Concolic Testing of Full-Stack JavaScript Web Applications

(Tearless Project Deliverable 1.3.2)

Maarten Vandercammen, Laurent Christophe, Coen De Roover

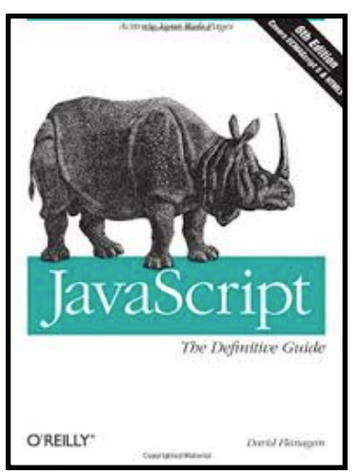
Maarten.Vandercammen@vub.be

VRIJE UNIVERSITEIT BRUSSEL

Automated Testing of Web Sites

StackFul automatically tests web sites and reports bugs

```
1 var shoppingCart = undefined;
2
3 document.getElementById("JS_book").addEventListener("click", function() {
4   shoppingCart = { name: "JavaScript_textbook", price: 19.99 };
5 });
6 document.getElementById("buy_button").addEventListener("click", function() {
7   var price = shoppingCart.price;
8     .....
9 });
```

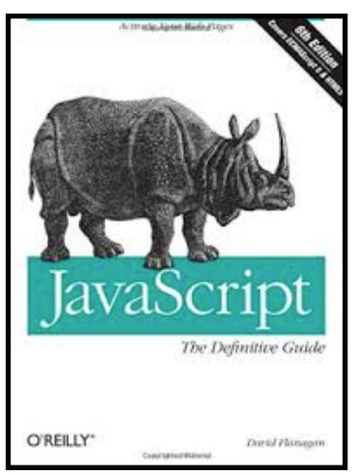


Automated Testing of Web Sites

StackFul automatically tests web sites and reports bugs

```
1 var shoppingCart = undefined;
2
3 document.getElementById("JS_book").addEventListener("click", function() {
4   shoppingCart = { name: "JavaScript_textbook", price: 19.99 };
5 });
6 document.getElementById("buy_button").addEventListener("click", function() {
7   var price = shoppingCart.price;
8     .....
9 });
```

Line 5: TypeError: Cannot read property 'price' of undefined

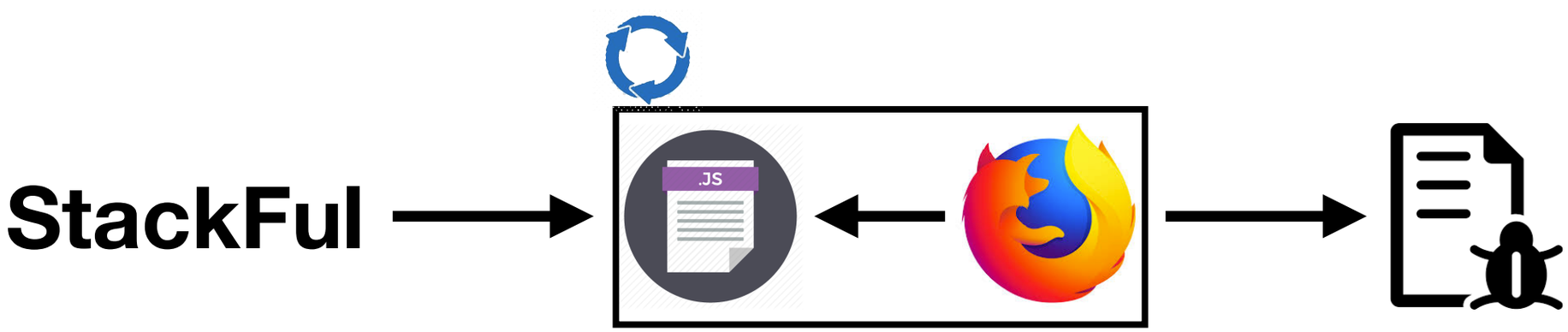


Automated Testing of Web Sites

StackFul automatically tests web sites and reports bugs

```
1 var shoppingCart = undefined;
2
3 document.getElementById("JS_book").addEventListener("click", function() {
4   shoppingCart = { name: "JavaScript_textbook", price: 19.99 };
5 });
6 document.getElementById("buy_button").addEventListener("click", function() {
7   var price = shoppingCart.price;
8     .....
9 });
```

```
Line 5: TypeError: Cannot read property 'price' of undefined
```



Testing of Regular Applications

```
var a = Math.randomInt();
var b = readUserInputNumber();
var c = 100;
if (a > c) {
  if (b == 10) {
    console.log("OK 1");
  } else {
    throw new Error();
  }
} else {
  console.log("OK 2");
}
```

Testing of Regular Applications

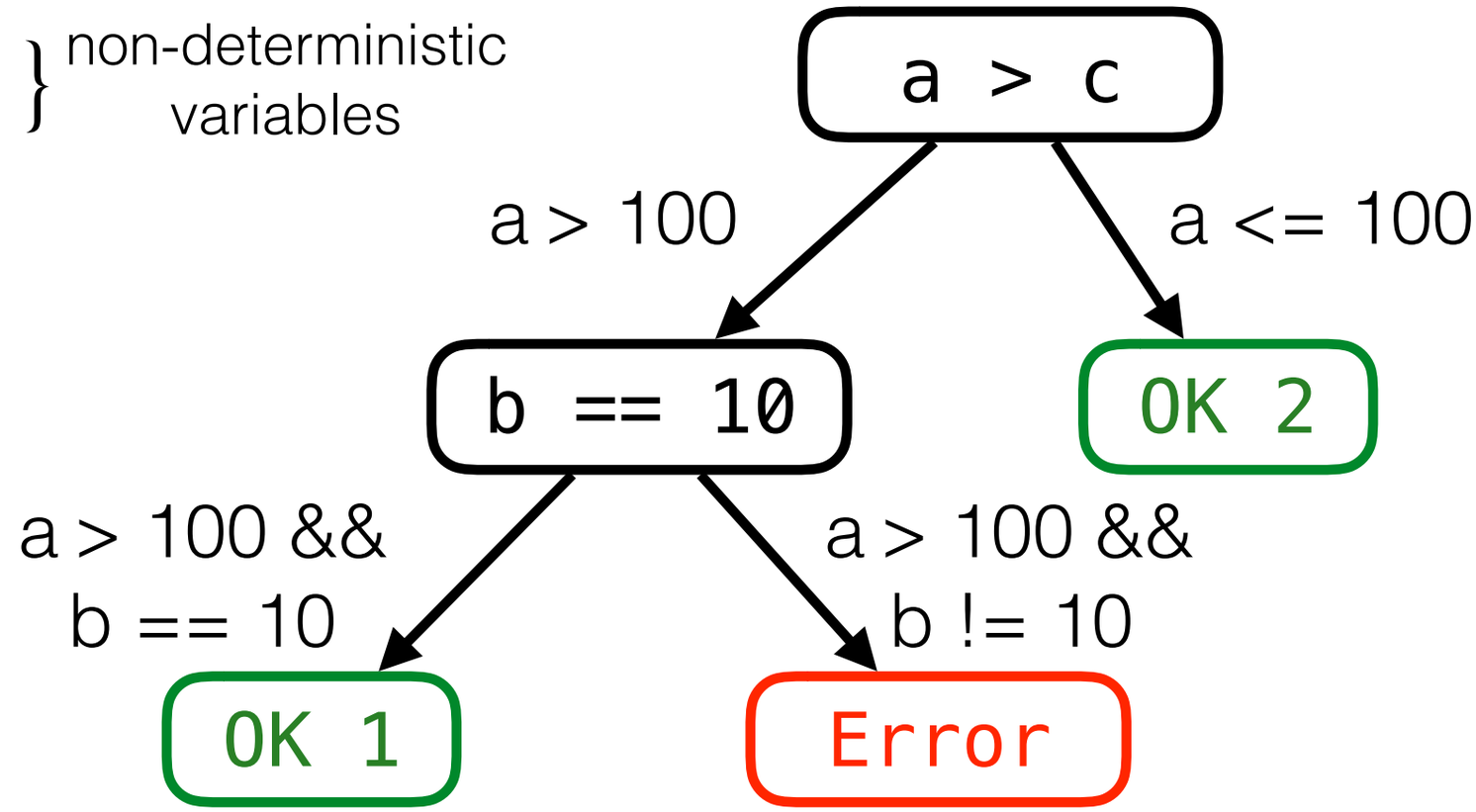
```
var a = Math.randomInt();
var b = readUserInputNumber();
var c = 100;
if (a > c) {
  if (b == 10) {
    console.log("OK 1");
  } else {
    throw new Error();
  }
} else {
  console.log("OK 2");
}
```

