roundcrisis.com

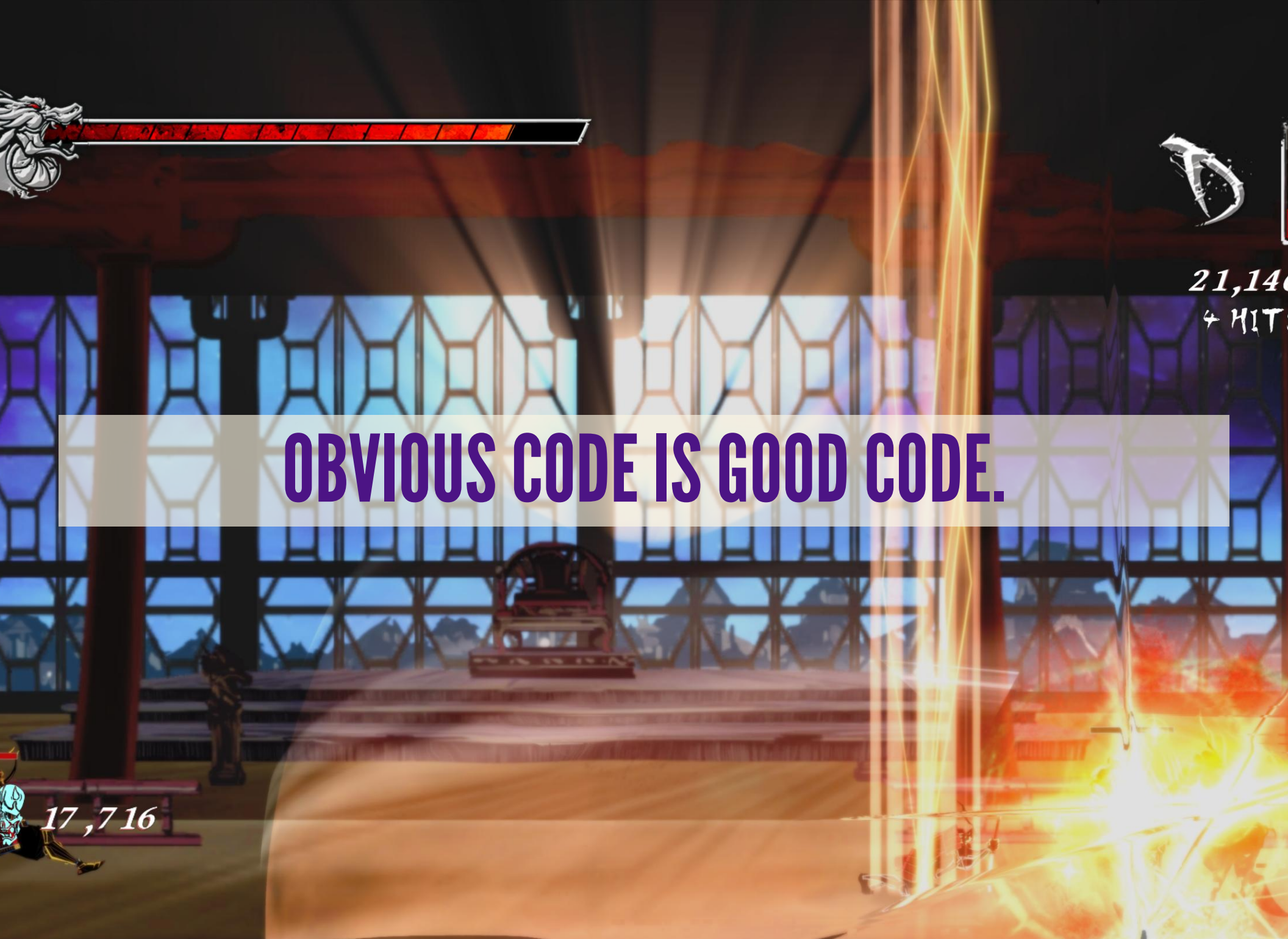
F# MENTORSHIP PROGRAME

FSHARP.ORG/MENTORSHIP

UNFRYING YOUR BRAIN WITH F#

QCON LONDON - MARCH 2016


KILLING DEMONS FOR FUN AND PROFIT
OniKira
悪魔キラー


OBVIOUS CODE IS GOOD CODE.

