Lints

`rustdoc` provides lints to help you writing and testing your documentation. You can use them like any other lints by doing this:

```
#![allow(rustdoc::broken_intra_doc_links)] // allows the lint, no diagnostics will be repor
#![warn(rustdoc::broken_intra_doc_links)] // warn if there are broken intra-doc links
#![deny(rustdoc::broken_intra_doc_links)] // error if there are broken intra-doc links
```

Note that, except for `missing_docs`, these lints are only available when running `rustdoc`, not `rustc`.

Here is the list of the lints provided by `rustdoc`:

`broken_intra_doc_links`

This lint **warns by default**. This lint detects when an intra-doc link fails to be resolved. For example:

```
/// I want to link to [`Nonexistent`] but it doesn't exist!
pub fn foo() {}
```

You'll get a warning saying:

```
warning: unresolved link to `Nonexistent`
 --> test.rs:1:24
  |
1 | /// I want to link to [`Nonexistent`] but it doesn't exist!
  |                        ^^^^^^^^^^^^^ no item named `Nonexistent` in `test`
```

It will also warn when there is an ambiguity and suggest how to disambiguate:

```
/// [`Foo`]
pub fn function() {}

pub enum Foo {}

pub fn Foo(){}
```

```
warning: `Foo` is both an enum and a function
 --> test.rs:1:6
  |
1 | /// [`Foo`]
  |      ^^^^^ ambiguous link
  |
  = note: `#[warn(rustdoc::broken_intra_doc_links)]` on by default
help: to link to the enum, prefix with the item type
  |
1 | /// [`enum@Foo`]
  |      ^^^^^^^^^^
```

```
help: to link to the function, add parentheses
  |
1 | /// [`Foo()`]
  |      ^^^^^^^
```

private_intra_doc_links

This lint **warns by default**. This lint detects when intra-doc links from public to private items. For example:

```
#![warn(rustdoc::private_intra_doc_links)] // note: unnecessary - warns by default.

/// [private]
pub fn public() {}
fn private() {}
```

This gives a warning that the link will be broken when it appears in your documentation:

```
warning: public documentation for `public` links to private item `private`
 --> priv.rs:1:6
  |
1 | /// [private]
  |      ^^^^^^^ this item is private
  |
  = note: `#[warn(rustdoc::private_intra_doc_links)]` on by default
  = note: this link will resolve properly if you pass `--document-private-items`
```

Note that this has different behavior depending on whether you pass `--document-private-items` or not! If you document private items, then it will still generate a link, despite the warning:

```
warning: public documentation for `public` links to private item `private`
 --> priv.rs:1:6
  |
1 | /// [private]
  |      ^^^^^^^ this item is private
  |
  = note: `#[warn(rustdoc::private_intra_doc_links)]` on by default
  = note: this link resolves only because you passed `--document-private-items`, but will br
```

missing_docs

This lint is **allowed by default**. It detects items missing documentation. For example:

```
#![warn(missing_docs)]
```

```rust
pub fn undocumented() {}
# fn main() {}
```

The `undocumented` function will then have the following warning:

```
warning: missing documentation for a function
  --> your-crate/lib.rs:3:1
   |
 3 | pub fn undocumented() {}
   | ^^^^^^^^^^^^^^^^^^^^^^^^
```

Note that unlike other rustdoc lints, this lint is also available from `rustc` directly.

`missing_crate_level_docs`

This lint is **allowed by default**. It detects if there is no documentation at the crate root. For example:

```rust
#![warn(rustdoc::missing_crate_level_docs)]
```

This will generate the following warning:

```
warning: no documentation found for this crate's top-level module
  |
  = help: The following guide may be of use:
          https://doc.rust-lang.org/nightly/rustdoc/how-to-write-documentation.html
```

This is currently "allow" by default, but it is intended to make this a warning in the future. This is intended as a means to introduce new users on *how* to document their crate by pointing them to some instructions on how to get started, without providing overwhelming warnings like `missing_docs` might.