} non-deterministic variables

Testing of Regular Applications

```
var a = Math.randomInt();
var b = readUserInputNumber();
var c = 100;
if (a > c) {
  if (b == 10) {
    console.log("OK 1");
  } else {
    throw new Error();
  }
} else {
  console.log("OK 2");
}
```

} non-deterministic variables

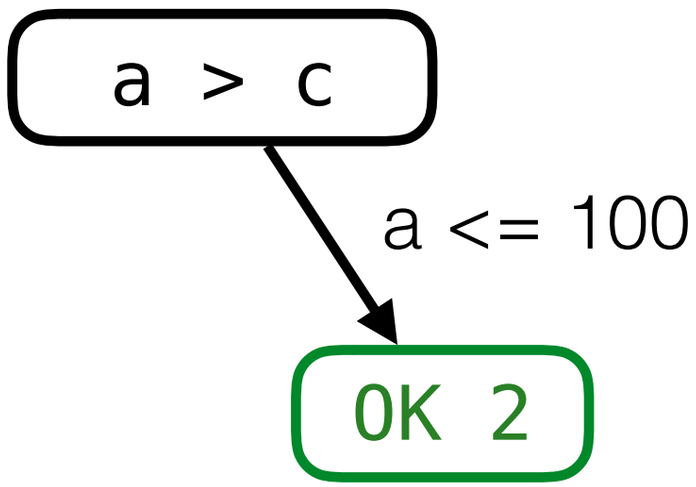


3 program paths
Choice of path depends on **a** and **b**
Iteratively execute program to explore all paths

Testing of Regular Applications

First iteration: assign random numbers to ND variables
Symbolically represent conditions (= *path condition*)

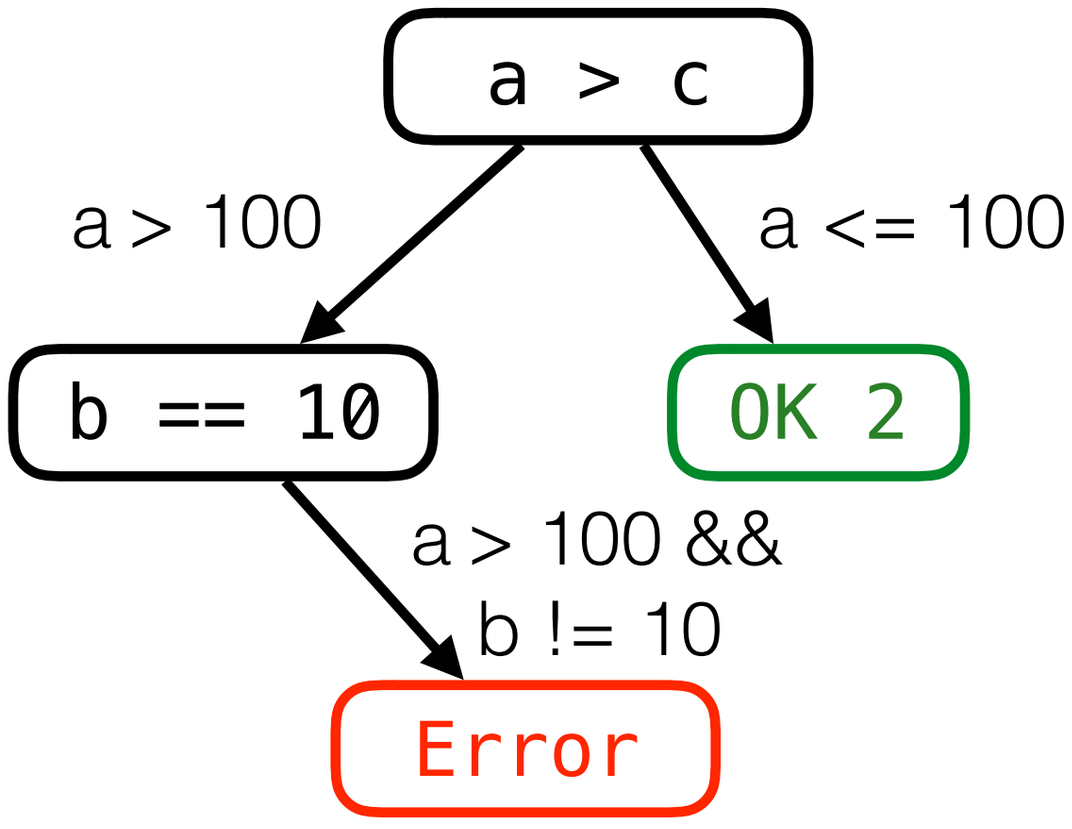
```
var a = Math.randomInt();          42
var b = readUserInputNumber();     123
var c = 100;
if (a > c) {
  if (b == 10) {
    console.log("OK 1");
  } else {
    throw new Error();
  }
} else {
  console.log("OK 2");
}
```



Testing Regular Applications

Second iteration: use path condition to manipulate value
Choose value so that a > 100

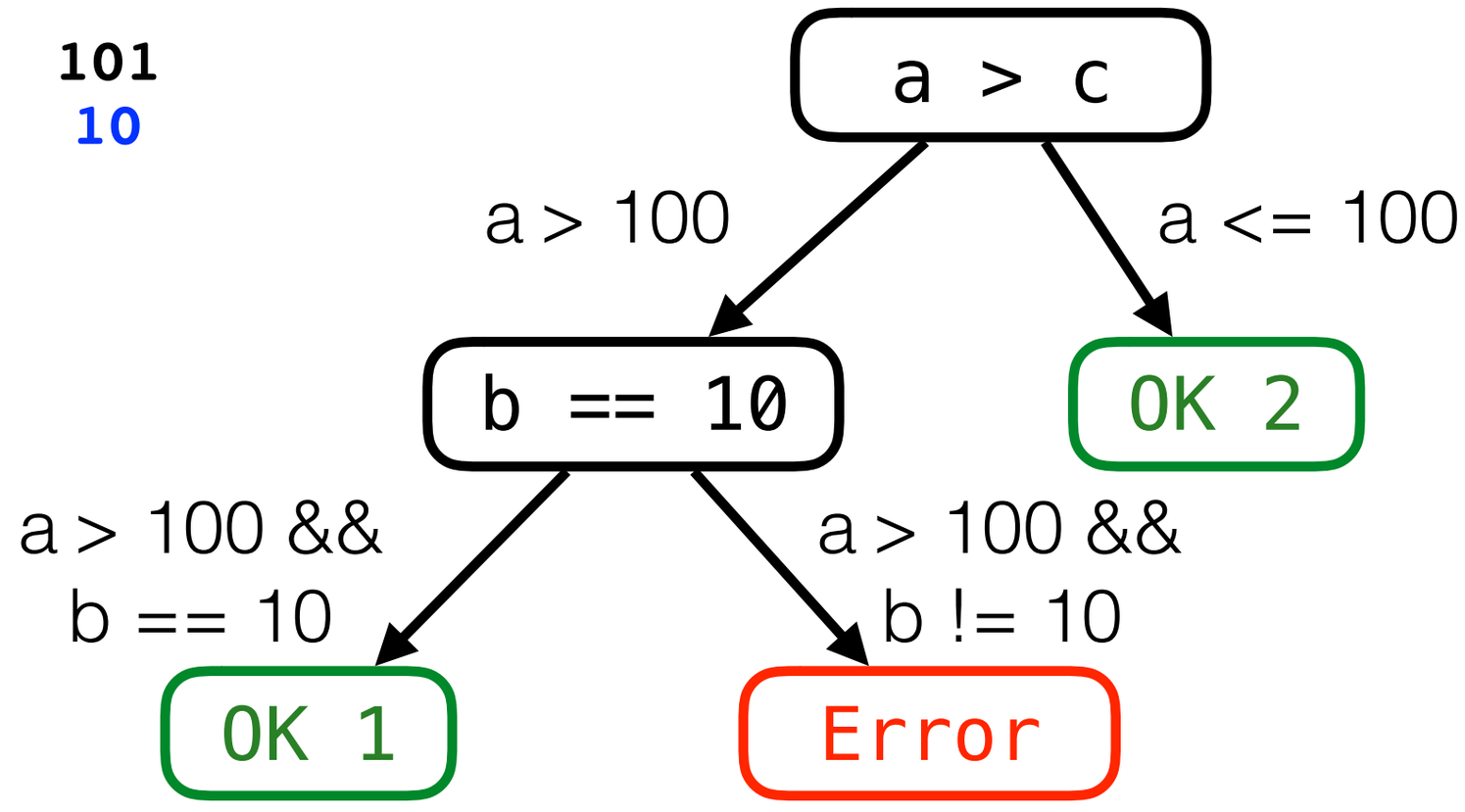
```
var a = Math.randomInt();          101
var b = readUserInputNumber();     123
var c = 100;
if (a > c) {
  if (b == 10) {
    console.log("OK 1");
  } else {
    throw new Error();
  }
} else {
  console.log("OK 2");
}
```



