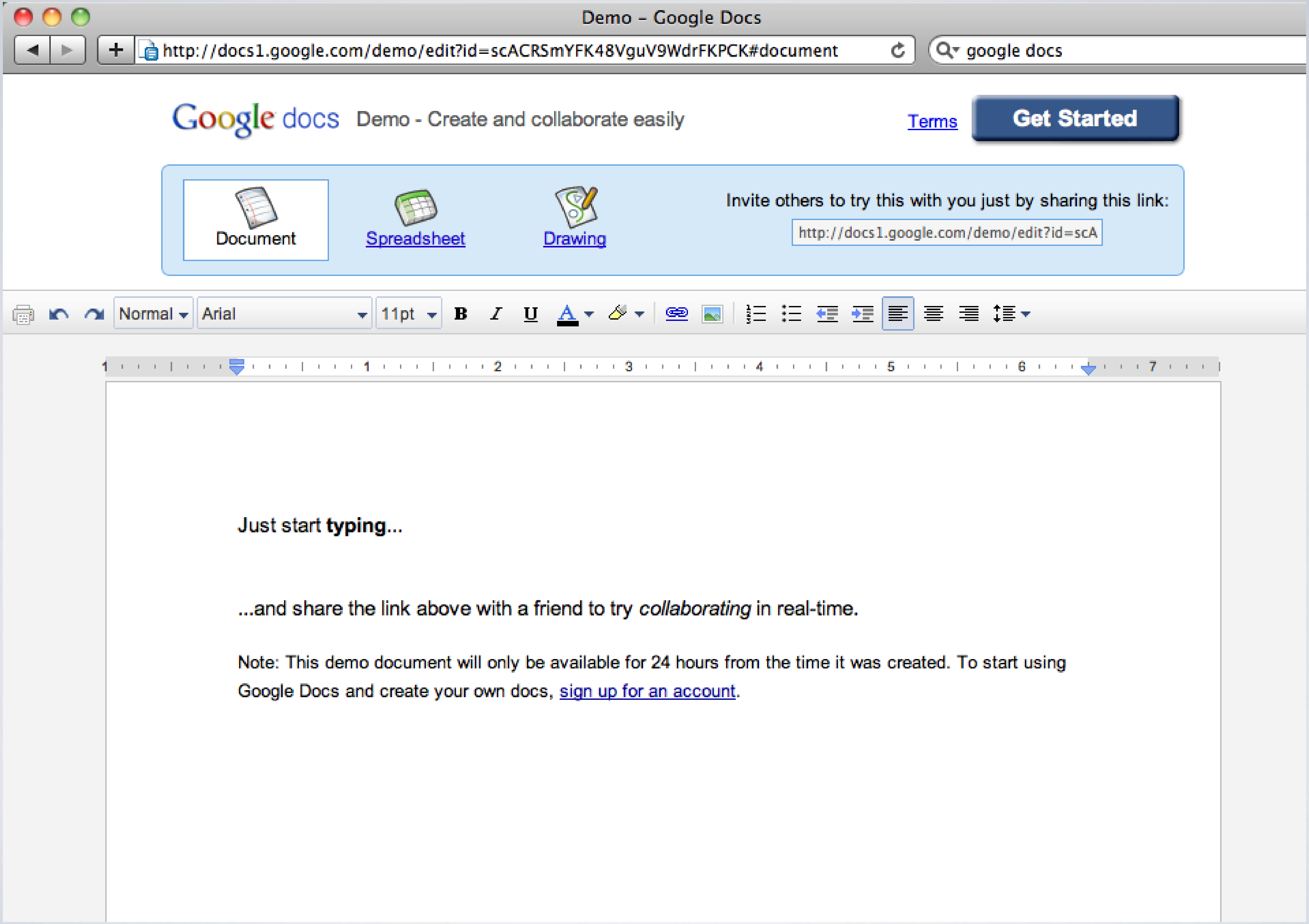
QWeSST

Type-Safe Web Programming

Thierry Sans and Iliano Cervesato

Carnegie Mellon University Qatar


Demo – Google Docs
http://docs1.google.com/demo/edit?id=scACRSmYFK48VguV9WdrFKPCK#document
google docs
Google docs Demo - Create and collaborate easily
Terms
Get Started
Document
Spreadsheet
Drawing
Invite others to try this with you just by sharing this link:
http://docs1.google.com/demo/edit?id=scA
Normal
Arial
11pt
Just start typing...
...and share the link above with a friend to try collaborating in real-time.
Note: This demo document will only be available for 24 hours from the time it was created. To start using Google Docs and create your own docs, sign up for an account.

