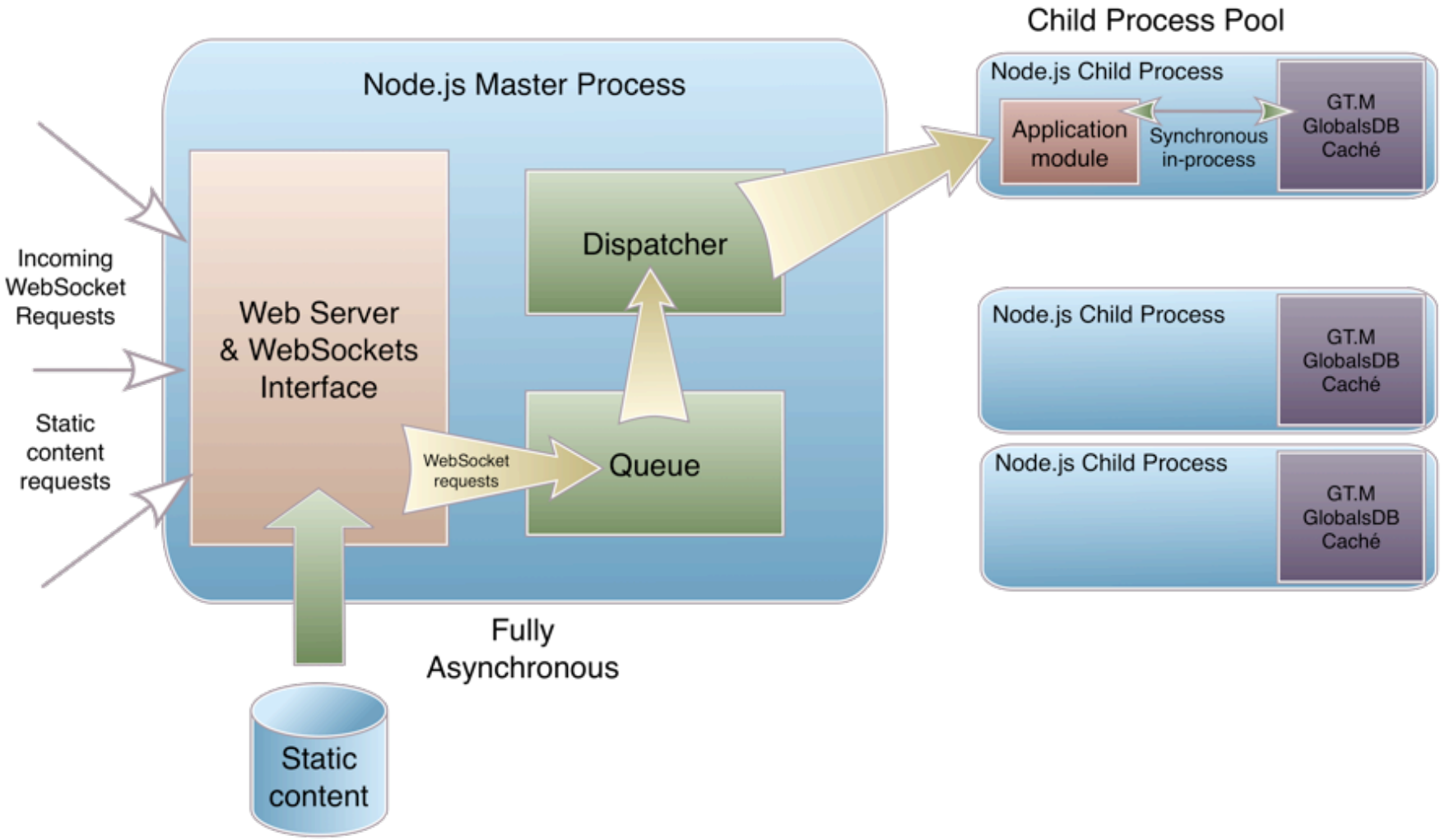
EWD.js Architecture

Rob Tweed
M/Gateway Developments Ltd

