the developer tool for Riverpod users

- help developers avoid common mistakes
- simplify repetitive tasks

Optional, but highly recommended to use

pubspec.yaml (dev_dependencies)

- riverpod_lint
- custom_lint

(caution)

- reload vscode window
- rerun vscode
- from command line
  dart run custom_lint

analysis_options.yaml

```
analyzer:
  plugins:
    - custom_lint
```

```
custom_lint:
  rules:
    - missing_provider_scope: false
```

```
custom_lint:
  enable_all_lint_rules: false
  rules:
    -missing_provider_scope
```

provider_parameters

```dart
// Parameters may override ==
ref.watch(myProvider(ClassThatOverridesEqual()));

// Constant objects are canonalized and therefore have a consistent ==
ref.watch(myProvider(const Object()));
ref.watch(myProvider(const [42]));

// Passing a variable disable the lint, as the variable may be cached.
ref.watch(myProvider(variable));
```

unsupported_provider_value

```
@riverpod
class CounterNotifier extends _$CounterNotifier {
 @override
 int build() => 0;

 void increment() => state++;
}
```

```
@riverpod
CounterNotifier counterNotifier(
  CounterNotifierRef ref
) => MyStateNotifier();

class CounterNotifier extends StateNotifier<int> {
  CounterNotifier(): super(0);

  void increment() => state++;
}
```

```
@riverpod
CounterNotifier counterNotifier(
  CounterNotifier.new
);

class CounterNotifier extends Notifier<int> {
 @override
 int build() => 0;

 void increment() => state++;
}
```

Your Dev Edge YDE

unsupported_provider_value

Sometimes, we need to return ChangeNotifier

go_router version below 9.0

```
class GoRouter extends ChangeNotifier implements RouterConfig<RouteMatchList>
```

go_router version 9.0 or later

```
class GoRouter implements RouterConfig<RouteMatchList>
```

```dart
// By using "Raw", we can explicitly return a ChangeNotifier in a provider
// without triggering `unsupported_provider_value`.
@riverpod
Raw<GoRouter> myRouter(MyRouterRef ref) {
  final router = GoRouter(...);
  // Riverpod won't dispose the ChangeNotifier for you in this case. Don't forget
  // to do it on your own!
  ref.onDispose(router.dispose);
  return router;
}
```

functional_ref, notifier_extends, avoid_ref_inside_state_dispose

```
@riverpod
int myProvider(MyProviderRef ref) => 0;
```

```
@riverpod
class Example extends _$Example {
  int build() => 0;
}
```

```
class _MyWidgetState extends ConsumerState<MyWidget> {
  @override
  void dispose() {
    // Do not use 'ref' in the dispose method
    ref.read(provider).doSomething();
    super.dispose();
  }

  // ...
}
```

Uncovered rules

- provider_dependencies

- scoped_providers_should_specify_dependencies

All assists

- Convert widget to ConsumerWidget
- Convert widget to ConsumerStatefulWidget
- Wrap widgets with a Consumer
- Wrap widgets with a ProviderScope
- Convert functional @riverpod to class variant
- Convert class @riverpod to functional variant

analysis_options.yaml

```
analyzer:
  plugins:
    - custom_lint
```

```
custom_lint:
  rules:
    - missing_provider_scope: false
```

```
custom_lint:
  enable_all_lint_rules: false
  rules:
    -missing_provider_scope
```