- **Project Goal**
  - ➡ Study the foundations of web programming
- **Outcomes**
  - ➡ QWeSST: a type-safe programming language for the web
  - ➡ Faithful semantics description for parallel languages
  - ➡ QWeSST$^{\varphi}$: managing distributed flow of data on the web

Web Programming

Anatomy of a Web Application

JavaScript

HTML
PHP
Java
ASP/.Net
Ruby
Python
Server JS

- Mobile code
- Remote execution
- State
- Security

Limitation of current web technologies

- ➡ Use of **heterogeneous languages** (not originally designed with distributed computing in mind)
- ➡ Require **heavy testing**
- ◉ Setting up the communication machinery is expensive and error prone

Partial solution – Better libraries

- Simplifying the communication machinery

➡ Abstract libraries (such as JQuery and Prototype)

- But we still have to care about requests and callbacks

```
<script type="text/javascript" src="prototype.js"></script>
<script language="javascript" type="text/javascript">
function sendRequest(arg)
{
    // Do something with the Ajax response
    function doSomething(result)
    {
        $('resultDiv').update(result.responseText);
    }

    // send the Ajax request
    new Ajax.Request('helloService.php',
                     { method: 'post',
                       parameters: arg,
                       onComplete: doSomething});
}
</script>
```

Partial solution – **One** language

Write an entire webapp in the same language

➡ Google Web Toolkit, LINKS, HOP

- Programmer designates code as client or server
- Compiled to JavaScript or Java

➡ Flash, Silverlight

- Interpreted in the browser

Complexity is rising

- Webapps are getting more and more **sophisticated** and **distributed**

- Current technologies are unlikely to be able to support this **growing complexity**

QWeSST

A Type-Safe Programming Language for the Web

Looking for foundations of web programming

- A language to carry out local computations

✓ A λ-calculus

- Constructs to **publish** code and **call** it through a URL

✓ Remote procedure mechanism

- Constructs to **suspend** and **resume** a computation

✓ Mobile code

in a well-typed fashion

Remote Procedures

Types $\tau \ ::= \ldots \mid \tau \hookrightarrow \tau'$

Expressions $e \ ::= \ldots \mid \mathrm{w}/u \mid \text{publish } x{:}\tau.\ e \mid \text{call } e_1 \text{ with } e_2$

- Browser to web server

  - **Web pages**
  - **Ajax**

```
function factorial(n){
    new Ajax.Request(www.example.com/factorial.php,{
            method: 'get',
            parameters: "arg="+n,
            onSuccess: function(response){alert(response.responseText)}});}
```

- Web server to web server
  - **XML/RPC** (web service)

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/en
   <soapenv:Header/>
   <soapenv:Body>
      <getFactorial>
         <param>6</param>
      </getFactorial>
   </soapenv:Body>
</soapenv:Envelope>
```

publish / call

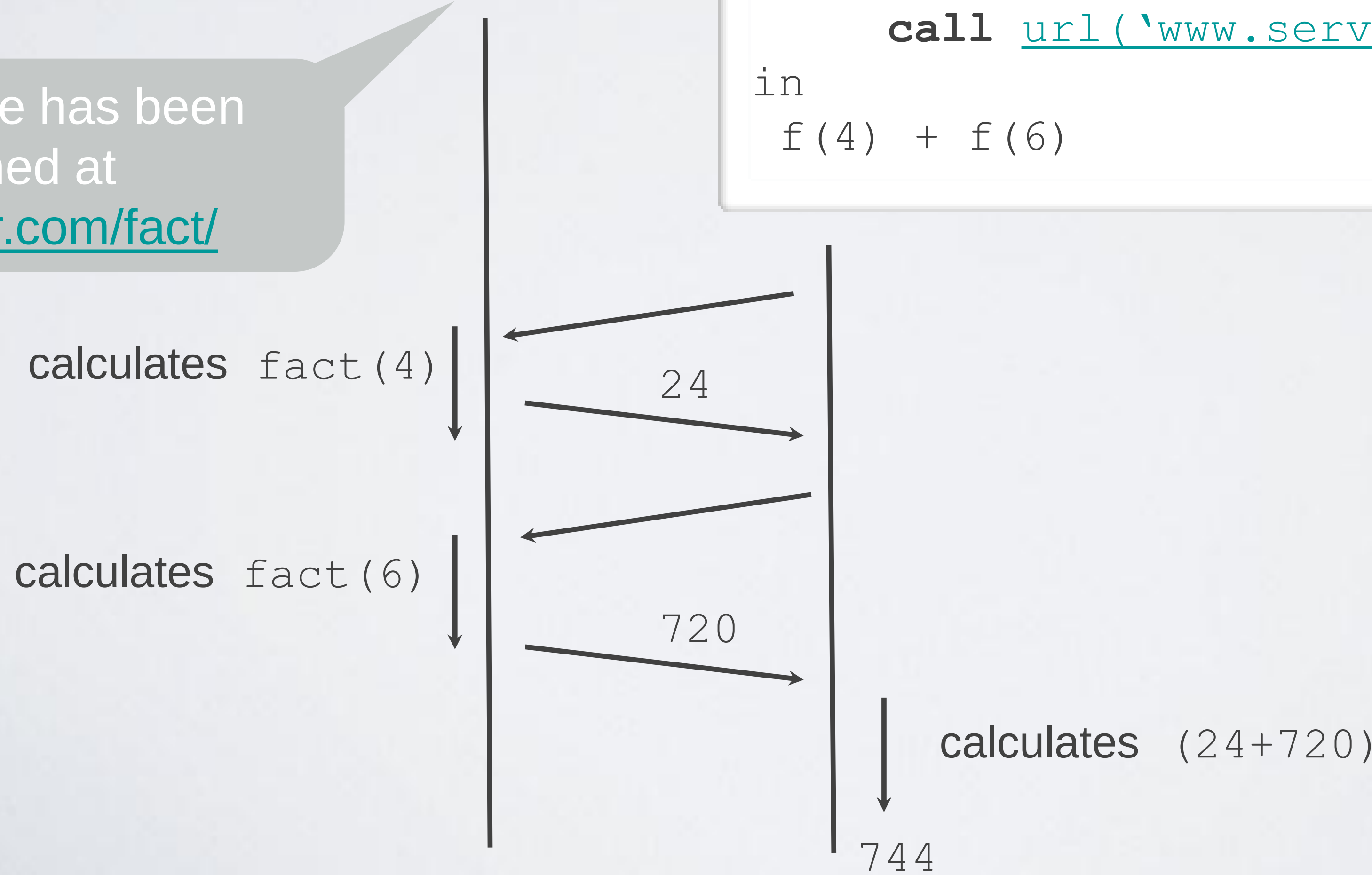
Server

