int	Base.Result.compare : ('ok -> 'ok -> int) -> ('err -> 'err -> int) -> ('ok, 'err) result-> ('ok, 'err) result-> int
		BatResult.default : 'a -> ('a, 'b) result -> 'a	
Result.equal : ok:('a -> 'a -> bool) -> error:('e -> 'e -> bool) -> ('a, 'e) result -> ('a, 'e) result -> bool	CCResult.equal : err:'err CCResult.equal -> 'a CCResult.equal -> ('a, 'err) result CCResult.equal	BatResult.equal : ok:('a -> 'a -> bool) -> error:('e -> 'e -> bool) -> ('a, 'e) result -> ('a, 'e) result -> bool	Base.Result.equal : ('ok -> 'ok -> bool) -> ('err -> 'err -> bool) -> ('ok, 'err) result-> ('ok, 'err) result-> bool
Result.error : 'e -> ('a, 'e) result	CCResult.fail : 'err -> ('a, 'err) result	BatResult.error : 'e -> ('a, 'e) result	Base.Result.fail : 'err -> ('a, 'err) result
			Base.Result.error : ('a, 'err) result-> 'err option
			Base.Result.failf : ('a, unit, string, ('b, string) result) format4 -> 'a
	CCResult.fail_fprintf : ('a, Format.formatter, unit, ('b, string) result) format4 -> 'a		

Stdlib	Containers	Batteries	Base
	CCResult.fail_printf : ('a, Buffer.t, unit, ('b, string) result) format4 -> 'a		
	CCResult.flat_map : ('a -> ('b, 'err) result) -> ('a, 'err) result -> ('b, 'err) result		
	CCResult.flatten_l : ('a, 'err) result list -> ('a list, 'err) result		
Result.fold : ok:('a -> 'c) -> error:('e -> 'c) -> ('a, 'e) result -> 'c	CCResult.fold : ok:('a -> 'b) -> error:('err -> 'b) -> ('a, 'err) result -> 'b	BatResult.fold : ok:('a -> 'c) -> error:('e -> 'c) -> ('a, 'e) result -> 'c	
	CCResult.fold_iter : ('b -> 'a -> ('b, 'err) result) -> 'b -> 'a CCResult.iter -> ('b, 'err) result		
	CCResult.fold_l : ('b -> 'a -> ('b, 'err) result) -> 'b -> 'a list -> ('b, 'err) result		
	CCResult.fold_ok : ('a -> 'b -> 'a) -> 'a -> ('b, 'c) result -> 'a		
		BatResult.get : ('a, exn) result -> 'a	
Result.get_error : ('a, 'e) result -> 'e		BatResult.get_error : ('a, 'e) result -> 'e	
	CCResult.get_lazy : ('b -> 'a) -> ('a, 'b) result -> 'a		
Result.get_ok : ('a, 'e) result -> 'a	CCResult.get_exn : ('a, 'b) result -> 'a	BatResult.get_ok : ('a, 'e) result -> 'a	
	CCResult.get_or : ('a, 'b) result -> default:'a -> 'a		
	CCResult.get_or_failwith : ('a, string) result -> 'a		
	CCResult.guard : (unit -> 'a) -> ('a, exn) result		
	CCResult.guard_str : (unit -> 'a) -> ('a, string) result		
	CCResult.guard_str_trace : (unit -> 'a) -> ('a, string) result		
			Base.Result.hash_fold_t : (Base.Ppx_hash_lib.Std.Hash.state -> 'ok -> Base.Ppx_hash_lib.Std.Hash.state) -> (Base.Ppx_hash_lib.Std.Hash.state -> 'err -> Base.Ppx_hash_lib.Std.Hash.state) -> Base.Ppx_hash_lib.Std.Hash.state -> ('ok, 'err) result-> Base.Ppx_hash_lib.Std.Hash.state
			Base.Result.ignore_m : ('a, 'e) result-> (unit, 'e) result
			Base.Result.invariant : 'a Base.Invariant_intf.inv -> 'b Base.Invariant_intf.inv -> ('a, 'b) resultBase.Invariant_intf.inv
		BatResult.is_bad : ('a, 'e) result -> bool	
Result.is_error : ('a, 'e) result -> bool	CCResult.is_error : ('a, 'err) result -> bool	BatResult.is_error : ('a, 'e) result -> bool	Base.Result.is_error : ('a, 'b) result-> bool
		BatResult.is_exn : exn -> ('a, exn) result -> bool	
Result.is_ok : ('a, 'e) result -> bool	CCResult.is_ok : ('a, 'err) result -> bool	BatResult.is_ok : ('a, 'e) result -> bool	Base.Result.is_ok : ('a, 'b) result-> bool
Result.iter : ('a -> unit) -> ('a, 'e) result -> unit	CCResult.iter : ('a -> unit) -> ('a, 'b) result -> unit	BatResult.iter : ('a -> unit) -> ('a, 'e) result -> unit	Base.Result.iter : ('ok, 'a) result-> f:('ok -> unit) -> unit
Result.iter_error : ('e -> unit) -> ('a, 'e) result -> unit	CCResult.iter_err : ('err -> unit) -> ('a, 'err) result -> unit	BatResult.iter_error : ('e -> unit) -> ('a, 'e) result -> unit	Base.Result.iter_error : ('a, 'err) result-> f:('err -> unit) -> unit