Testing of Regular Applications

Third iteration: use path condition to manipulate value
Choose value so that a > 100 **and** b == 10

```
var a = Math.randomInt();                101
var b = readUserInputNumber();           10
var c = 100;
if (a > c) {
  if (b == 10) {
    console.log("OK 1");
  } else {
    throw new Error();
  }
} else {
  console.log("OK 2");
}
```



Testing Web Sites

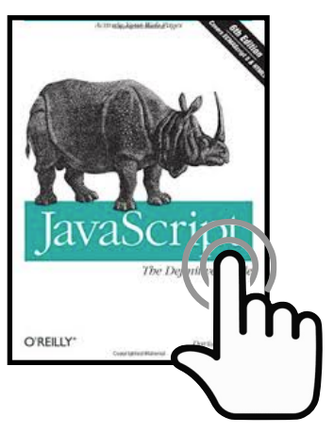
```
1 var shoppingCart = undefined;
2
3 document.getElementById("JS_book").addEventListener("click", function() {
4   shoppingCart = { name: "JavaScript_textbook", price: 19.99 };
5 });
6 document.getElementById("buy_button").addEventListener("click", function() {
7   var price = shoppingCart.price;
8     .....
9 });
```

Testing Web Sites

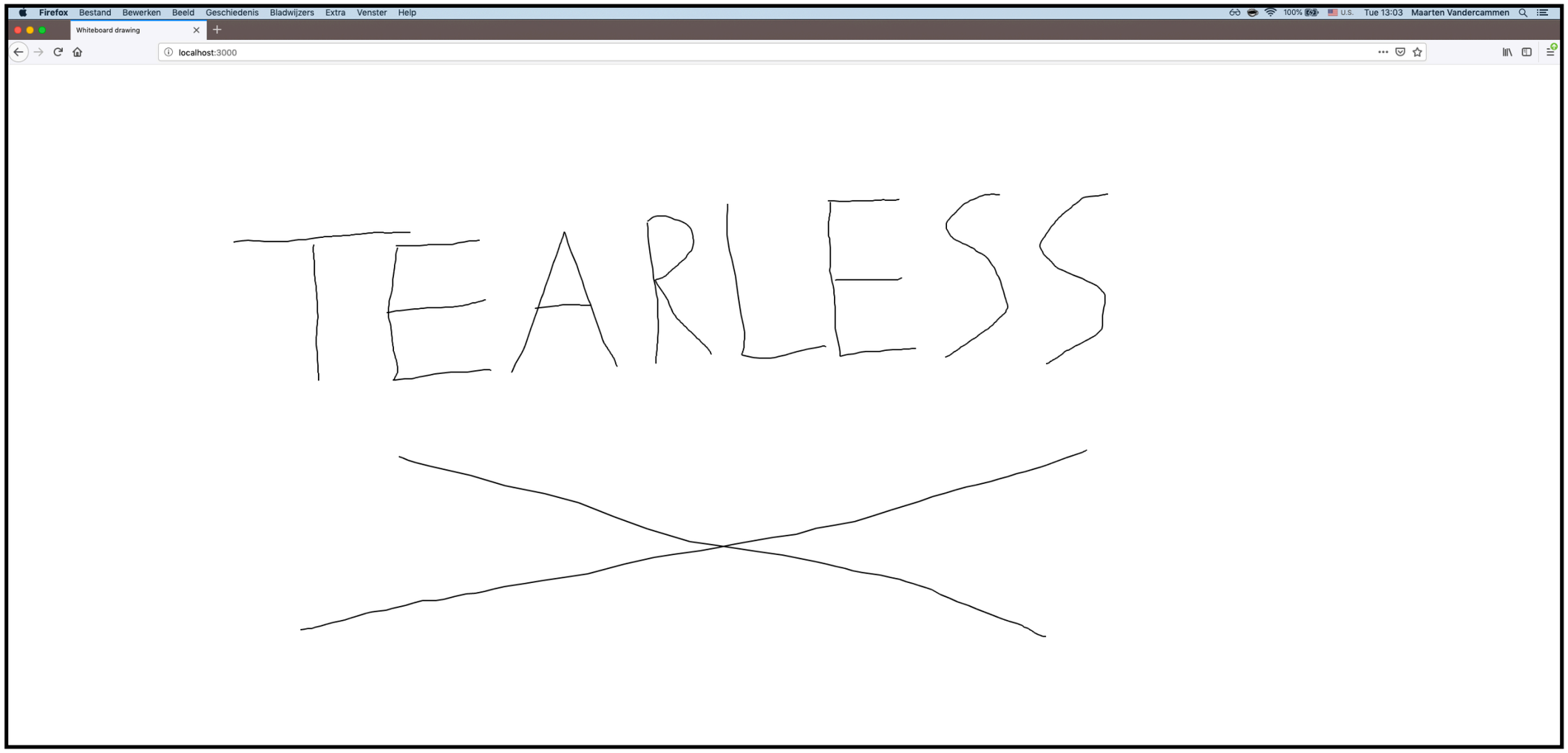
```
1 var shoppingCart = undefined;
2
3 document.getElementById("JS_book").addEventListener("click", function() {
4   shoppingCart = { name: "JavaScript_textbook", price: 19.99 };
5 });
6 document.getElementById("buy_button").addEventListener("click", function() {
7   var price = shoppingCart.price;
8     .....
9 });
```