```
let
fun fact(n) => if = 0 then 1
                else n * fact(n-1)
in
publish x => fact(x)
```

A new service has been published at www.server.com/fact/

```
let
 fun f(x) =>
     call url('www.server.com/fact/') with x
in
 f(4) + f(6)
```

Client



Mobile Code

Types τ ::= ... | susp[τ]

Expressions e ::= ... | hold e | resume e

- Web server to browser
  - **Javascript** code

```
<script type="text/javascript" src="includes/prototype.js"></script>

<script language="javascript" type="text/javascript">

function logout()
{
    new Ajax.Request('clearSession', {onComplete: function (obj){locat
```

- Web server to web server
  - Not done in practice

hold / resume

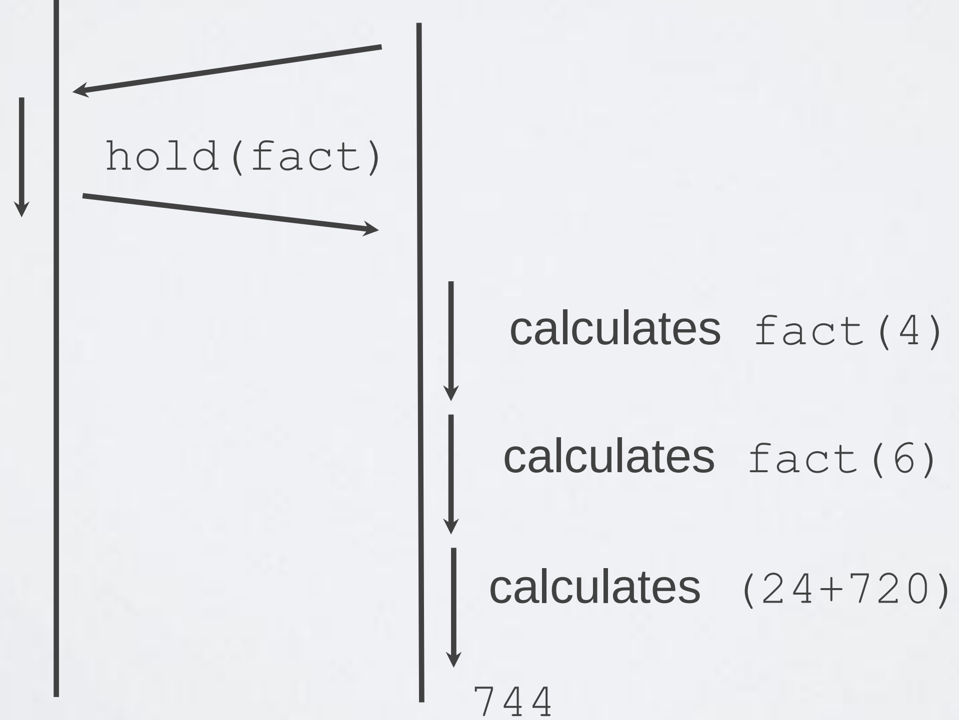
Server

```
let
fun fact(n) => if n= 0 then 1
                 else n * fact(n-1)
in
   publish x => hold(fact)
```

A new service has been published at www.server.com/fact/

```
let
 f = resume (call url('www.server.com/fact/') with ())
in
 f(4) + f(6)
```

Client



Web pages vs. Web services

✓ Web pages and web services are treated uniformly

➡ It is all about calling a URL (with some parameters) and getting a result back

➡ The difference is how the result is used


```
function factorial(n){
    new Ajax.Request(www.example.com/factorial.php,{
            method: 'get',
            parameters: "arg="+n,
            onSuccess: function(response){alert(response.responseText)}});}
```

QWeSST - A language for web programming

- A simple abstraction of the way we program the web

✓ Easier to reason about complex web programs

- Currently a **pure** language (no effects)
- **Static** and **localized** type semantics
  - **Localized** type checking

✓ **Globally type safe** language

More examples

- Custom Web Service
- Web API
- Custom Web API
- Web service auto-installer

➡ Check the Qwesst website:
http://tsans-mac.qatar.win.cmu.edu/

An API

Server

```
let
search = url('www.server.com/search/')
script = hold (fn x => call search with x)
in
publish x => script
```

Client

```
let
api = url('www.server.com/api/')
s = resume (call api with ())
in
s('myRequest')
```

A new service has been published at www.server.com/api/

A Web Service Auto-installer

Server

```
let
search = url('www.server.com/search/')
f = (fn x => call search with x)
script = hold (publish x => f(x))
in
publish x => script
```

Client