`missing_doc_code_examples`

This lint is **allowed by default** and is **nightly-only**. It detects when a documentation block is missing a code example. For example:

```rust
#![warn(rustdoc::missing_doc_code_examples)]

/// There is no code example!
pub fn no_code_example() {}
# fn main() {}
```

The `no_code_example` function will then have the following warning:

```
warning: Missing code example in this documentation
  --> your-crate/lib.rs:3:1
   |
LL | /// There is no code example!
   | ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

To fix the lint, you need to add a code example into the documentation block:

```
/// There is no code example!
///
/// ```
/// println!("calling no_code_example...");
/// no_code_example();
/// println!("we called no_code_example!");
/// ```
pub fn no_code_example() {}
```

private_doc_tests

This lint is **allowed by default**. It detects documentation tests when they are on a private item. For example:

```
#![warn(rustdoc::private_doc_tests)]

mod foo {
    /// private doc test
    ///
    /// ```
    /// assert!(false);
    /// ```
    fn bar() {}
}
# fn main() {}
```

Which will give:

```
warning: Documentation test in private item
  --> your-crate/lib.rs:4:1
   |
 4 | /     /// private doc test
 5 | |     ///
 6 | |     /// ```
 7 | |     /// assert!(false);
 8 | |     /// ```
   | |___________^
```

invalid_codeblock_attributes

This lint **warns by default**. It detects code block attributes in documentation examples that have potentially mis-typed values. For example:

```
#![warn(rustdoc::invalid_codeblock_attributes)]  // note: unnecessary - warns by default.

/// Example.
```

```rust
///
/// ```should-panic
/// assert_eq!(1, 2);
/// ```
pub fn foo() {}
```

Which will give:

```
warning: unknown attribute `should-panic`. Did you mean `should_panic`?
 --> src/lib.rs:1:1
  |
1 | / /// Example.
2 | | ///
3 | | /// ```should-panic
4 | | /// assert_eq!(1, 2);
5 | | /// ```
  | |_______^
  |
  = note: `#[warn(rustdoc::invalid_codeblock_attributes)]` on by default
  = help: the code block will either not be tested if not marked as a rust one or won't fail
```

In the example above, the correct form is `should_panic`. This helps detect typo mistakes for some common attributes.

invalid_html_tags

This lint is **allowed by default** and is **nightly-only**. It detects unclosed or invalid HTML tags. For example:

```rust
#![warn(rustdoc::invalid_html_tags)]

/// <h1>
/// </script>
pub fn foo() {}
```

Which will give:

```
warning: unopened HTML tag `script`
 --> foo.rs:1:1
  |
1 | / /// <h1>
2 | | /// </script>
  | |_____________^
  |
  note: the lint level is defined here
 --> foo.rs:1:9
  |
1 | #![warn(rustdoc::invalid_html_tags)]
  |         ^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
warning: unclosed HTML tag `h1`
 --> foo.rs:1:1
  |
1 | / /// <h1>
2 | | /// </script>
  | |_____________^

warning: 2 warnings emitted
```

invalid_rust_codeblocks

This lint **warns by default**. It detects Rust code blocks in documentation examples that are invalid (e.g. empty, not parsable as Rust). For example:

```
/// Empty code blocks (with and without the `rust` marker):
///
/// ```rust
/// ```
///
/// Invalid syntax in code blocks:
///
/// ```rust
/// '<
/// ```
pub fn foo() {}
```

Which will give:

```
warning: Rust code block is empty
 --> lint.rs:3:5
  |
3 |   /// ```rust
  |  _____^
4 | | /// ```
  | |_______^
  |
  = note: `#[warn(rustdoc::invalid_rust_codeblocks)]` on by default

warning: could not parse code block as Rust code
  --> lint.rs:8:5
   |
8  |   /// ```rust
   |  _____^
9  | | /// '<
10 | | /// ```
   | |_______^
```

```
   |
   = note: error from rustc: unterminated character literal
```

bare_urls

This lint is **warn-by-default**. It detects URLs which are not links. For example:

```
#![warn(rustdoc::bare_urls)] // note: unnecessary - warns by default.

/// http://example.org
/// [http://example.net]
pub fn foo() {}
```

Which will give:

```
warning: this URL is not a hyperlink
 --> links.rs:1:5
  |
1 | /// http://example.org
  |     ^^^^^^^^^^^^^^^^^^ help: use an automatic link instead: `<http://example.org>`
  |
  = note: `#[warn(rustdoc::bare_urls)]` on by default

warning: this URL is not a hyperlink
 --> links.rs:3:6
  |
3 | /// [http://example.net]
  |      ^^^^^^^^^^^^^^^^^^ help: use an automatic link instead: `<http://example.net>`

warning: 2 warnings emitted
```