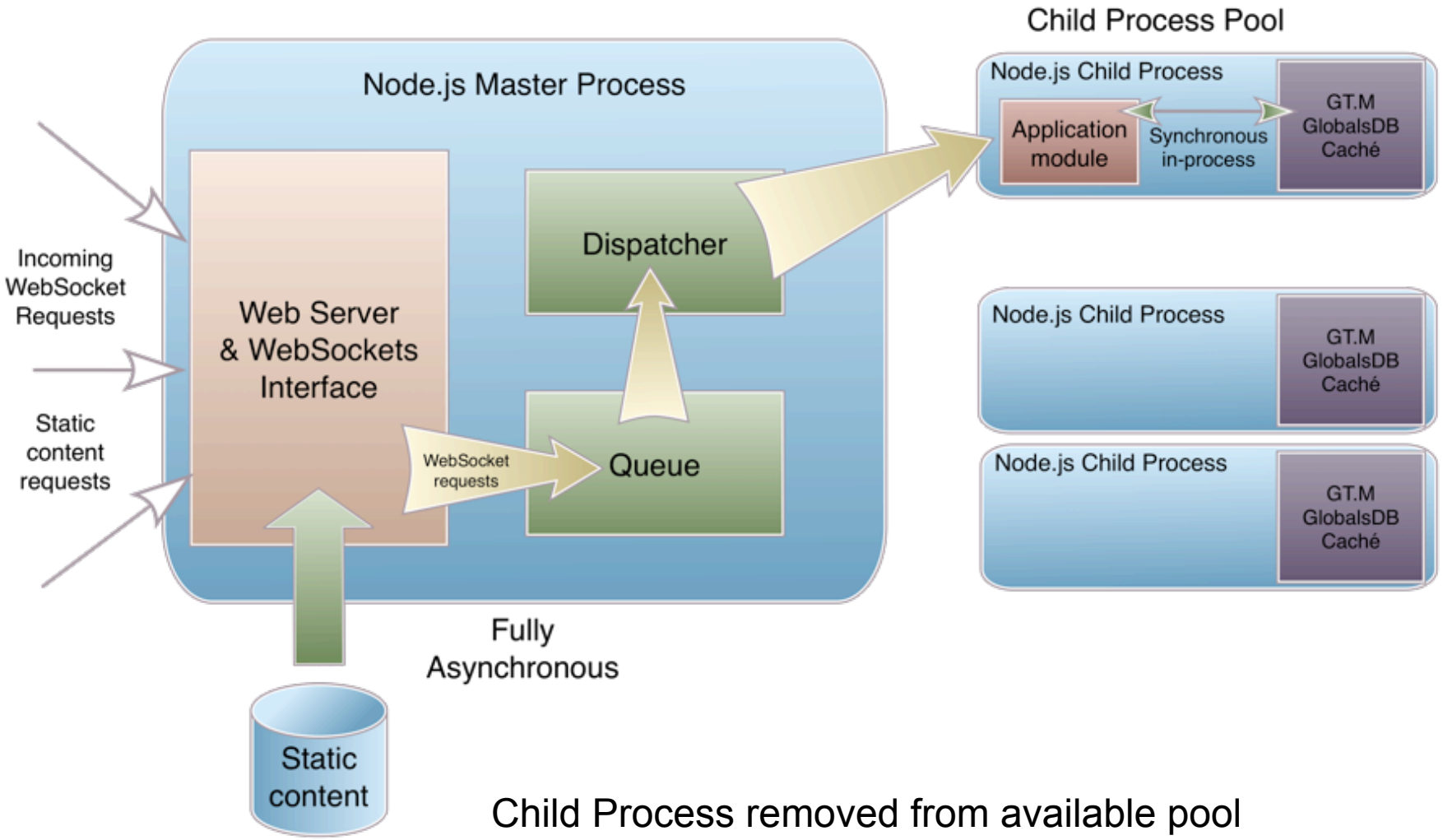
Twitter: @rtweed
Email: rtweed@mgateway.com