```
let
installer = url('www.server.com/inst/')
in
resume (call installer with ())
```

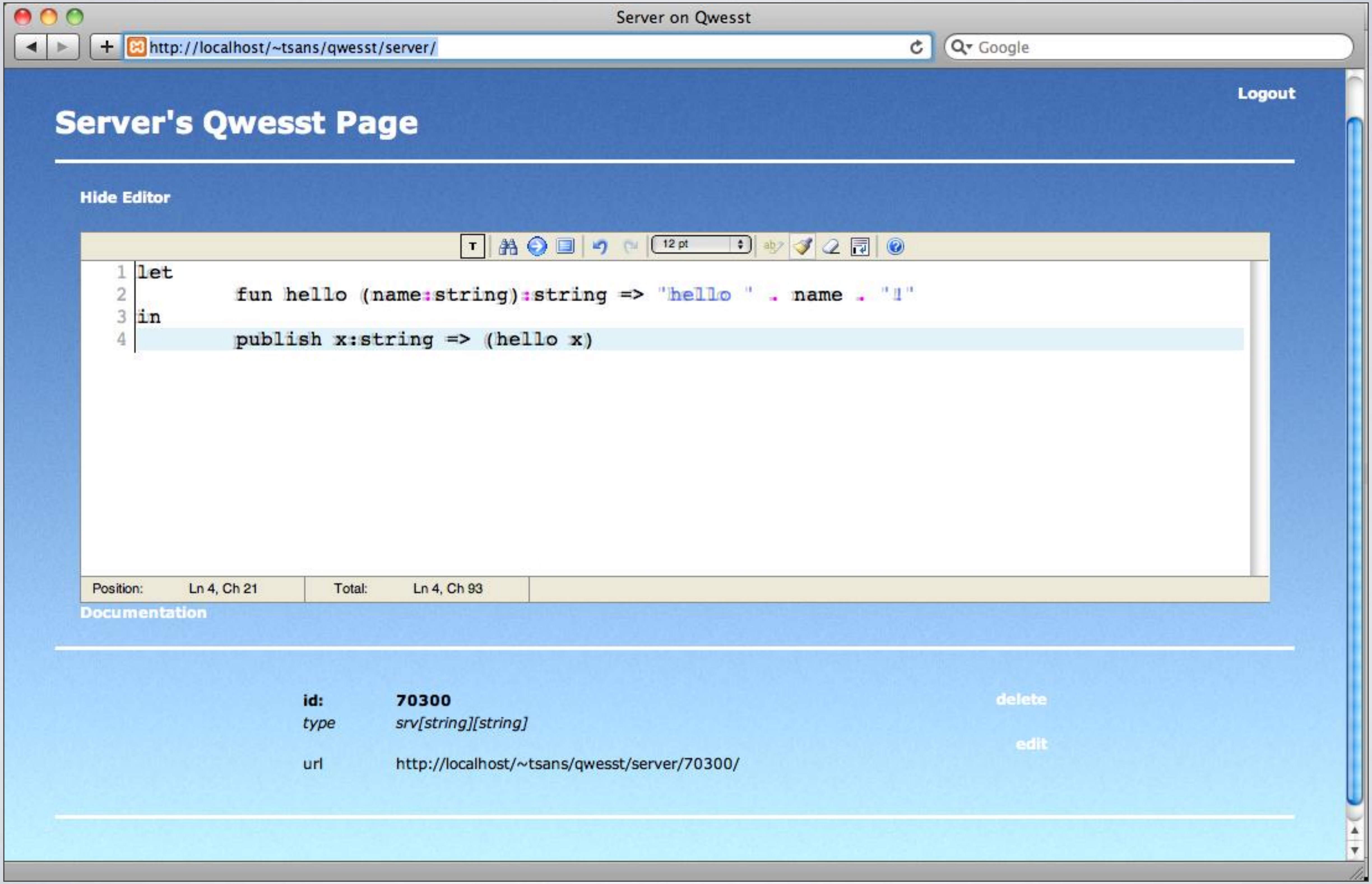
Customer