Web sites event-driven

Automatically **detect** events-handlers and **generate** events

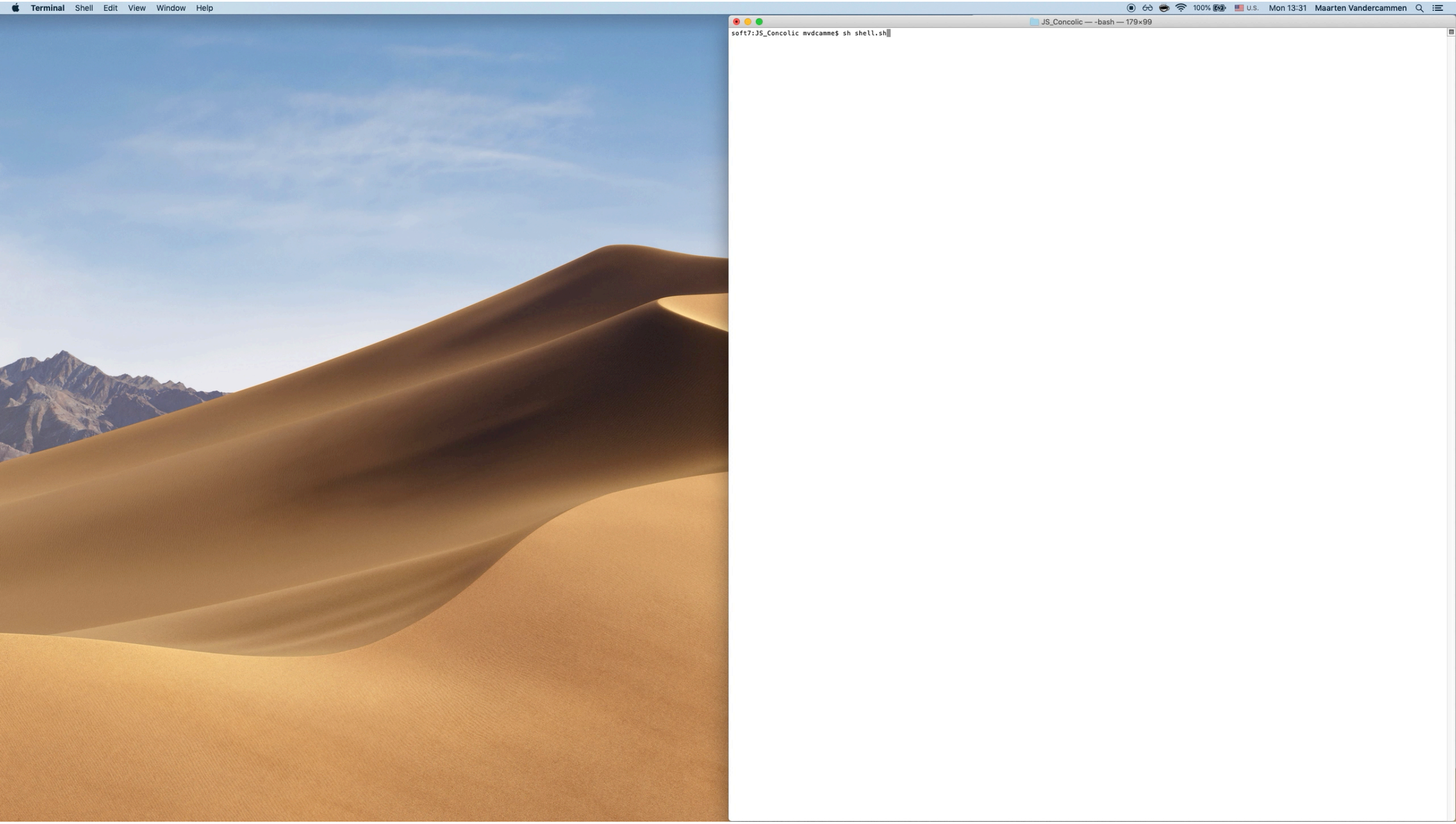


StackFul Demo

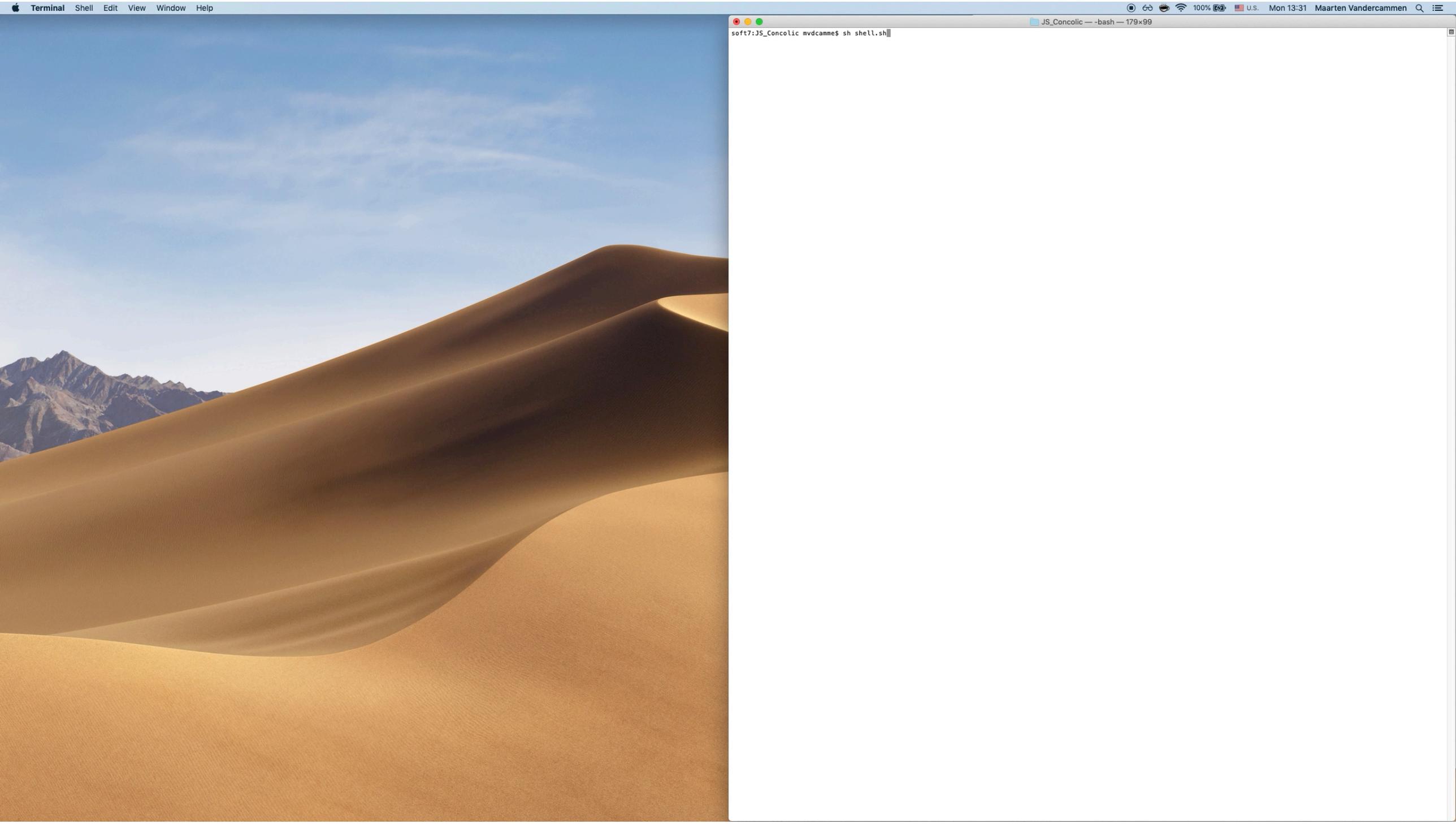


```
assert(! linesAreOverlapping(linesDrawn), "Shouldn't happen");

function sanitizeMouseClicks(x0, y0, x1, y1) {
    ...
}
```