Result.join : (('a, 'e) result, 'e) result -> ('a, 'e) result	CCResult.join : (('a, 'err) CCResult.t, 'err) result -> ('a, 'err) result	BatResult.join : (('a, 'e) result, 'e) result -> ('a, 'e) result	Base.Result.join : (('a, 'e) result, 'e) result-> ('a, 'e) result
Result.map : ('a -> 'b) -> ('a, 'e) result -> ('b, 'e) result	CCResult.map : ('a -> 'b) -> ('a, 'err) result -> ('b, 'err) result	BatResult.map : ('a -> 'b) -> ('a, 'e) result -> ('b, 'e) result	Base.Result.map : ('ok, 'err) result-> f:('ok -> 'c) -> ('c, 'err) result
	CCResult.map2 : ('a -> 'b) -> ('err1 -> 'err2) -> ('a, 'err1) result -> ('b, 'err2) result		
		BatResult.map_both : ('a1 -> 'a2) -> ('b1 -> 'b2) -> ('a1, 'b1) result -> ('a2, 'b2) result	
		BatResult.map_default : 'b -> ('a -> 'b) -> ('a, 'c) result -> 'b	
Result.map_error : ('e -> 'f) -> ('a, 'e) result -> ('a, 'f) result	CCResult.map_err : ('err1 -> 'err2) -> ('a, 'err1) result -> ('a, 'err2) result	BatResult.map_error : ('e -> 'f) -> ('a, 'e) result -> ('a, 'f) result	Base.Result.map_error : ('ok, 'err) result-> f:('err -> 'c) -> ('ok, 'c) result

Stdlib	Containers	Batteries	Base
	CCResult.map_l : ('a -> ('b, 'err) result) -> 'a list -> ('b list, 'err) result		
	CCResult.map_or : ('a -> 'b) -> ('a, 'c) result -> default:'b -> 'b		
			Base.Result.of_either : ('ok, 'err) Base.Either0.t -> ('ok, 'err) result
	CCResult.of_err : ('a, 'b) CCResult.error -> ('a, 'b) result		
	CCResult.of_exn : exn -> ('a, string) result		
	CCResult.of_exn_trace : exn -> ('a, string) result		
	CCResult.of_opt : 'a option -> ('a, string) result	BatResult.of_option : 'a option -> ('a, unit) result	Base.Result.of_option : 'ok option -> error:'err -> ('ok, 'err) result
Result.ok : 'a -> ('a, 'e) result		BatResult.ok : 'a -> ('a, 'b) result	
			Base.Result.ok_exn : ('ok, exn) result-> 'ok
			Base.Result.ok : ('ok, 'a) result-> 'ok option
			Base.Result.ok_fst : ('ok, 'err) result-> ('ok, 'err) Base.Either0.t
			Base.Result.ok_if_true : bool -> error:'err -> (unit, 'err) result
			Base.Result.ok_or_failwith : ('ok, string) result-> 'ok
	CCResult.pp : 'a CCResult.printer -> ('a, string) result CCResult.printer		
	CCResult.pp' : 'a CCResult.printer -> 'e CCResult.printer -> ('a, 'e) result CCResult.printer		
		BatResult.print : ('b BatInnerIO.output -> 'a -> unit) -> 'b BatInnerIO.output -> ('a, exn) result -> unit	
	CCResult.pure : 'a -> ('a, 'err) result		
	CCResult.retry : int -> (unit -> ('a, 'err) result) -> ('a, 'err list) result		
	CCResult.return : 'a -> ('a, 'err) result		Base.Result.return : 'a -> ('a, 'b) result
			Base.Result.sexp_of_t : ('a -> Sexplib0__.Sexp.t) -> ('b -> Sexplib0__.Sexp.t) -> ('a, 'b) result-> Sexplib0__.Sexp.t
			Base.Result.t_of_sexp : (Sexplib0__.Sexp.t -> 'a) -> (Sexplib0__.Sexp.t -> 'b) -> Sexplib0__.Sexp.t -> ('a, 'b) result
			Base.Result.to_either : ('ok, 'err) result-> ('ok, 'err) Base.Either0.t
	CCResult.to_err : ('a, 'b) result -> ('a, 'b) CCResult.error		
	CCResult.to_iter : ('a, 'b) result -> 'a CCResult.iter		
Result.to_list : ('a, 'e) result -> 'a list		BatResult.to_list : ('a, 'e) result -> 'a list	
Result.to_option : ('a, 'e) result -> 'a option	CCResult.to_opt : ('a, 'b) result -> 'a option	BatResult.to_option : ('a, 'b) result -> 'a option	
Result.to_seq : ('a, 'e) result -> 'a Seq.t	CCResult.to_seq : ('a, 'b) result -> 'a Seq.t	BatResult.to_seq : ('a, 'e) result -> 'a BatSeq.t	
			Base.Result.try_with : (unit -> 'a) -> ('a, exn) result
Result.value : ('a, 'e) result -> default:'a -> 'a		BatResult.value : ('a, 'e) result -> default:'a -> 'a	
	CCResult.wrap1 : ('a -> 'b) -> 'a -> ('b, exn) result		
	CCResult.wrap2 : ('a -> 'b -> 'c) -> 'a -> 'b -> ('c, exn) result		
	CCResult.wrap3 : ('a -> 'b -> 'c -> 'd) -> 'a -> 'b -> 'c -> ('d, exn) result		