```
let
f = url('www.client.com/search/')
in
call f with 'myQuery'
```

A new service has been published at www.server.com/inst/

A new service has been published at www.client.com/search/

Demo



Server on Qwesst
http://localhost/~tsans/qwesst/server/
Google
Logout
Server's Qwesst Page
Hide Editor
12 pt
1 let
2 fun hello (name:string):string => "hello " . name . "!"
3 in
4 publish x:string => (hello x)
Position: Ln 4, Ch 21
Total: Ln 4, Ch 93
Documentation
id: 70300
type srv[string][string]
url http://localhost/~tsans/qwesst/server/70300/
delete
edit

QWeSST

Formal Semantics

Typing

Internet Expression Type

$\Sigma; \Gamma \vdash_{w} e : \tau$ (e has type τ at w in Σ and Γ)

Local context Location

- Inspired to ***ML5's type system for localized computation*** by Tom Murphy VII, Karl Crary and Robert Harper

Typing Semantics

Remote Procedure Call

$$\frac{\tau \hookrightarrow \tau' \text{ mobile}}{\Sigma, \mathrm{w}'/u{:}\ \tau \hookrightarrow \tau';\ \Gamma \mid\!-_{\mathrm{w}} \mathrm{w}'/u : \tau \hookrightarrow \tau'}$$

$$\frac{\tau \hookrightarrow \tau' \text{ mobile} \qquad \Sigma;\ \Gamma, x : \tau \mid\!-_{\mathrm{w}} e : \tau'}{\Sigma;\ \Gamma \mid\!-_{\mathrm{w}} \text{publish}\ \ x{:}\tau.\ e : \tau \hookrightarrow \tau'}$$

$$\frac{\Sigma;\ \Gamma \mid\!-_{\mathrm{w}} e_1 : \tau \hookrightarrow \tau' \qquad \Sigma;\ \Gamma \mid\!-_{\mathrm{w}} e_2 : \tau}{\Sigma;\ \Gamma \mid\!-_{\mathrm{w}} \text{call}\ e_1\ \text{with}\ e_2 : \tau'}$$

Mobile Code

$$\frac{\Sigma;\ \Gamma \mid\!-_{\mathrm{w}} e : \tau}{\Sigma;\ \Gamma \mid\!-_{\mathrm{w}} \text{hold}\ e : \text{susp}[\tau]}$$

$$\frac{\Sigma;\ \Gamma \mid\!-_{\mathrm{w}} e : \text{susp}[\tau]}{\Sigma;\ \Gamma \mid\!-_{\mathrm{w}} \text{resume}\ e : \tau}$$

Evaluation

Internet

Expression

$\Delta; e \Rightarrow_w \Delta'; e'$

Location

(Δ; e steps to Δ'; e')

Evaluation Semantics

Remote Procedure Call

$$\frac{}{\Delta;\ \text{publish}\ \ x{:}\tau.\ e \ \Rightarrow_{\mathrm{w}}\ (\Delta,\ \mathrm{w}/u = x{:}\tau.\ e);\ \mathrm{w}/u}$$

$$\frac{v_2\ \text{val}}{\underbrace{(\Delta',\ \mathrm{w'}/u = x{:}\tau.\ e)}_{\Delta};\ \text{call}\ \mathrm{w'}/u\ \text{with}\ v_2 \ \Rightarrow_{\mathrm{w}}\ \Delta;\ \text{expect}\ [v_2\,/x]\ e\ \text{from}\ \mathrm{w'}}$$