StackFul Demo



StackFul Demo



Full-Stack Web Applications

Traditional Web Applications

Client

Server

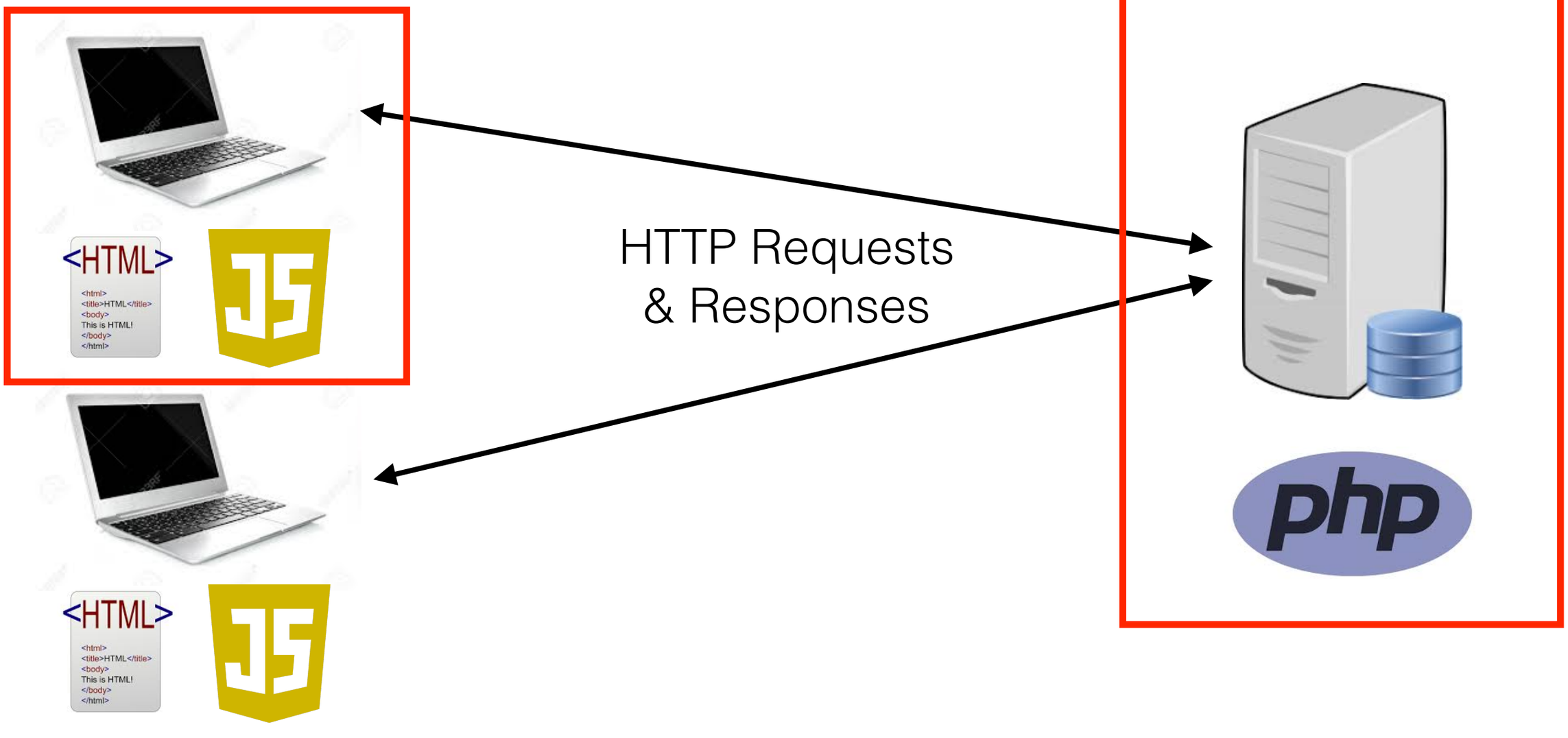
HTTP Requests
& Responses

Traditional Web Applications

Test client and server in isolation (i.e., intra-process testing)

Client

Server



Full-Stack Applications

Client

Server

Full-Stack Applications

Client

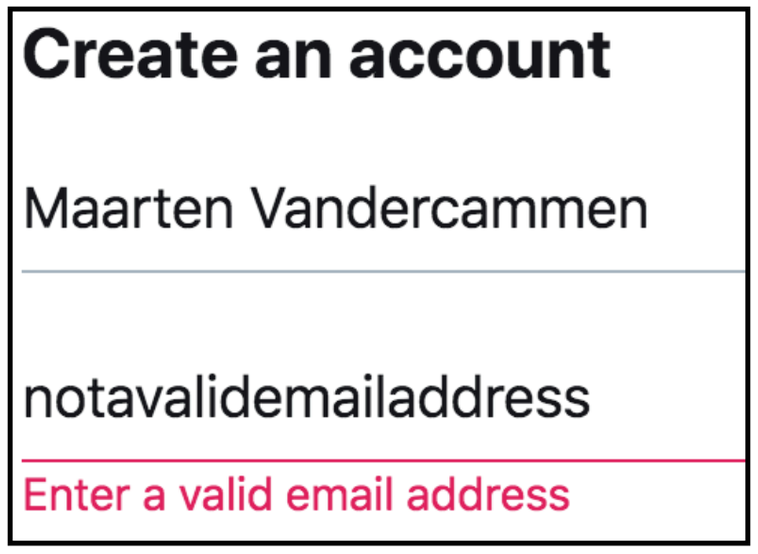
Inter-process testing

Server

Whole application tested holistically → Increase precision

Intra-process Testing

Client



Server

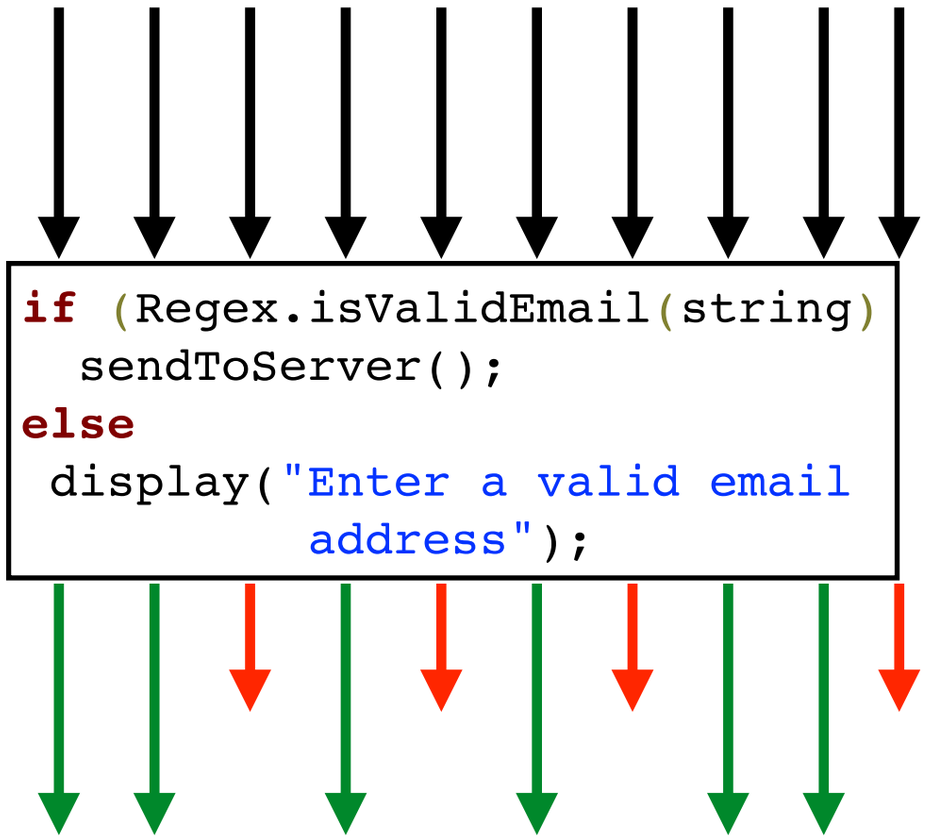
```
if (Regex.isValidEmail(string)
  sendToServer();
else
 display("Enter a valid email
          address");
```

```
if (filter_var($email,
FILTER_VALIDATE_EMAIL))
  register_account();
else
  throw new Exception();
```

Intra-process Testing

Client

random input strings

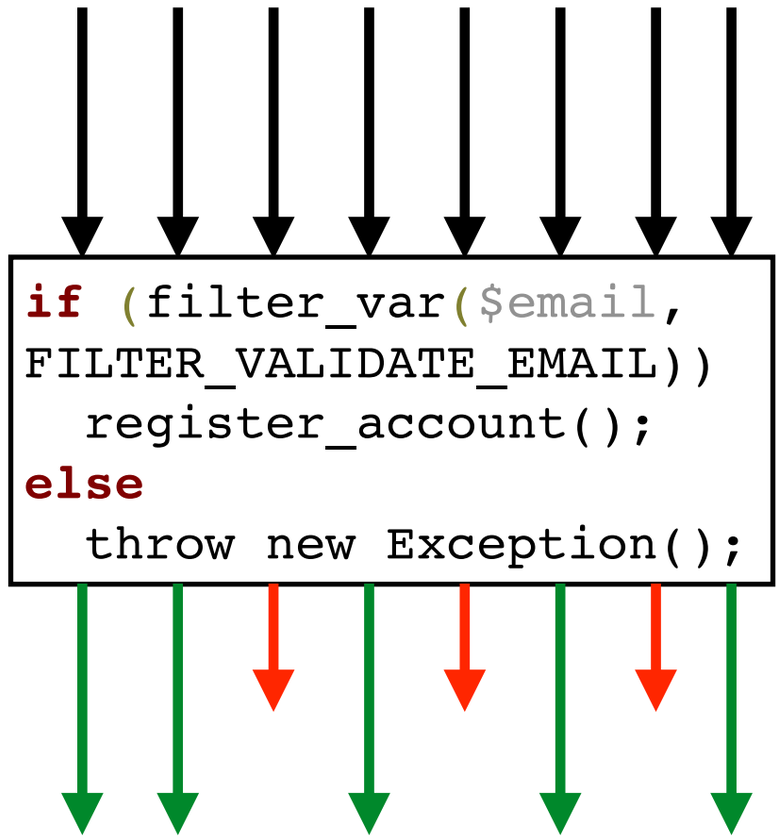


Valid strings:
proceed with testing

Invalid strings:
testing aborted early

Server

random input strings



Inter-process Testing

Client

random input strings

```
if (Regex.isValidEmail(string)
  sendToServer();
else
 display("Enter a valid email
          address");
```