F# IS A GENERAL PURPOSE LANGUAGE.

INTEROP

```
1: type Order =
2:   | GoldOrder
3:   | PlatinumOrder of string
4:
5:   member this.OrderInfo =
6:     match this with
7:     | GoldOrder -> ""
8:     | PlatinumOrder(extraInfo ) -> "A foamy latte"
```

PATTERN MATCHING

```
1: let openFile (filePath) =
2:   match filePath with
3:   | path when
4:       Path.GetExtension(path) = ".txt" ||
5:       Path.GetExtension(path) = ".md" ->
6:                   openText path
7:   | path when
8:       Path.GetExtension(path) = ".jpg" ||
9:       Path.GetExtension(path) = ".png" ||
10:      Path.GetExtension(path) = ".gif" -> openText p
11:
12:  | _ -> "oh noes"
```

TOO MANY WHEN GUARDS

ACTIVE PATTERNS

```
1: let (|Extension|) (path: string) =
2:     Path.GetExtension <| path.ToLower()
3:
4: let openFile' path =
5:  match path with
6:   | Extension ".png"
7:   | Extension ".jpg"
8:   | Extension ".gif" -> openPictures path
9:   | Extension ".txt"
10:  | Extension ".md" -> openText path
11:  | _ -> "oh noes"
```

(| BANNANA OPERATOR |)

ACTIVE PATTERNS

- Use them outside of a match expression
- Pass parameters
- Nest them and combine them
- Should not be expensive or cause side effects.

Can you hangout? There's a cute guy here that I want you to meet.

No it's my cats birthday

What?

TYPE PROVIDERS

```
1: type GiphyTP = JsonProvider<"http://api.giphy.com/API
2: let query = ["api_key", key; "q", searchTerm]
3: let response = Http.RequestString (baseUrl,  query)
4:
5: let giphy = GiphyTP.Parse(response)
```