$$\frac{\Delta;\ e \ \Rightarrow_{\mathrm{w'}}\ \Delta';\ e'}{\Delta;\ \text{expect}\ e\ \text{from}\ \mathrm{w'} \ \Rightarrow_{\mathrm{w}}\ \Delta';\ \text{expect}\ e'\ \text{from}\ \mathrm{w'}}$$

$$\frac{v\ \text{val}}{\Delta;\ \text{expect}\ v\ \text{from}\ \mathrm{w'} \ \Rightarrow_{\mathrm{w}}\ \Delta;\ v}$$

Mobile Code

$$\frac{\Delta;\ e \ \Rightarrow_{\mathrm{w}}\ \Delta';\ e'}{\Delta;\ \text{resume}\ e \ \Rightarrow_{\mathrm{w}}\ \Delta';\ \text{resume}\ e'}$$

$$\frac{}{\Delta;\ \text{resume}\ (\text{hold}\ e) \ \Rightarrow_{\mathrm{w}}\ \Delta;\ e}$$

Meta-theory

✓ QWeSST is **type safe** (proof verified using Twelf)

➡ Type preservation

If $\Sigma; . \vdash_w e : \tau$ and $\Sigma \vdash \Delta$ and $\Delta; e \Rightarrow_w \Delta'; e'$,
then $\Sigma'; . \vdash_w e' : \tau$ and $\Sigma' \vdash \Delta'$

➡ Progress

If $\Sigma; . \vdash_w e : \tau$ and $\Sigma \vdash \Delta$, then

- either e val
- or $\Delta; e \Rightarrow_w \Delta'; e'$

Parallel Semantics

A Semantic Mismatch

$$\Delta; e \Rightarrow_{\mathrm{w}} \Delta'; e'$$

- One expression at a time is evaluating
  - Single-threaded
- This is not the way the web works
  - Millions of executions occurring simultaneously
  - Possibly on the same node

Serialized semantics

- Parallelism reduced to non-deterministic interleaving

- Macro-step as series of micro-steps

$$\frac{}{\Delta;\ .\ \Rightarrow\ \Delta;\ .} \qquad \frac{\Delta;\ e\ {}_{?}\!\!\Rightarrow_{\mathrm{w}}\ (\Delta,\Delta');\ e' \qquad \Delta;\ E\ \Rightarrow\ (\Delta,\Delta'');\ E'}{\Delta;\ (e\ @\ \mathrm{w},\ E)\ \Rightarrow\ (\Delta,\Delta',\Delta'');\ (e'\ @\ \mathrm{w},\ E')}$$

  - Serialized typing semantics

$$\frac{}{\Sigma\ |\text{-}\ .\ :\ .} \qquad \frac{\Sigma;\ .\ |\text{-}_{\mathrm{w}}\ e : \tau \qquad \Sigma\ |\text{-}\ E : \mathrm{T}}{\Sigma\ |\text{-}\ (e\ @\ \mathrm{w},\ E) : \tau\ ,\ \mathrm{T}}$$

  - Serialized safety proof if working with sequences
  - Large overhead if working with multisets

Multiset-Oriented Rules

- Rules can talk about multisets
- Rules can have multisets of premises
- Specified by **parametric multiset comprehension**