Server

```
if
(Regex.isValidEmail(string))
  registerAccount();
else
  throw new Error();
```

Inter-process Testing

Prototype: **StackFul**

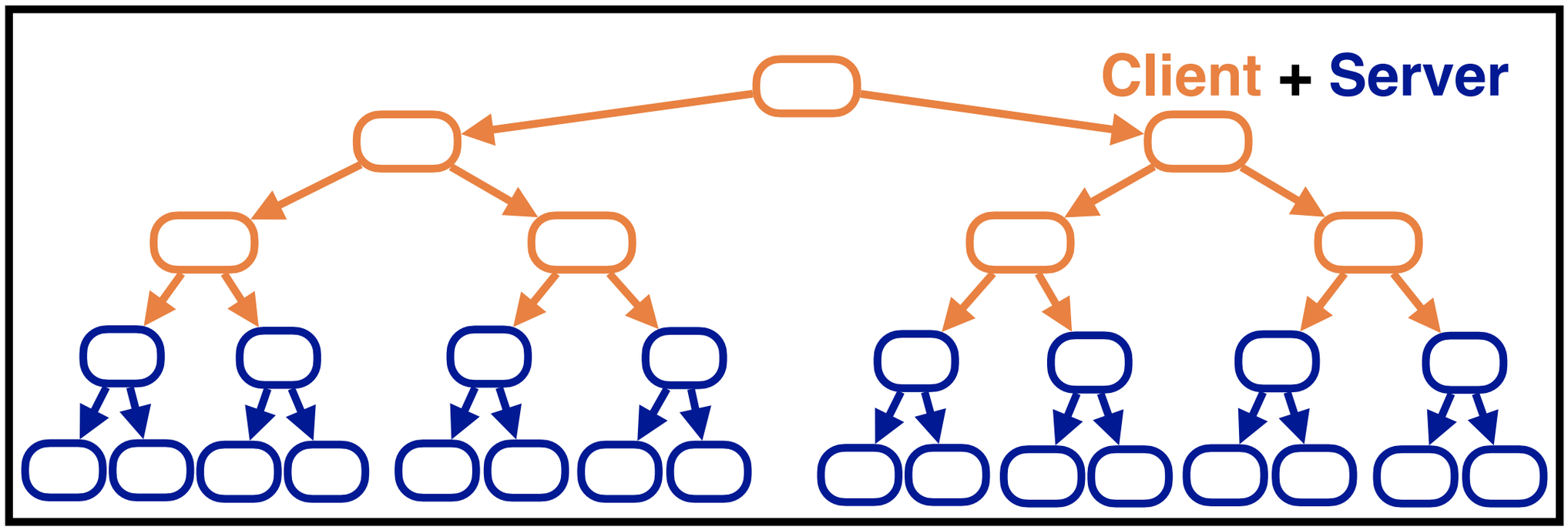
Features:

+ More information on how data flows throughout the entire system
+ Eliminate certain types of false positive errors
+ More practical error reports
- Slower

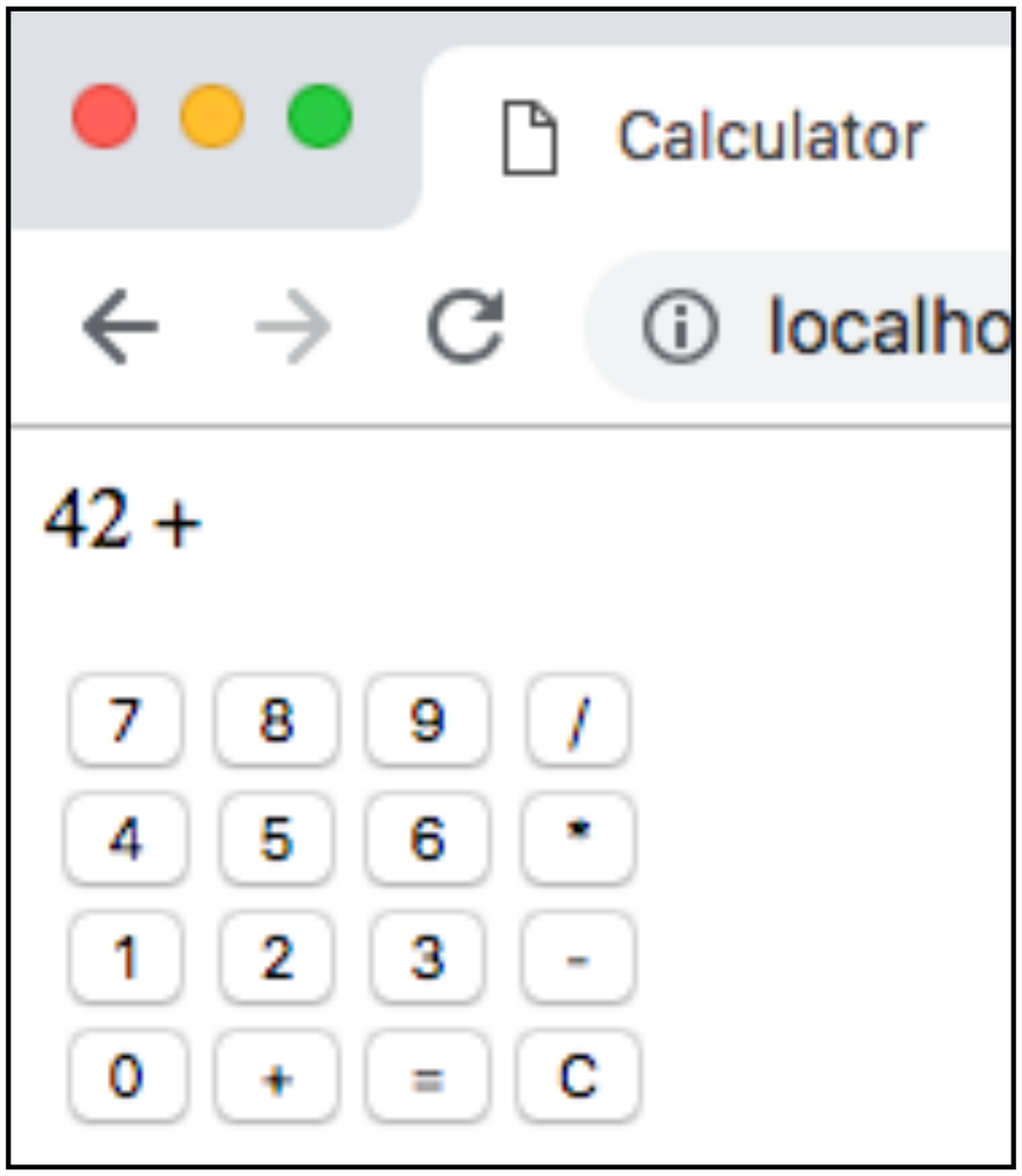
Not intended to **replace** intra-process testing

Inter-process Slowdown

Number of paths exponential in function of branches & events



Motivating Example



Motivating Example

```
// ... Setting up the server
var ws = new require('ws').Server({ port: 3000 });
ws.on("connection", function(socket) {
  // A new client has connected
  ws.onmessage = function(msg) {
    // Receive input from client
    var left = msg.left, op = msg.op, right = msg.right;
    var result;
  switch (op) {
  case "+": result = left + right; break;
  case "-": result = left - right; break;
  case "*": result = left * right; break;
  case "/":
    if (right === 0) {
      throw new Error("Dividing by zero");
    }
    result = left / right; break;
  default:
    throw new Error("Unknown operator");
  }
socket.send(result); // Send the result back to the client
```

Server

Error Report

Intra-process Testing

```
(Server): Tester detected error in file "index.js", at position (22:4)
ERROR: Dividing by zero
```

Error Report

Intra-process Testing

```
(Server): Tester detected error in file "index.js", at position (22:4)
ERROR: Dividing by zero
```

Inter-process Testing

```
(Server): Tester detected error in file "index.js", at position (22:4)
ERROR: Dividing by zero
Error encountered by triggering the following user events:
Clicked button "Button1"
Clicked button "Button/"
Clicked button "Button0"
Clicked button "Button="
```

