# BlazeHtml

A blazingly fast Html generator in Haskell

# Hello

I'm Jasper. Together with Simon Meier, I started the BlazeHtml project.

@jaspervdj
jaspervdj.be

# Disclaimer

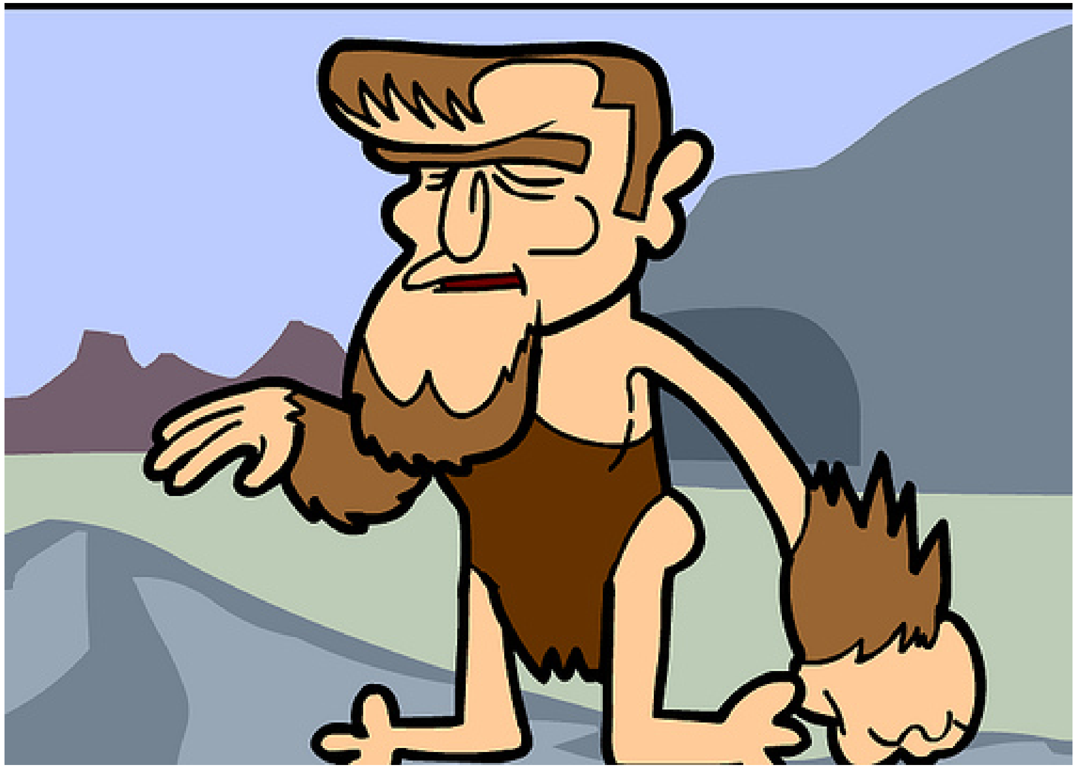
These are *very* opinionated slides. They express my own opinion, not the opinion of the BlazeHtml team.

# The Problem

A web server usually consists of three layers:

- A web application server
- A data storage layer
- An Html generation library *(oh look, that's us!)*

# how html formed



how is Html formed
how computer get page

# Mini-Languages

```
<html>
    <head>
        $title$
    </head>
    <body>
        $recentevents:event()$
    </body>
</html>
```

# Mini-Languages

# PHP, ERB, …

- Use of an embedded syntax.
- So designers can write templates, too.

# PHP, ERB, ...

Once the page gets sufficiently dynamic and thus complicated, designers will not be able to write it.

