WEB SOCKETS

James Hunt

Chief Architect

Stark & Wayne

@iamjameshunt

https://github.com/jhunt

https://starkandwayne.com

A brief history of Web Applications

PART I

1992: Research Papers

```
#                                                World Wide Web Consortium (W3C) (p5 of 15)
   information appears in a manner consistent with author
   intent. Read about the Accessible Rich Internet Applications
   Working Group and the Web Accessibility Initiative (WAI).

   Comments are welcome through 16 November 2018.

Call for Review: Pointer Events Level 2 is a W3C Proposed
Recommendation

   16 October 2018 | Archive

   The Pointer Events Working Group has published a Proposed
   Recommendation of Pointer Events Level 2. The features in
   this specification extend or modify those found in Pointer
   Events, a W3C Recommendation that describes events and
   related interfaces for handling hardware agnostic pointer
   input from devices including a mouse, pen, touchscreen, etc.
   For compatibility with existing mouse based content, this
   specification also describes a mapping to fire Mouse Events
   for other pointer device types.

   Comments are welcome through 13 November 2018.

Upcoming Workshop: Web Standardization for Graph Data

   15 October 2018 | Archive

-- press space for next page --
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

```
GET /~tbernerslee/cern.html HTTP/0.9
```

```
HTTP/0.9 200 OK
Content-Type: text/html

<HTML>
  <HEAD>
    <TITLE>CERN - A Research Paper</TITLE>
  </HEAD>
  <BODY>
    <BLINK>It is pronounced NOO-klee-ur</BLINK>
  </BODY>
</HTML>
```

1999: Server-side rendering

```
GET /HTTP/1.0
Content-Length: 0
Accept: */*
Cookie: trackmeplease=ohyes;itsthe=90s



HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 32768
SetCookie: okay;iwill=trackyou;itsthe=90s

<!-- web32.hotbot.com; generated in 9.7s blistering seconds -->
<html><head><title><meta name="keywords"
value="hotbot,lycos,search,whateven-is-google">
<title>Hotbot So Awesome Lol</title><head><body>...
```

2006:
AJAX

```
GET /v1/index.html
GET /v1/data

GET /v1/data?since=201810261130
GET /v1/data?since=201810261135
GET /v1/data?since=201810261140
...
```

2007:
COMET

?

```
GET /v1/index.html
GET /v1/data/pipe    <---- never "closes"
```

2011:
Web Sockets

```
GET /v1/index.html
GET /v1/data

GET /v1/events HTTP/1.1
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Version: 13
Sec-WebSocket-Protocol: firehose
Origin: http://example.com


HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: firehose
```

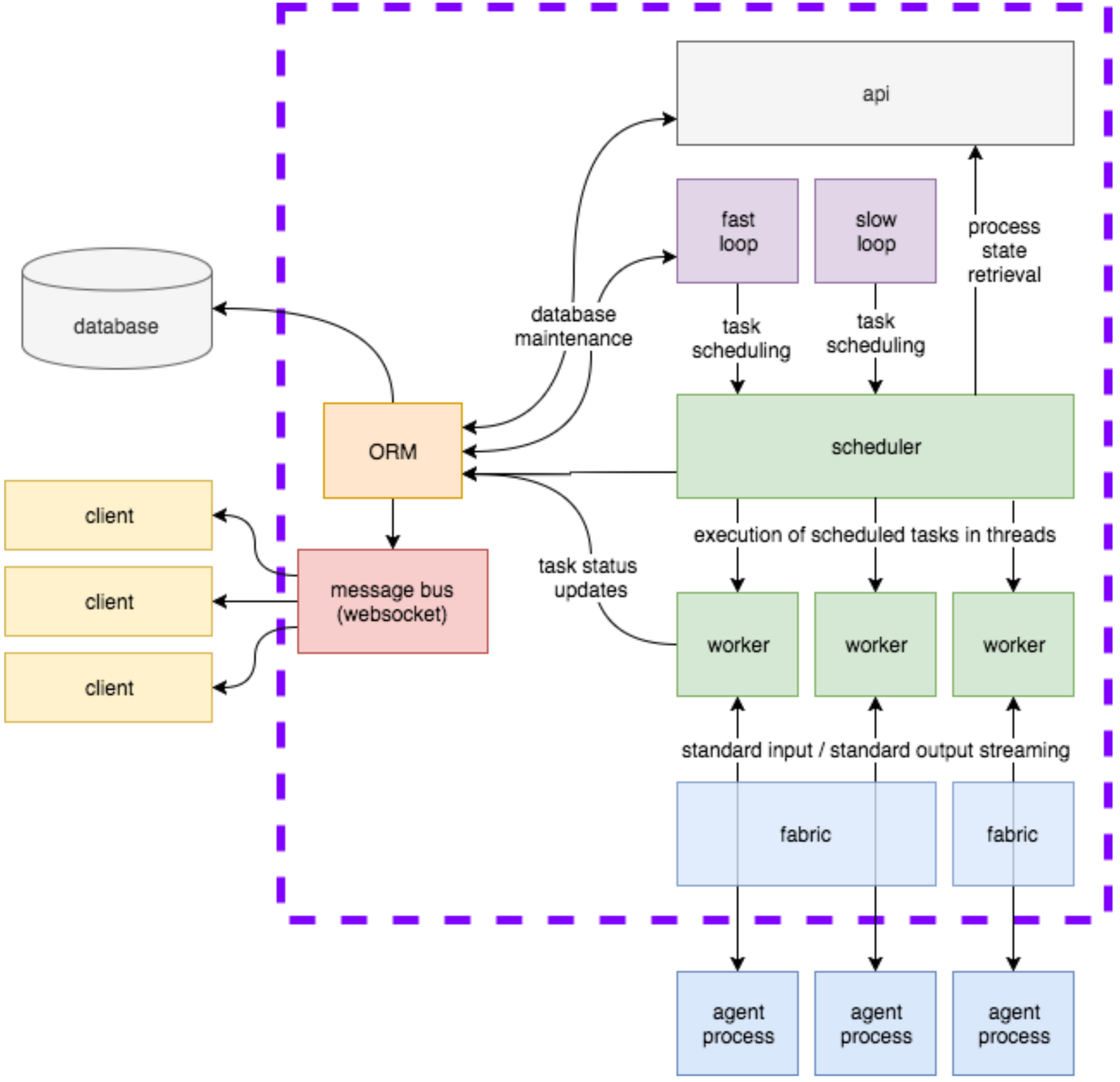
SO WHAT CAN WE DO WITH WEB SOCKETS?

A DEMO

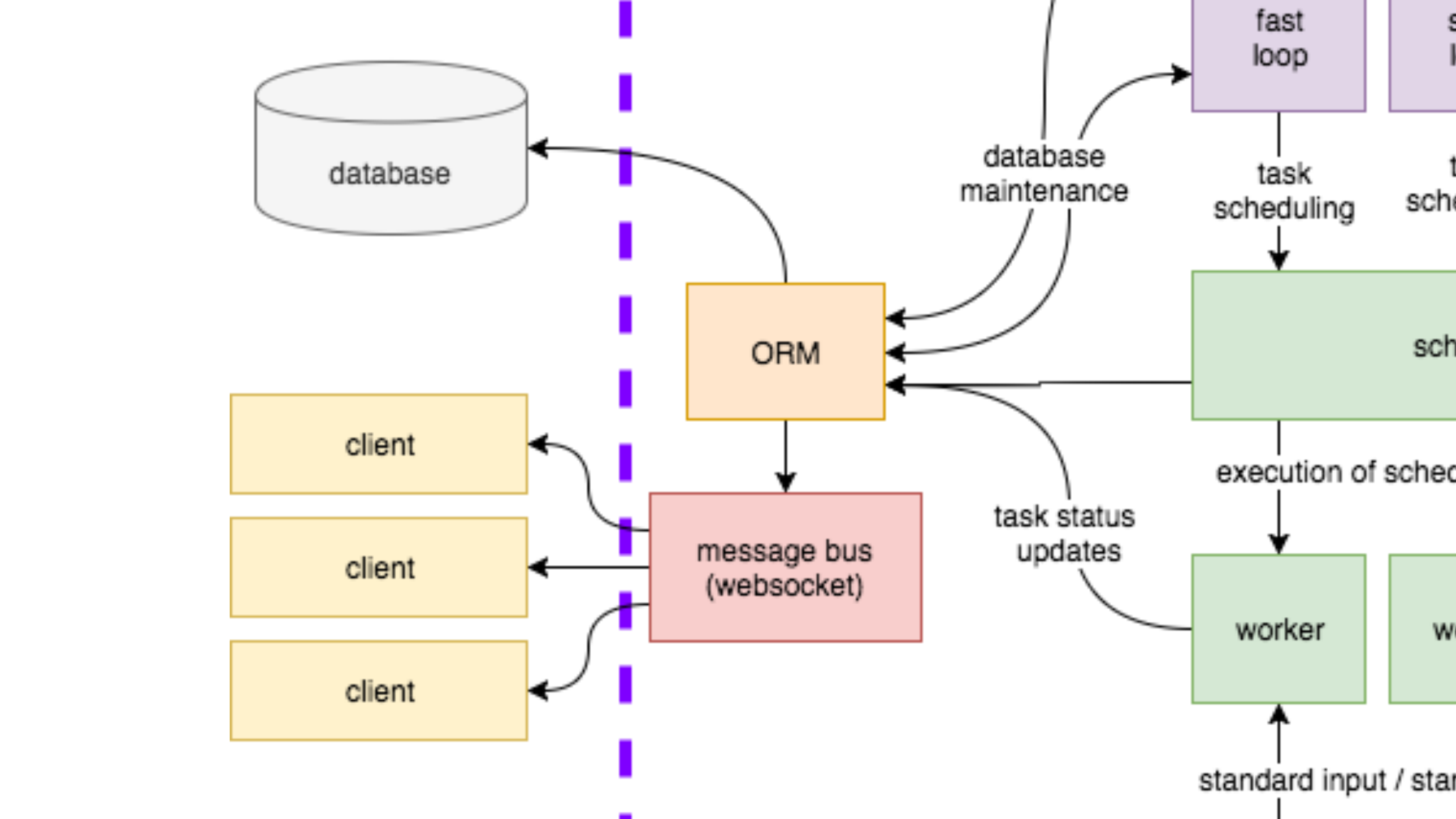
CECI N'EST PAS UN SLIDE

New Paradigm for SHIELD