EWD.js Architecture



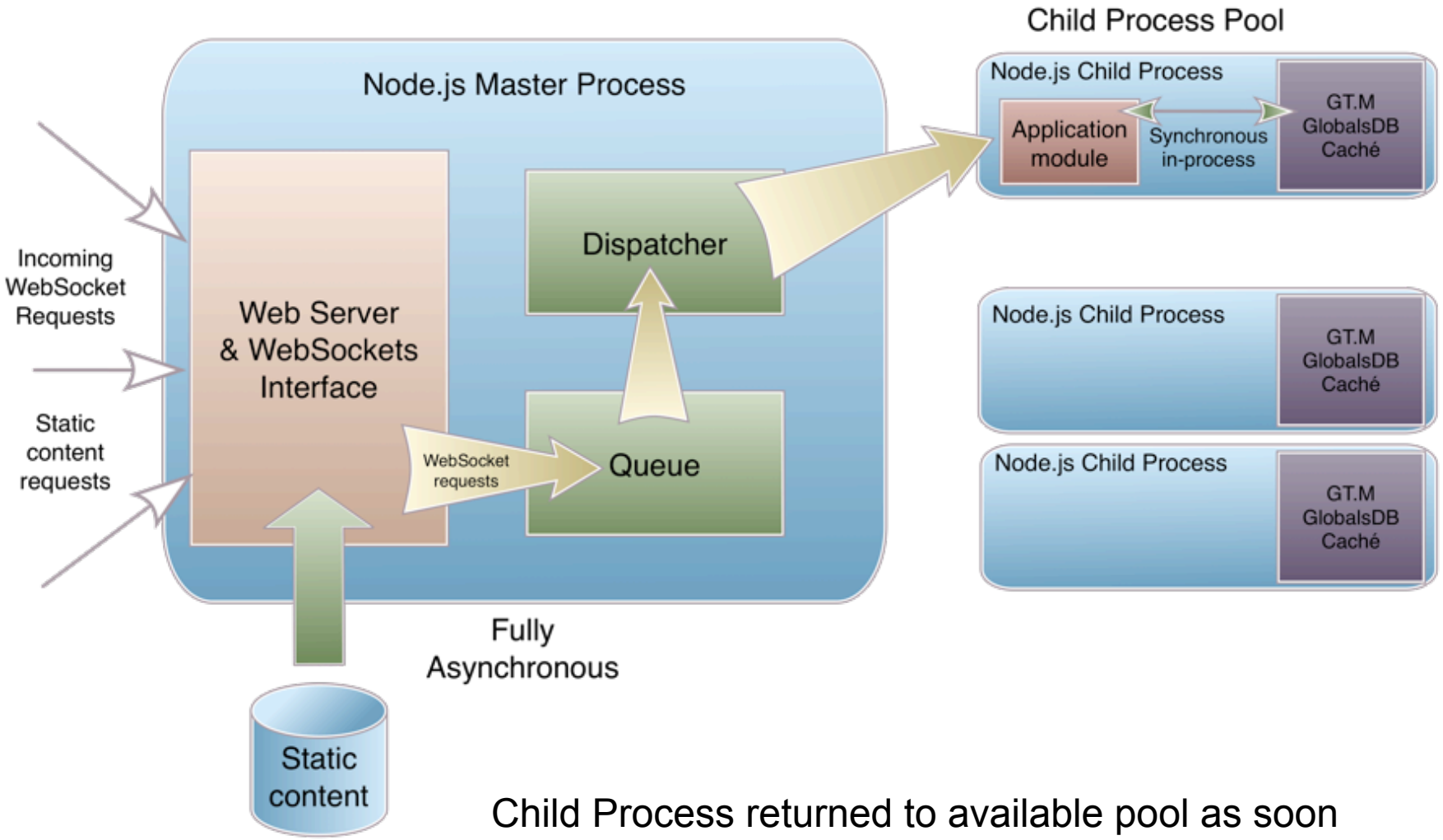
M/GATEWAY DEVELOPMENTS

EWD.js Architecture