$$\frac{\{\ e_i \ \mathrm{val}\ \}}{\{e_i \ @\ \mathrm{w}_i\}\ \mathrm{final}}\ (i \in I)$$

Linear Destination Passing Style

- "Branching" stack machine with explicit return addresses
  - $(e)^d$ – evaluate e for d
  - $(v)_d$ – return v to d
  - $(\text{call } d_1 \text{ with } d_2)^d$ – wait for results

$$\frac{}{(\text{hold } e)^d \Rightarrow_w (\text{hold } e)_d} \qquad \frac{}{(\text{resume } e)^d \Rightarrow_w (\text{resume } d')^d, (e)^{d'}}$$

$$\frac{}{(\text{resume } d')^d, (\text{hold } e)_{d'} \Rightarrow_w (e)^d}$$

LDP rules for `call`

$$\frac{}{(\text{call } e_1 \text{ with } e_2)^d \Rightarrow_{\text{w}} (\text{call } d' \text{ with } d'')^d, (e_1)^{d'}, (e_2)^{d''}}$$

$$\frac{\text{w}'/u = x{:}\tau.\, e \in \Delta}{\begin{array}{l}(\text{call } d' \text{ with } d'')^d, (\text{w}'/u)_{d'}, (v)_{d''} \Rightarrow_{\text{w}} (\text{expect } d''' \text{ from w}')^d \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot \Rightarrow_{\text{w}'} ([v/x]e)^{d'''}\end{array}}$$

$$\frac{v' \text{ val}}{\begin{array}{l}(\text{expect } d''' \text{ from w}')^d \Rightarrow_{\text{w}} (v')_d \\ \qquad\qquad (v')^{d'''} \Rightarrow_{\text{w}'} \cdot\end{array}}$$

Orchestration

Simplified for typesetting reasons

- Evaluation

$$\frac{\{\ \Delta;\ e_i \Rightarrow_{w_i} (\Delta, \Delta_i);\ e_i'\ \}}{\Delta; \{e_i\ @\ w_i\}, E \Rightarrow (\Delta, \{\Delta_i\}); \{e_i'@\ w_i\}, E}\ (i \in I)$$

- Typing

$$\frac{\{\ \Sigma;\ d_i{:}\tau_i \mid\!-_{w_i} e_i\ \} \qquad \Sigma \mid\!- \Delta}{\Sigma; \{d_i : \tau_i\} \mid\!- \Delta; \{e_i\ @\ w_i\}}\ (i \in I)$$

Substructural meta-theory

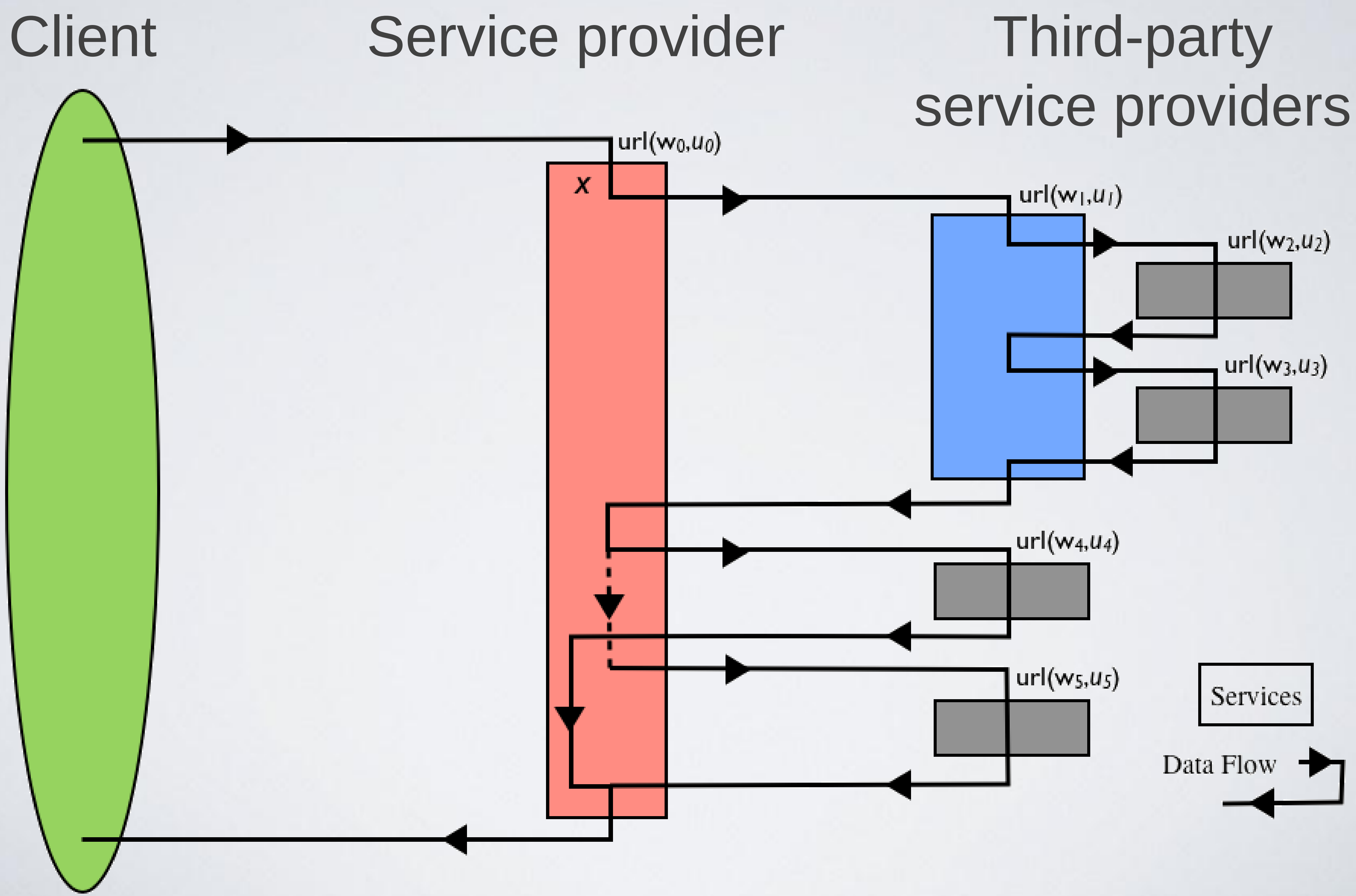
	Type Preservation	**Progress**
Local	If $\Sigma; d{:}\tau \vdash_w e$ and $\Sigma \vdash \Delta$ and $\Delta; e \Rightarrow_w \Delta'; e'$, then $\Sigma'; d{:}\tau \vdash_w e'$ and $\Sigma' \vdash \Delta'$	If $\Sigma; d{:}\tau \vdash_w e$ and $\Sigma \vdash \Delta$, then • either e val • or $\Delta; e \Rightarrow_w \Delta'; e'$
Global	If $\Sigma; \Lambda \vdash \Delta; E$ and $\Delta; E \Rightarrow \Delta'; E'$, then $\Sigma'; \Lambda \vdash_w \Delta'; e'$	If $\Sigma; \Lambda \vdash \Delta; E$, then • either E final • or $\Delta; E \Rightarrow \Delta'; E'$

Managing Data Flow on the Web