```
GET  /v2/bearings
GET  /v2/events
```


api
fast loop
slow loop
process state retrieval
database
database maintenance
task scheduling
task scheduling
ORM
scheduler
client
client
client
message bus (websocket)
task status updates
execution of scheduled tasks in threads
worker
worker
worker
standard input / standard output streaming
fabric
fabric
agent process
agent process
agent process

THE MESSAGE BUS
(HONK, HONK!)


fast
loop
database
database
maintenance
task
scheduling
ORM
client
client
client
message bus
(websocket)
task status
updates
worker
execution of
standard input / sta

EASY PEASY

```go
type WebSocket struct {
    conn *websocket.Conn
}

func (r *Request) Upgrade() *WebSocket {
    log.Debugf("%s upgrading to WebSockets", r)

    upgrader := websocket.Upgrader{
        CheckOrigin: func(r *http.Request) bool { return true },
    }

    conn, err := upgrader.Upgrade(r.w, r.Req, nil)
    if err != nil {
        r.Fail(Oops(err, "an unknown error has occurred"))
        return nil
    }

    return &WebSocket{
        conn: conn,
    }
}

func (ws *WebSocket) Discard() {
    for {
        if _, _, err := ws.conn.NextReader(); err != nil {
            log.Infof("discarding message from ws client...")
            ws.conn.Close()
            break
        }
    }
}

func (ws *WebSocket) Write(b []byte) error {
    return ws.conn.WriteMessage(websocket.TextMessage, b)
}
```

```
ws = new WebSocket(url);

ws.onopen = function () {
  console.log('all socketed up!');
};

ws.onclose = function () {
  console.log('websocket closing...');
};

ws.onerror = function () {
  console.log('oops...');
};

ws.onmessage = function (m) {
  try {
    process(JSON.parse(m.data));

  } catch (e) {
    console.log("unable to parse event '%s' from stream: ", m.data, e);
  }
};
```

EASY PEASY

Things you can use WebSockets For:

A CHAT system!

An Event Stream!

Skyrim in the BROWSER?

Tailing Logs...

PAT HAS QUESTIONS
(I JUST KNOW HE DOES)