U1NME8MS8 : How would you store colors in a library? rgb(1, 2, 3) or triples/quadruples etc.?
U4872964V : there is a `Color` type in Elm if that is sufficient
U4872964V : but if there weren't I'd probably make a triple/quadruple or a corresponding single constructor union type
U4872964V : or a record
U3SJEDR96 : It depends on what you need them for and what you plan to do with them, really
U1NME8MS8 : <@U3SJEDR96> personally I think the usecase would be css or SVG
U3SJEDR96 : There are quite a few libraries that deal with that already, though. <http://package.elm-lang.org/packages/elm-lang/core/5.1.1/Color> in core, <http://package.elm-lang.org/packages/eskimoblood/elm-color-extra/5.0.0/Color-Convert> for blending, transforming, converting, manipulating, <http://package.elm-lang.org/packages/mdgriffith/elm-color-mixing/latest> for mixing and whatnot, ...
U67HJ10TX : Hi, i'm tinkering with Elm and I wrote this simple opinion poll, just wondering if the "if..else..." in the Update function is idiomatic Elm```
module Main exposing (..)

import Html exposing (Html, div, fieldset, input, label, text)
import Html.Attributes exposing (name, style, type_)
import Html.Events exposing (onClick)


-- MODEL


type alias Model =
    { question : String
    , choiceOne : Int
    , choiceTwo : Int
    }


type Party
    = Jubilee
    | NASA
    | Wareva


type Msg
    = NoOp
    | Vote Party


initModel : Model
initModel =
    { question = ""
    , choiceOne = 0
    , choiceTwo = 0
    }



-- UPDATE


update : Msg -> Model -> Model
update msg model =
    case msg of
        NoOp ->
            model

        Vote party ->

```
        let
            partyName =
                toString party
        in
            if partyName == "Jubilee" then
                { model | choiceOne = model.choiceOne + 1 }
            else if partyName == "NASA" then
                { model | choiceTwo = model.choiceTwo + 1 }
            else
                model



-- VIEW


view : Model -&gt; Html Msg
view model =
    div []
        [ fieldset []
            [ radio "Jubilee" (Vote Jubilee)
            , radio "NASA" (Vote NASA)
            , radio "Wareva!!" (Vote Wareva)
            ]
        ]


radio : String -&gt; msg -&gt; Html msg
radio value msg =
    label
        [ style [ ( "padding", "20px" ) ] ]
        [ input [ type_ "radio", name "font-size", onClick msg ] []
        , text value
        ]


main =
    Html.beginnerProgram { model = initModel, view = view, update = update }
```

U1NME8MS8 : <@U3SJEDR96> I try to provide a package which integrates the colorbrewer colors
U153UK3FA : <@U67HJ10TX> you should use a `case..of` block for that
U1NME8MS8 : <@U3SJEDR96> I went with examples like ```set23 : List (Int, Int, Int)set23 = [(102, 194, 165), (252, 141, 98), (141, 160, 203)]``` now

U4872964V : <@U67HJ10TX> `if then else` or `case` are both fine here.
U4872964V : oh i missed that part, how about just `case` matching on the actual `Party` type?
U4872964V : that's the idiomatic Elm in this case, for sure
U663M2MB7 : Is there a way to make elm-format not ruining my comments? It treats every comment as standard ones from the elm architecture. I want my comments to sit on top of the functions I write, not with two new lines in between them.
U153UK3FA : <@U663M2MB7> the philosophy to elm-format is that elm-format formats your code how it wants and you get used to reading code formatted as elm-format formats it.
U663M2MB7 : I get that, but having comments three lines above the actual function? Surely I cannot be the only one who feels that is pretty awkward?
U153UK3FA : ah, that doesn't sound right.
U153UK3FA : can you give an example of the code?
U663M2MB7 : Well, it does it pretty much when ever I put in a comment, but sure
U3SJEDR96 : try using `{-` and `-}` to delimit your comments for functions
U3SJEDR96 : or `{-|` and `-}` to create doc-comments

U3SJEDR96 : `-- foo` at the top-level is interpreted as a "section" delimiter, or that's how I think of it anyway
U153UK3FA : <https://github.com/avh4/elm-format/blob/master/Style%20Guide/Sections.md>
U153UK3FA : <https://github.com/avh4/elm-format/blob/master/Style%20Guide/Declarations.md>
U663M2MB7 : <@U3SJEDR96> that has the same result though
U3SJEDR96 : ah, indeed, it needs to be a doc-comment
U663M2MB7 : <@U153UK3FA> guess you don't need that code example after all
U663M2MB7 : But okay, so `--` is strictly for sections then.
U663M2MB7 : Opening a comment with `{-|` is good :slightly_smiling_face:
U3SJEDR96 : note, also, that those comments are also used to generate documentation for packages
U663M2MB7 : I see. It stills feel kinda awkward though,  having two new lines inserted when ever you do simple `--` on a comment though. I guess I will just get used to it.
U4872964V : Just use the doc-comments, i suppose you are documenting your functions, right?
U2D07QZN3 : I need to use csv data.  I see elm-csv for parsing, but can someone point me to an example of getting the data from a server?
U4872964V : <@U2D07QZN3> is it the basics of getting data from a server you're after, or specifically how to treat it as CSV?
U4872964V : the parser expects a `String`, so `Http.getString` should do the job
U4872964V : <http://package.elm-lang.org/packages/elm-lang/http/1.0.0/Http#send>
U2D07QZN3 : Thanks, that should to it (I'm very new at this_)
U4TG1J2KA : hey, I'm trying to encode a Maybe Int to send to the back end. It's easy to decode a field that may be nullable, using something like Json.Decode.nullable or Json.Decode.oneOf, but I don't see any similar function on the encoding side. What's the best practice way of doing this?
U1UGYHGCA : ```maybeInt
|> Maybe.map <http://Json.Encode.int|Json.Encode.int>
|> Maybe.withDefault Json.Encode.null
```

U2SR9DL7Q : say you've defined a type `FunType = type FunType Int Int`. How would you extract just the `Ints` later? Or would you just do `type alias FunType = { firstInt : Int, secondInt: Int }` ?
U1UGYHGCA : Pattern matching
U1UGYHGCA : ```extractor: FunType -&amp;gt; (Int, Int)
extractor value =
  case value of
    FunType i1 i2 -&amp;gt; (i1, i2)
```

U1UGYHGCA : Or maybe even
U1UGYHGCA : ```extractor FunType -> (Int, Int)
extractor (FunType i1 i2) =
  (i1, i2)
```

U1UGYHGCA : Since your type can only have one value
U2SR9DL7Q : <@U1UGYHGCA> brilliant. Thanks. I'm thinking the type alias with the named fields might be better for this example then. Avoids a whole new function on my end.
U1UGYHGCA : That was just an example, you would do the pattern matching directly when using the type, you wouldn't need such function in real code
U1UGYHGCA : But yeah, depends on your need, the type alias might be more readable if you assign names to both int
```