Step-by-step guide for triggering error anywhere in the application

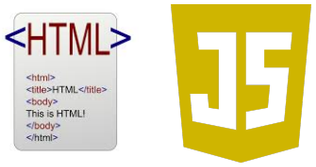
Motivating Example

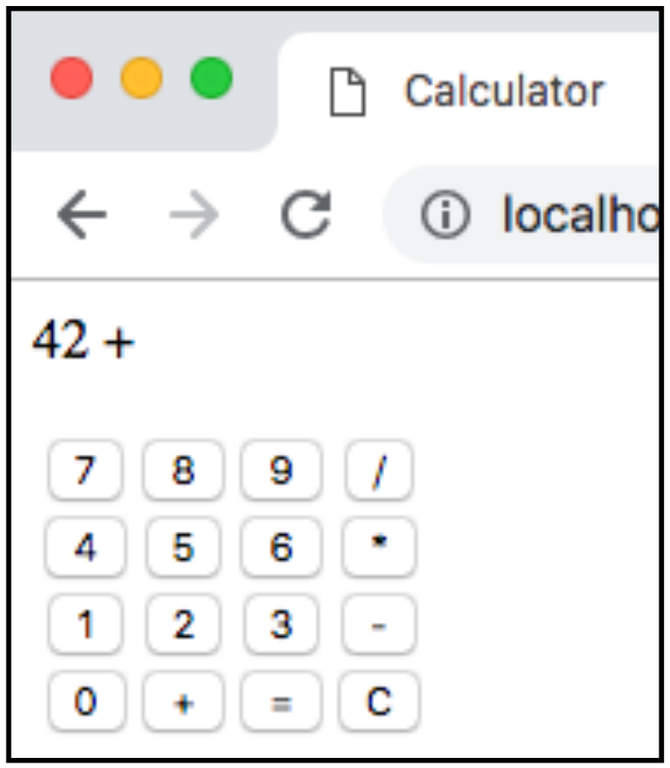
```
// Connect with the server via a WebSocket
var ws = new WebSocket('ws://localhost:3000');

document.getElementById("0").addEventListener("click",
  function (e) {
    clickDigit(0);
})}
document.getElementById("+").addEventListener("click",
  function (e) {
    clickOperator("+");
})
document.getElementById("=").addEventListener("click",
  function (e) {
    compute();
})

var input = {left:0, op:"", right:0};
function compute() {
  if (! isValidExpression(input)) {
    resultElement.innerHTML = "Expression is invalid";
  } else if (input.op === "/" && input.op === 0) {
    resultElement.innerHTML = "Cannot divide by zero";
  } else {
    // Send the expression to the server
    ws.send(input);
  }
}
ws.onmessage = function(result) {
  // Receive computation result from server
  resultElement.innerHTML = result
}
```

Client

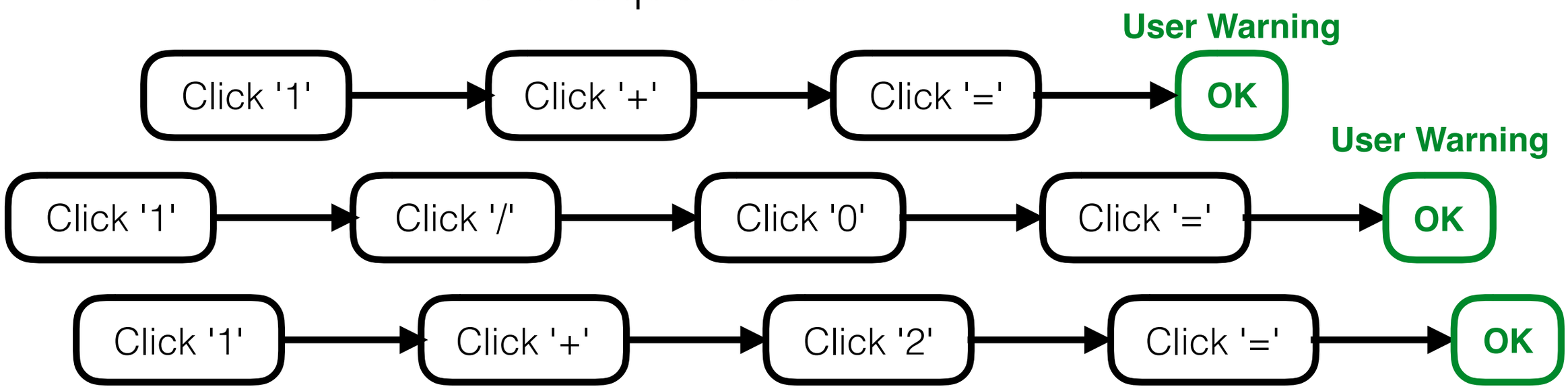
Traditional Automated Testing

Client





Automatically generate event sequences

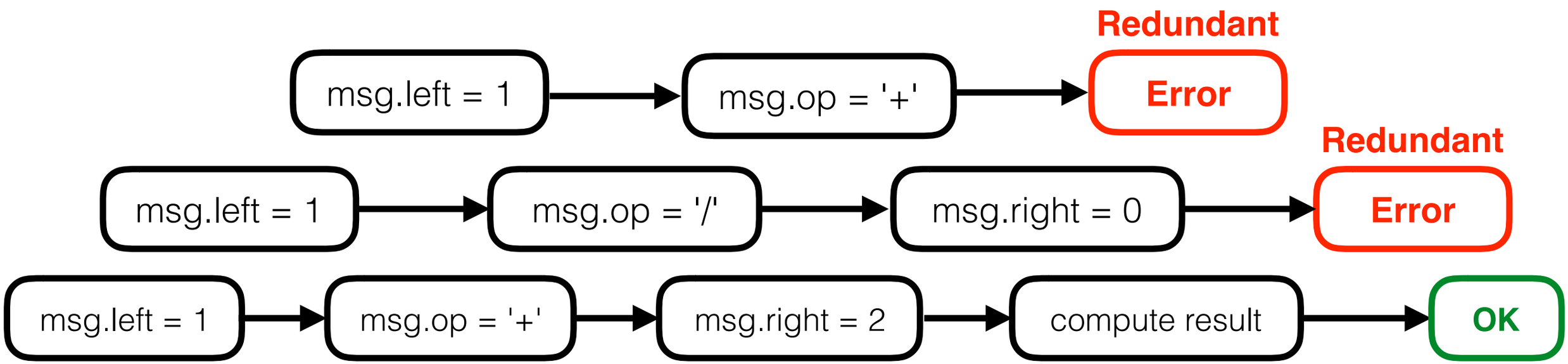


Traditional Automated Testing

Server

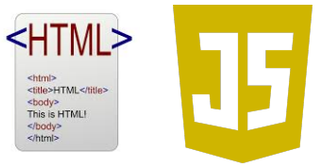
```
ws.onmessage = function(msg) {
  // Receive input from client
  var left = msg.left, op = msg.op, right = msg.right;
  var result;
  switch (op) {
  case "+": result = left + right; break;
  case "-": result = left - right; break;
  case "*": result = left * right; break;
  case "/":
    if (right === 0) {
      throw new Error("Dividing by zero");
    }
    result = left / right; break;
  default:
    throw new Error("Unknown operator");
}
```

Automatically generate input values for incoming messages

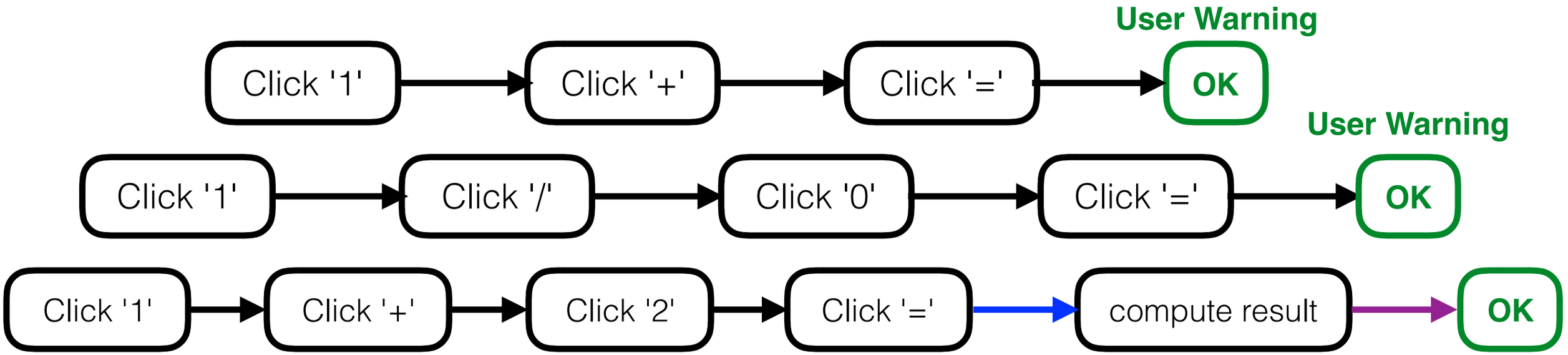


Inter-process Testing

Client Server



No false positive errors



Implementation

Implementation

Instrument code via **Aran** to perform
regular (*"concrete"*) + symbolic execution

```
var x = 1;
x + 10;
f(1);
```