ASYNCHRONOUS WORKFLOWS

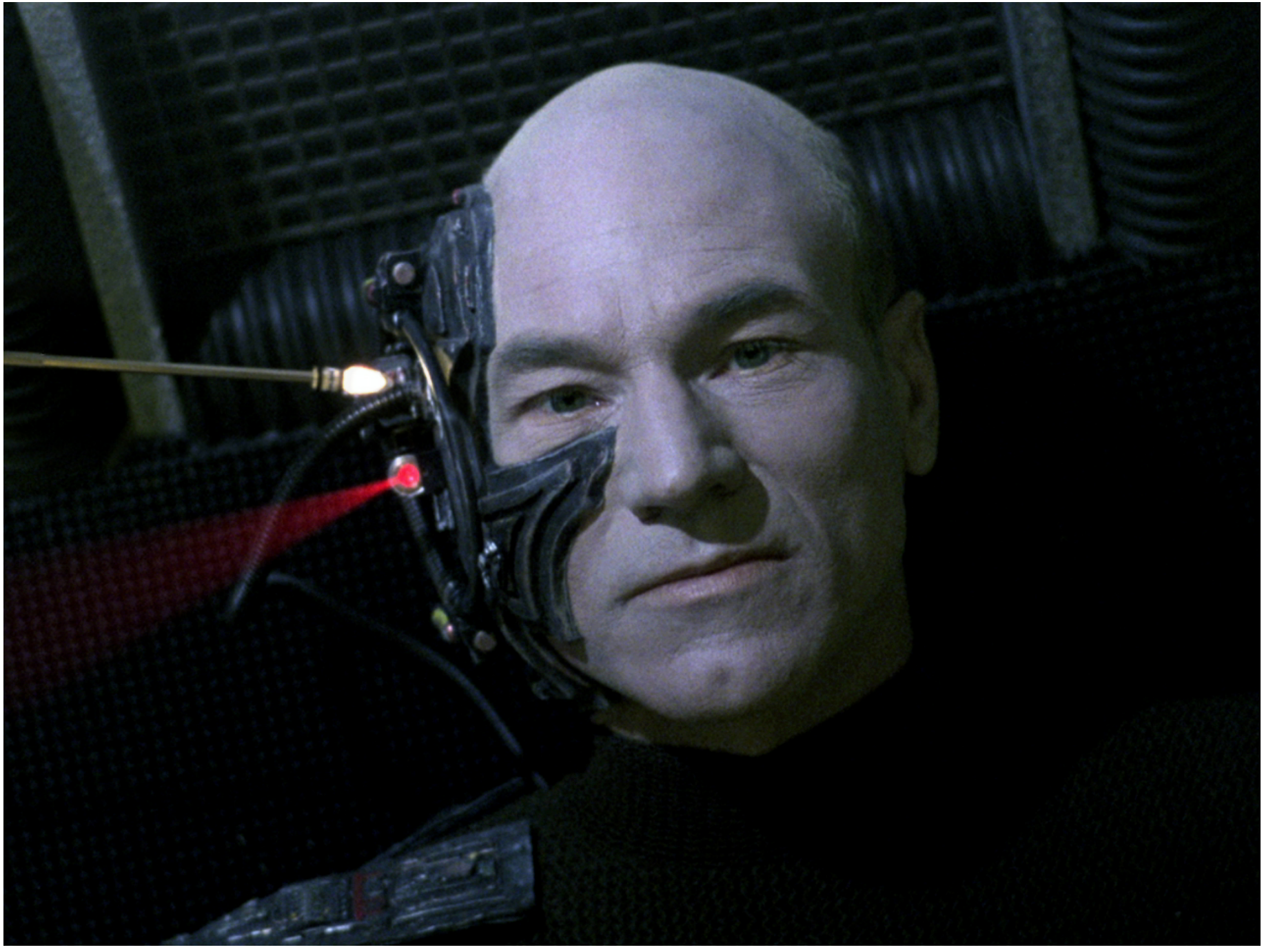
```
1: let getHtml(url:string) =
2:    let req = WebRequest.Create url
3:
4:    let response = req.GetResponse()
5:    use streatm = response.GetResponseStream()
6:    use reader = new StreamReader(streatm)
7:
8:    reader.ReadToEnd().Length
```

```
1: let getHtmlA(url:string) =
2:   async{
3:       let req = WebRequest.Create url
4:       let! response = req.AsyncGetResponse() // ding!
5:       use streatm = response.GetResponseStream()
6:       use reader = new StreamReader(streatm)
7:       return reader.ReadToEndAsync().Length // ding!
8:       }
```

```
1: sites
2: |> List.map (getHtmlAsync)
3: |> Async.Parallel
4: |> Async.RunSynchronously
```

COMPUTATION EXPRESSIONS

FAMILIAR



```
 1: let division a b c d =
 2:     match b with
 3:     | 0 -> None
 4:     | _ ->
 5:         match c with
 6:         | 0 -> None
 7:         | _ ->
 8:             match d with
 9:             | 0 -> None
10:             | _ -> Some(((a / b) / c) / d)
```

```
 1: let divide a b =
 2:     match b with
 3:     | 0 -> None
 4:     | _ -> Some(a / b)
 5:
 6: type MaybeBuilder() =
 7:     member __.Bind(value, func) =
 8:         match value with
 9:         | Some value -> func value
10:         | None -> None
11:
12:     member __.Return value = Some value
13:     member __.ReturnFrom value = __.Bind(value, __.R
```

COMPILER SERIVICES

SEPARATE HOW TO DEAL WITH DATA, FROM WHAT THE DATA DOES

ENJOY DYNAMIC LIKE FEATURES WITH TYPE SAFETY

EASE YOUR WAY INTO ASYNCHRONOUS CODE

WHEN YOU NEED TO DO SOMETHING DIFICULT SHOW THE RIGHT PATTERNS WITH FAMILIAR IDIOMS

MAKE EASY THINGS EASY, AND DIFICULT THINGS POSSIBLE

- @SilverSpoon
- roundcrisis.com

HTTPS://GITHUB.COM/ANDREA/UNFRYINGYOURBRAIN

EVENTS AND USER GROUPS

Functional Kats

- Functional Kats
- F#unctional Londoners meetup group
- Other user groups about programming languages that have no cats with capes on their logos :D

RESOURCES

- Extensible Pattern Matching Via a Lightweight Language Extension
- Active Patterns Series: Pattern Matching- Richard Dalton
- Interesting active patterns - Luke
- Using F# active patterns with Linq
- Denatured proteins rescued by trio of chaperones
- F# usage survey