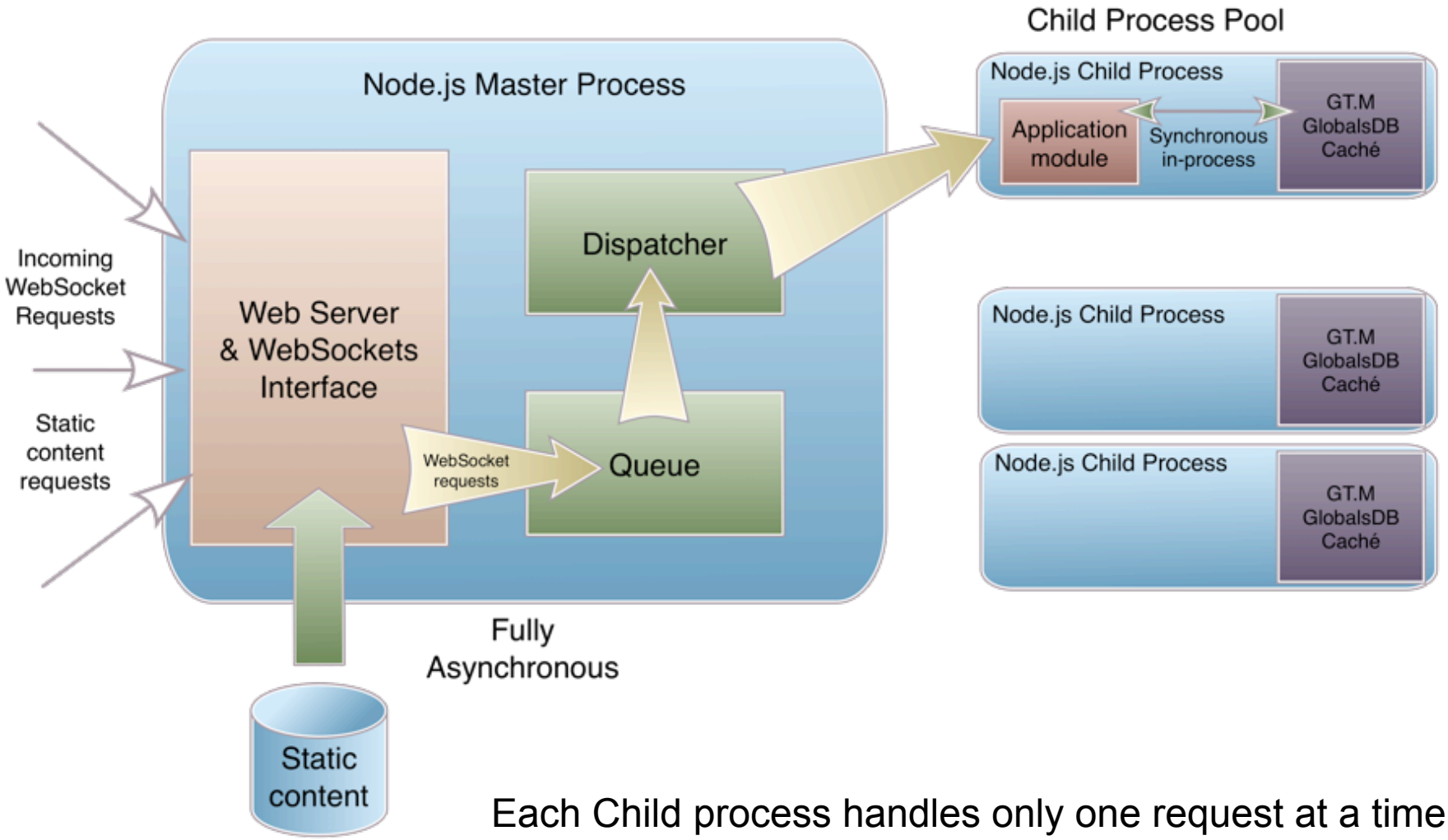
Child Process removed from available pool as soon as a request is sent to it

