Circuit breaker strategy **CHEAT SHEET**

Basics

This reactive resilience strategy **allows you to shortcut execution** if the underlying resource detected as unhealthy.

You can configure the behaviour of the strategy via the **CircuitBreakerStrategyOptions{<T>}**.

This is a stateful strategy and should be **shared across multiple invocations**.

In **Closed** state the circuit allows traffic to pass through and it monitors the failures.
In **Open** state the circuit blocks traffic.
In **HalfOpen** state the circuit allows a single request to pass through as a probe.

The circuit shortcuts the execution with a **BrokenCircuitException** if it was in Open state.
The circuit shortcuts the execution with an **IsolatedCircuitException** if it was Isolated state.

Specify sampling period + monitor exceptions

```
new ResiliencePipelineBuilder()
  .AddCircuitBreaker(new CircuitBreakerStrategyOptions()
  {
    ShouldHandle = new PredicateBuilder().Handle<CustomException>(),
    FailureRatio = 0.5,
    SamplingDuration = TimeSpan.FromSeconds(10),
    MinimumThroughput = 10,
  })
```

Specify state access + monitor unsuccessful responses

```
var stateProvider = new CircuitBreakerStateProvider();
new ResiliencePipelineBuilder<HttpResponseMessage>()
  .AddCircuitBreaker(new CircuitBreakerStrategyOptions<HttpResponseMessage>()
  {
    ShouldHandle = new PredicateBuilder<HttpResponseMessage>()
      .HandleResult(res => !res.IsSuccessStatusCode),
    StateProvider = stateProvider
  })
…
if (stateProvider.CircuitState is not CircuitState.Open and not CircuitState.Isolated)
{
  …
}
```

Specify sleep duration + notifications

```
new ResiliencePipelineBuilder<string>()
  .AddCircuitBreaker(new CircuitBreakerStrategyOptions<string>()
  {
    BreakDuration = TimeSpan.FromSeconds(5),
    OnOpened = async args => await NotifyToOpen(args.BreakDuration),
    OnClosed = async args => await NotifyToClose(args.Outcome),
    OnHalfOpen = async args => await NotifyToHalfOpen(args.Context)
  })
```