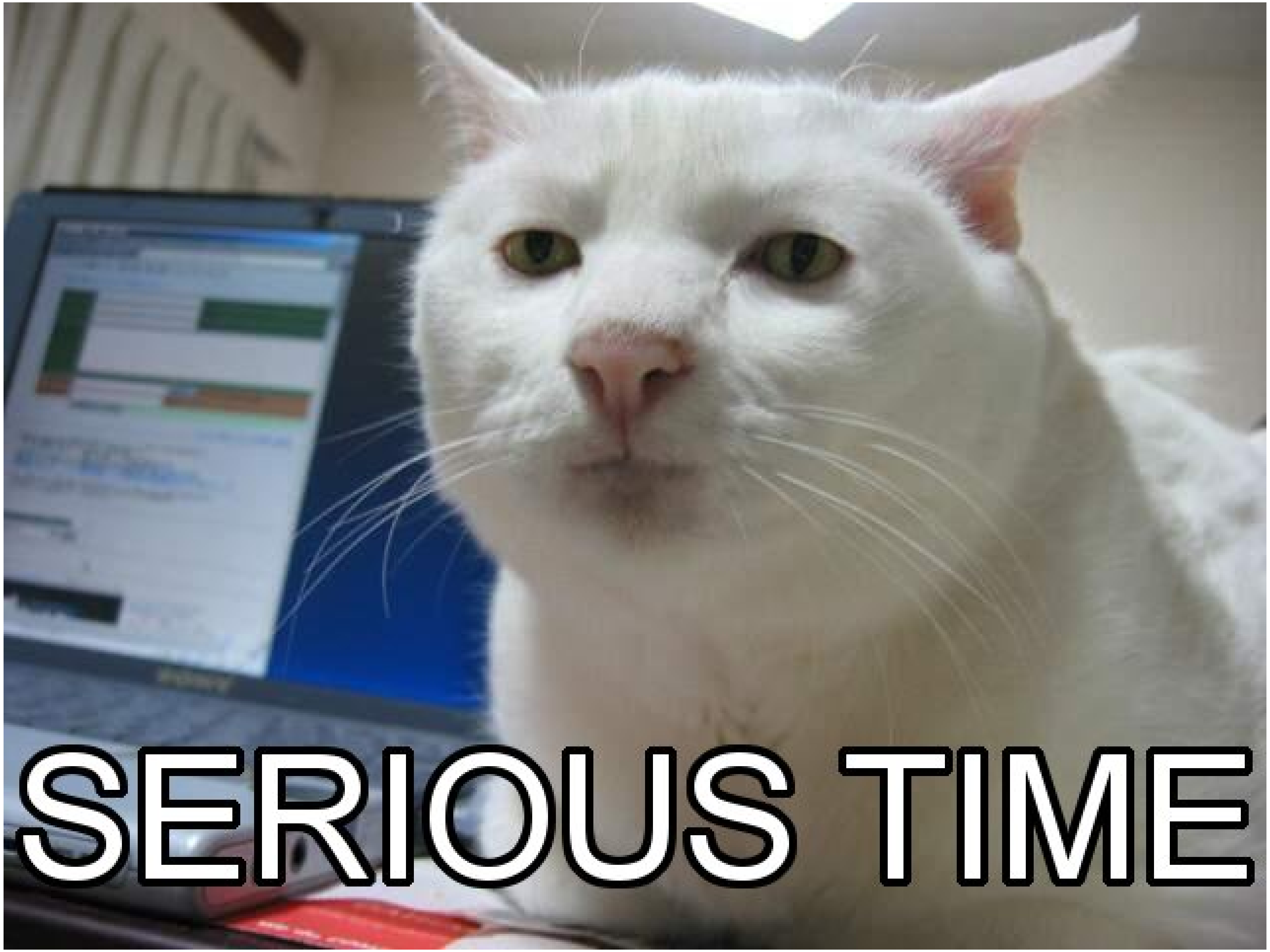
# PHP, ERB, ...

What happens at Google: designers fill templates with comments — and programmers fill in the dynamic stuff.

# combinators kthx

Yeah, well, you know, that's just, like, your opinion, man.

# Stop!



# We Want

In BlazeHtml we want:

- Efficiency.
- Composability.
- Extensibility.
- Clear syntax.

# Efficiency

- Aggressive inlining and rewrite rules.
- Use of `Data.Text` instead of `String`.
- We use the `Monoid` from `Data.Binary.Builder`.

# Composability

```
snippet :: Html h => h
```

- Use first class values.
- `Html` is a `Monoid`.
- Haskell gives us composability.

# Extensibility

`Html` is a typeclass.

- Fast instance will be provided.
- A pretty-printing instance will be provided.
- You can write your own! *(But you hopefully won't!)*

# Clear syntax

We kind of cheat and make `Html` a `Monad`.

- Unused parameter `a`.
- Unused `>>=`.
- ... but a clear do-notation syntax.

# Syntactic Sugar

# Syntactic Sugar

Use the `OverloadedStrings` pragma, so we can define custom string literals.

```
p "These & those."
```

# Stop!

Demo time!

# Questions?