Transformed into

```
Aran.declare("x", 1);
Aran.binary(Aran.read("x", x),
            "+",
            Aran.primitive(10));
Aran.apply("f", [arguments])
```

Symbolic +
concrete execution

```
Math.randomInt() + 10;
```

```
{ concrete: 42,
  symbolic: SymArithExp(SymInput(),
                        "+",
                        SymInt(10)}
```

Implementation

Symbolic execution is **cross-tier**

Client

```
var r = Math.randomInt();
sendToServer(r);
```

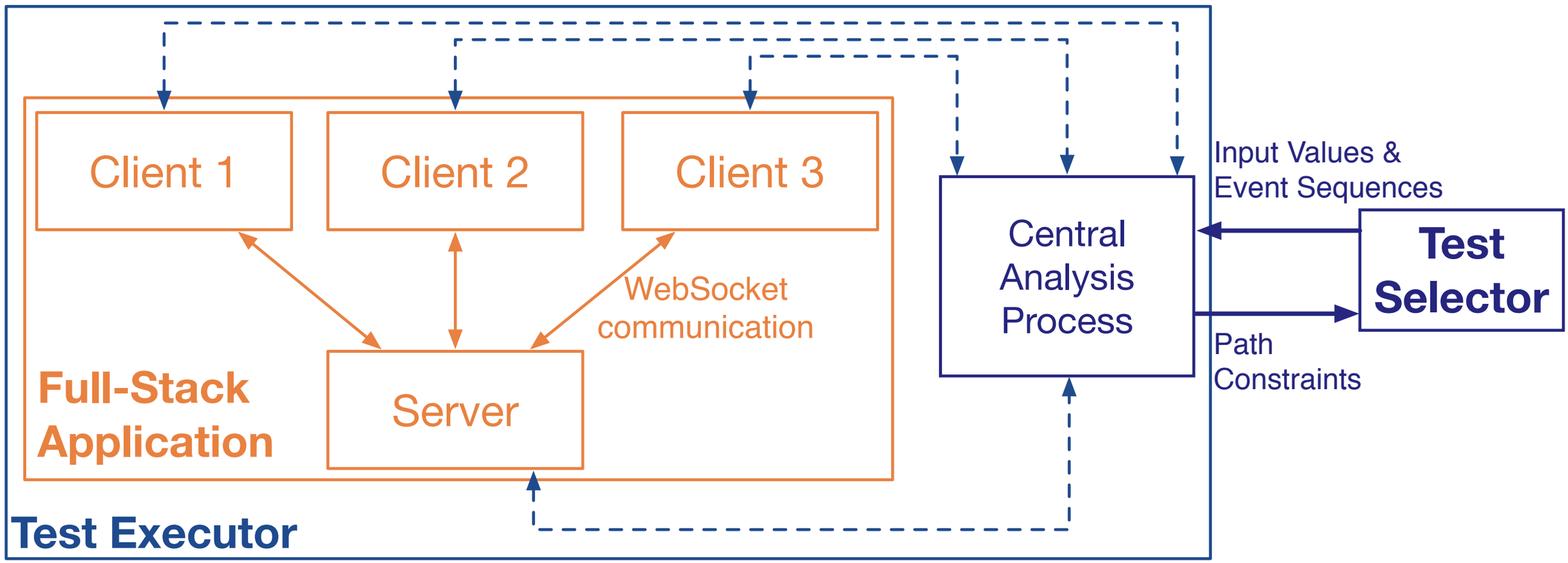
Server

```
var clientR = receiveFromClient();
if (clientR == 42) {
   doThen();
} else {
   doElse();
}
```

clientR == 42 ⟶

```
{ concrete: true,
  symbolic: SymComparisonExp(SymInput("client"),
                             "==",
                             SymInt(42) }
```

StackFul Architecture

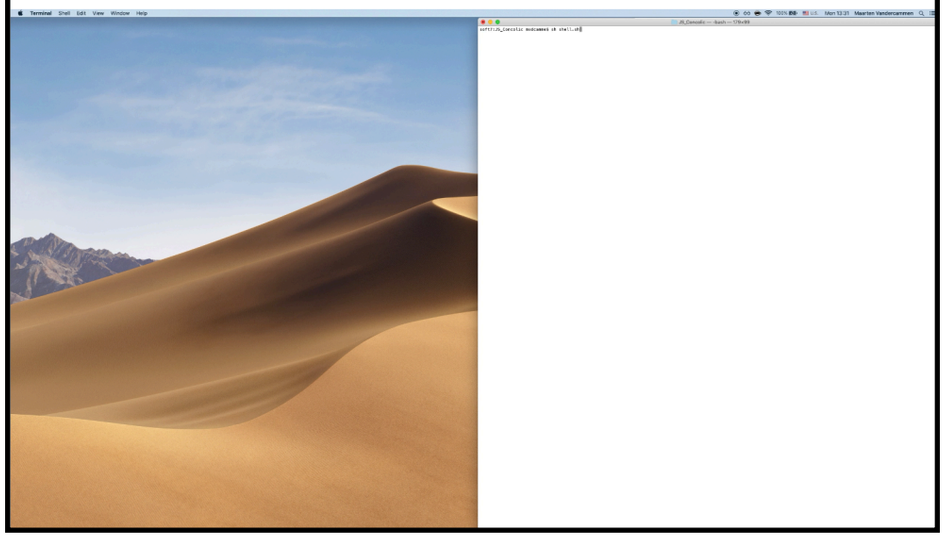



```
{
  concrete:
    42,
  symbolic:
    SymInput("client")
}
```

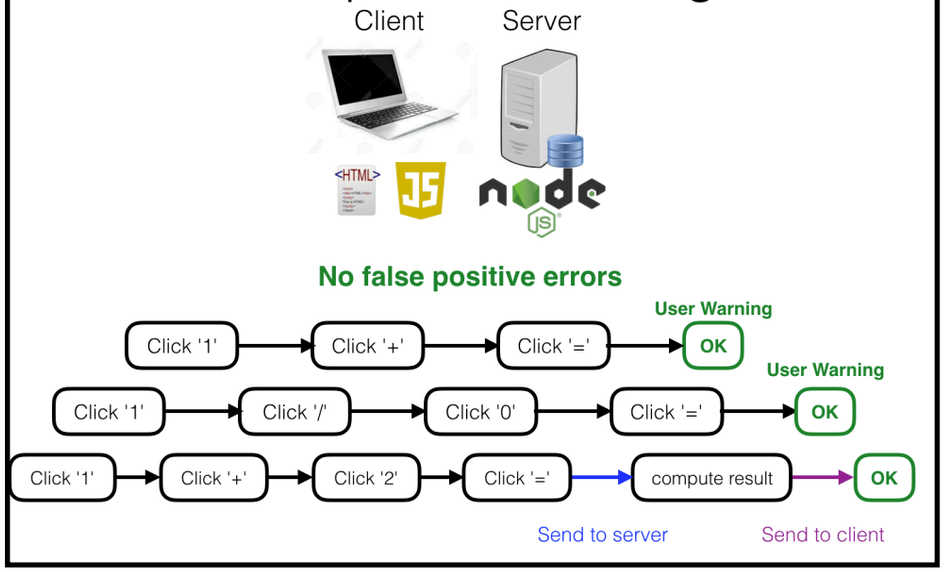
```
{
  concrete:
       true,
  symbolic:
       SymComparisonExp(SymInput("client"),
                        "==",
                        SymInt(42)
}
```

Summary

StackFul Demo



Inter-process Testing



StackFul Architecture