Services [use other services]*

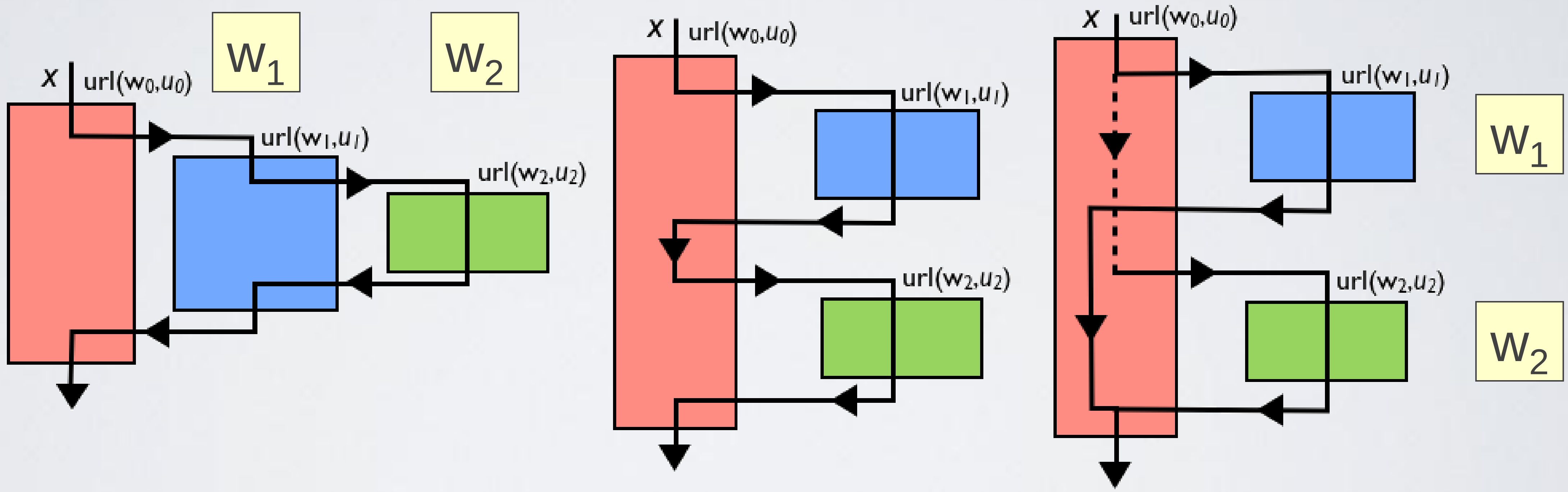
- How does a service provider describe data paths through the web?
- How can a client control where her data goes?

Scenario

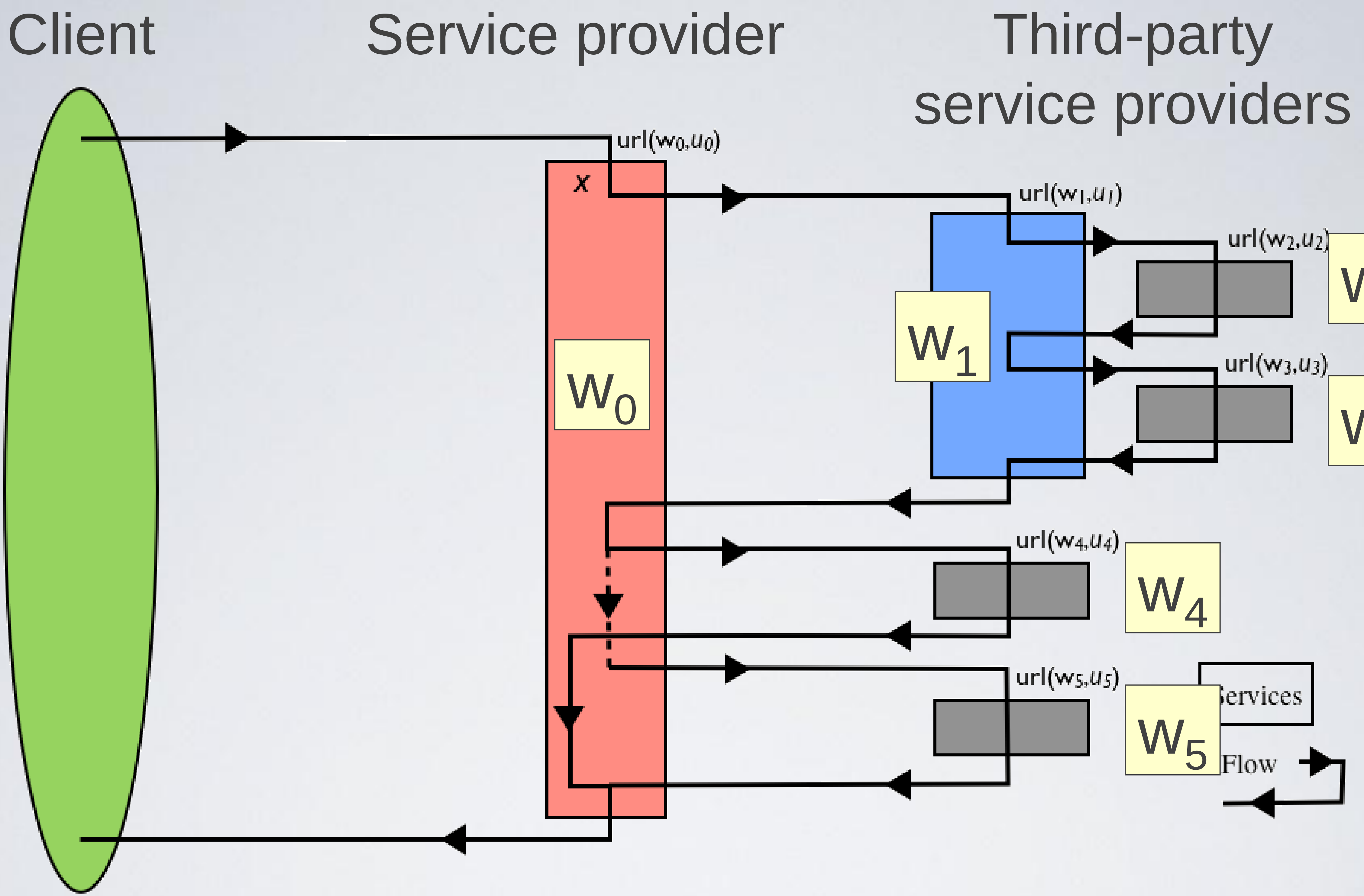

Client
Service provider
Third-party
service providers
url(w_0,u_0)
x
url(w_1,u_1)
url(w_2,u_2)
url(w_3,u_3)
url(w_4,u_4)
url(w_5,u_5)
Services
Data Flow

Describing data paths

$$\mu \ ::= \ \bullet \ \mid \ w; \mu \ \mid \ \mu \text{ o } \mu' \ \mid \ \mu \parallel \mu'$$



$w_1; w_2$ $\qquad$ $w_1 \text{ o } w_2$ $\qquad$ $w_1 \parallel w_2$



w_0 ;(w_1 ; (w_2 o w_3)) o (w_4 || w_5)

Describing flow policies

$$
\begin{aligned}
\rho \ ::= \ & T \mid F \mid \neg\rho \mid \rho \wedge \rho' \mid \rho \vee \rho' \\
& \mid \bullet \mid w;\rho \mid \rho \circ \rho' \\
& \mid \{w_i\}^*;\rho \mid \{w_i\}?;\rho \\
& \mid (\rho)^* \circ \rho' \mid (\rho)? \circ \rho'
\end{aligned}
$$

➡ Can describe

✓ Basic permissions and prohibitions

✓ Strict sequencing (e.g., anonymization policies)

✓ Flow isolation (a la Chinese wall policy)

Incorporating paths and policies into Qwesst

- Data paths in local and remote function types

➡ τ ::= ... | $\tau[\mu] \rightarrow \tau'$ | $\tau[\mu] \hookrightarrow^{w} \tau'$

✓ Type annotations are inferred

- Policies in call

➡ call e_1 with e_2 [ρ]

Incorporating paths and policies into Qwesst

- Flow inference and control in type checking

$$\frac{\Sigma; \Gamma \vdash_{w} e_1 : \tau[\mu] \hookrightarrow^{w'} \tau' \qquad \Sigma; \Gamma' \vdash_{w} e_2 : \tau \qquad \mu \models \rho}{\Sigma; (\Gamma \parallel (\Gamma' \circ (w'; \mu))) \vdash_{w} \text{call } e_1 \text{ with } e_2 [\rho] : \tau'}$$

- Evaluation remains unchanged

Meta-theory

- The language remains type safe

Perspectives and Future Work

Short Term

- More expressive constructs and data structures
- Features for “real” web development
  - Browser embedded interpreter
  - DOM implementation

✓ We want to build a higher level language that relies on Javascript and markup languages

Longer Term

- More security
- Effects & concurrency
- A way to track and manage dead links
- A logical framework based on multiset comprehension

Thank You

Any Qwesstion?