The transparent web

Bridging the chasm in web development

“The best way to predict the future is to invent it.”

-Alan Kay

How did we get here?

- Shout-out to TBL!
- Web: 21 years and 12 days old!
- Was: Pages
- Nowadays: Applications...

What do I mean by “transparent web?”

- You know you are designing for two different platforms:
  - Server-side
  - Client-side
- “Transparent web” - *you shouldn’t have to!*

How?

- It’s early and techniques vary, but there are some themes:
  - [Purely] functional
  - Strongly typed
  - Automatic generation of client/server communication
  - Explicit syntax for DB/HTML (or a DSL)
  - Functional reactive code for UI

Why?

- Three main problems of webapps:
  - Security
  - Tons of sometimes-conflicting languages
  - You’re often also the sysadmin

Security

- Injection attacts:
  - Often a confusion of type: string vs. SQL vs. JavaScript
  - Fix with types:

```
<html>foo</html> != “<html>foo</html>”
```

Tons of languages

- Tower of Babel...
- Languages or markup needed by the working web developer (there are more + var. frameworks):
  - Markup (HTML), application code (Ruby), client-side (JavaScript), javascript abstraction layer (jQuery &etc.), style (CSS), DB (SQL)

Sysadminery

- Configuring hosts files
- Setting up:
  - RDBMSs
  - Web servers
  - Deployment
- This is all *vitally* important, but it’s not programming

It’s going to ruin my code

- It is common to have that feeling:

1. “That C code is going to produce crap compared to what I could write in assembly” - J. Random Hacker
2. “Garbage collection means that my program is going to be slow and crappy.” - J. Random Hacker, Jr.

It’s going to ruin my code

- But, history has chosen:

  1. Assembly vs. C (or other HLL) is settled:
     high-level languages.

  2. Manual vs. automatic memory management *perhaps* settled:
     automatic (garbage collection).

- Programmer productivity & the <u>elimination of certain kinds</u> of errors is usually a big win

It's going to ruin my code

- I have seen the future and it is:
  - Separate client/server apps & programmer-managed distribution vs. unified app & automatic distribution: unified & automatic

What's the alternative?

- For the first time, some are starting to show up!
- Opa
- ~~Ur/Web~~
- ~~Meteor.js~~ ran out of time :(

Not really JS-related, skipping

So, Opa...

- I don’t know what counts as a JS framework, but...
- Opa is a language (with libraries!) that *compiles* to JS, more like *coffeescript*
- See also: List of languages that compile to JS · jashkenas/coffee-script Wiki · GitHub (Opa falls under “Tierless languages”)

Example: hello world

- Let’s do “Hello world!”... <sigh>
- But it gets you past the “how the !@#&^$ do I even *compile* this?” phase (or at least for me)
- And it fits on a slide :)

Opa: Hello world!

helloworld.opa

```
function main() {
    <h1>Hello, world!</h1>
}

Server.start(Server.http,
             {title: "Hello, world!", page: main})
```

cough node.js *cough*

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
```

Opa: Hello world!

...compile, run:

```
$ opa helloworld.opa
$ ./helloworld.js
```

http://localhost:8080

Opa: Hello world!

- main returns an xhtml fragment, the compiler knows that this is XHTML and not a string
  - notice lack of: “ “
- the server then sets us up with a page

Example: comments

- Let’s <u>storycard</u> this: I’d like to be able to...
  - Fill in name, comment, email, click “comment” and
  - Show past comments (implies persistent storage)

Opa: Comments

- Source...

Opa: Comments

- To compile and run:

```
opa comments.opa
./comments.js
[...creates mongodb datastore...]
```

Example: real-time chat

- Start a project:

```
$ opa create chat_opa
```

Real-time chat

- login is a work in progress...
- run:
  `./chat_opa.exe`
- browse:
  `http://localhost:8080/chat`

Thanks!

- I'm Chris Wilson:
  - chris@bendyworks.com
  - @twopoint718
- Normally, I hack Rails for Bendyworks
  - ...and thanks to them for letting me use some *20% time* for this :)

Questions

- I’ll do my best :)

Resources

1. http://opalang.org/
   1. http://www.mongodb.org/
   2. http://nodejs.org/