EWD.js Architecture



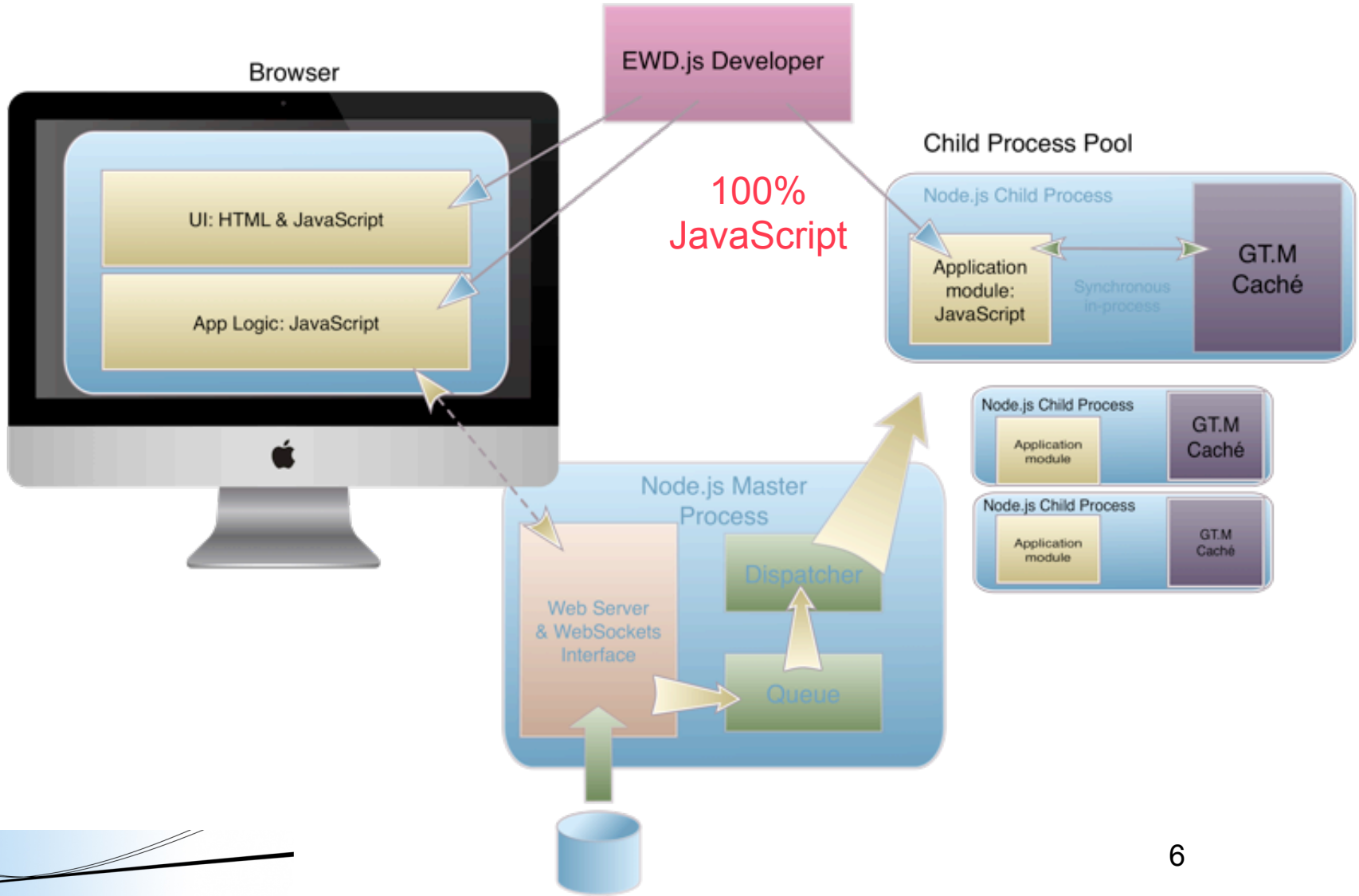
Child Process returned to available pool as soon as processing is completed

