★ Builder design pattern

"When construction gets a little bit too complicated"

⇒ Some objects are simple and can be created in a single constructor call.

⇒ Other objects require a lot of ceremony to create.

⇒ Having an object with 10 constructor arguments is not productive.

⇒ Instead, opt for piecewise construction.

⇒ Builder provides an API for constructing an object step-by-step.

Example

HtmlElement ← Class you want to build

HtmlElementBuilder ← Class used to build HtmlElement

⇒ Fluent interface {return class reference}

⇒ You can have static HtmlElementBuilder function inside the HtmlElement class that return instance of HtmlElementBuilder.

⇒ You can define operator HtmlElement() const for implicit conversion HtmlElementBuilder to HtmlElement.

⇒ You can make constructor of HtmlElement private to force use of HtmlElementBuilder.
↳ Make HtmlElementBuilder friend in HtmlElement.

⇒ Domain specific language [illegible] with C++.
↳ Using initializer list.

★ <u>Builder Facets</u>

"Facets is another design pattern"

⇒ More than one builder to work on an object.

——— ✕ ——— ✕ ———