EWD.js Architecture



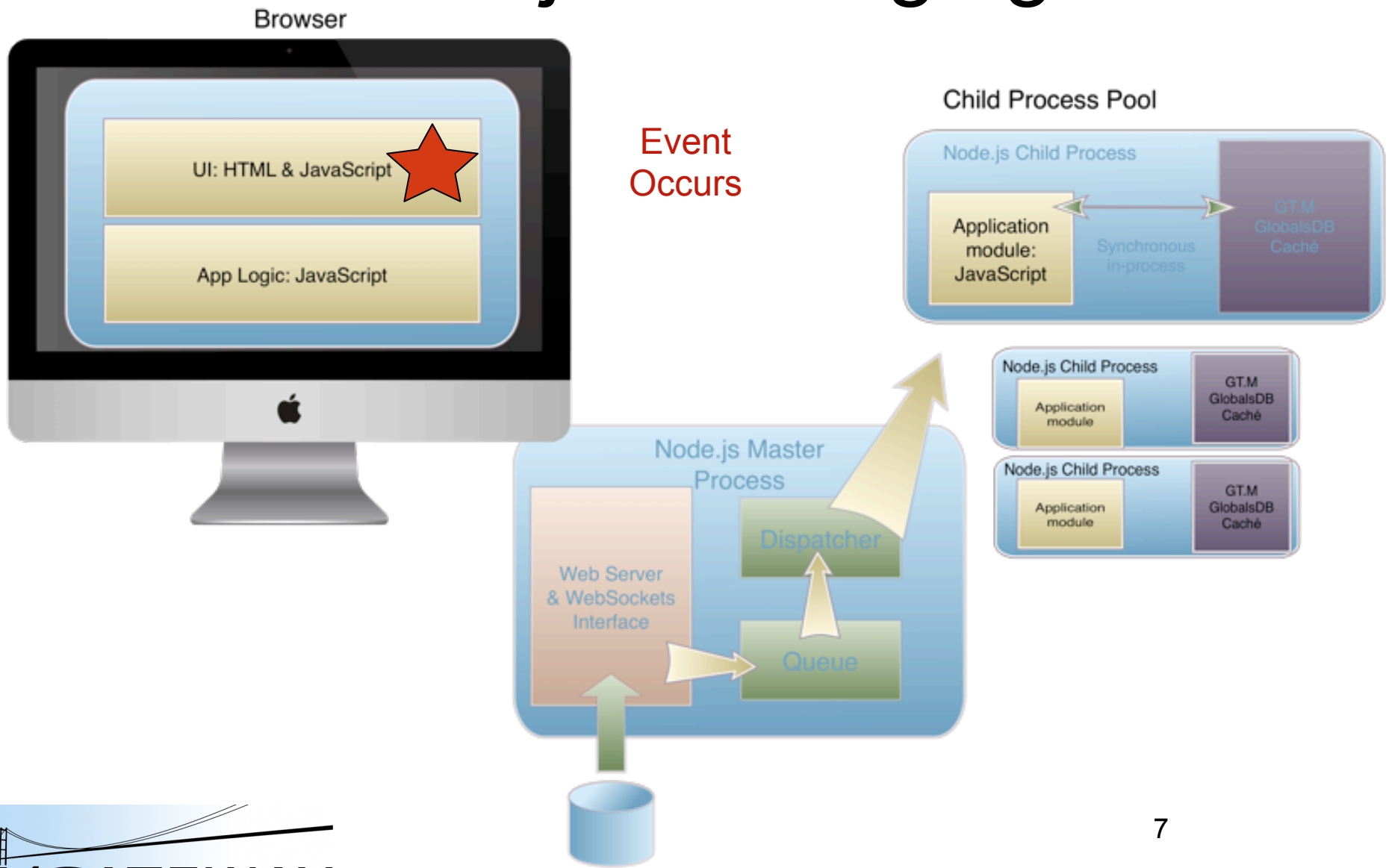
Each Child process handles only one request at a time

EWD.js Development

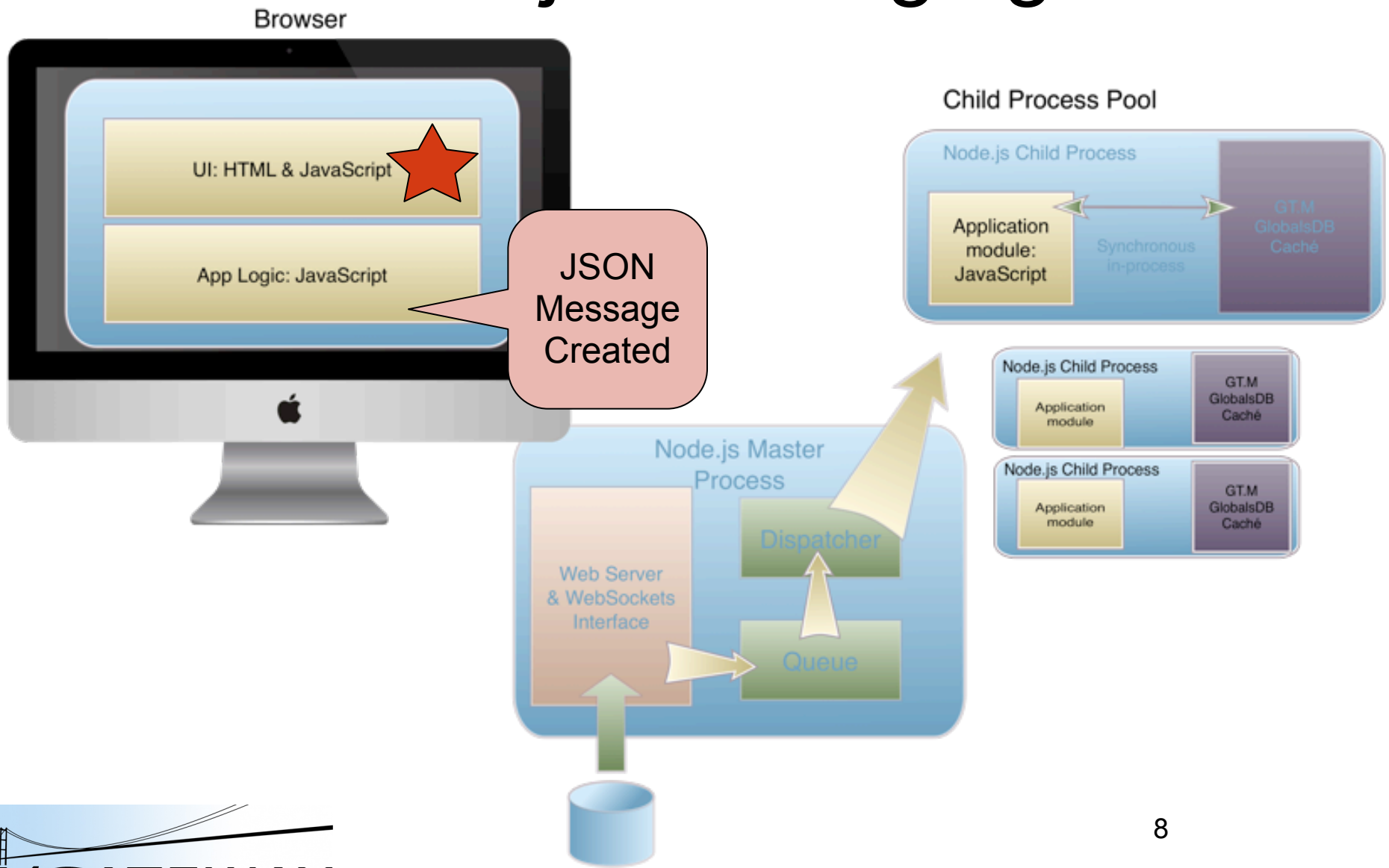


M/GATEWAY DEVELOPMENTS

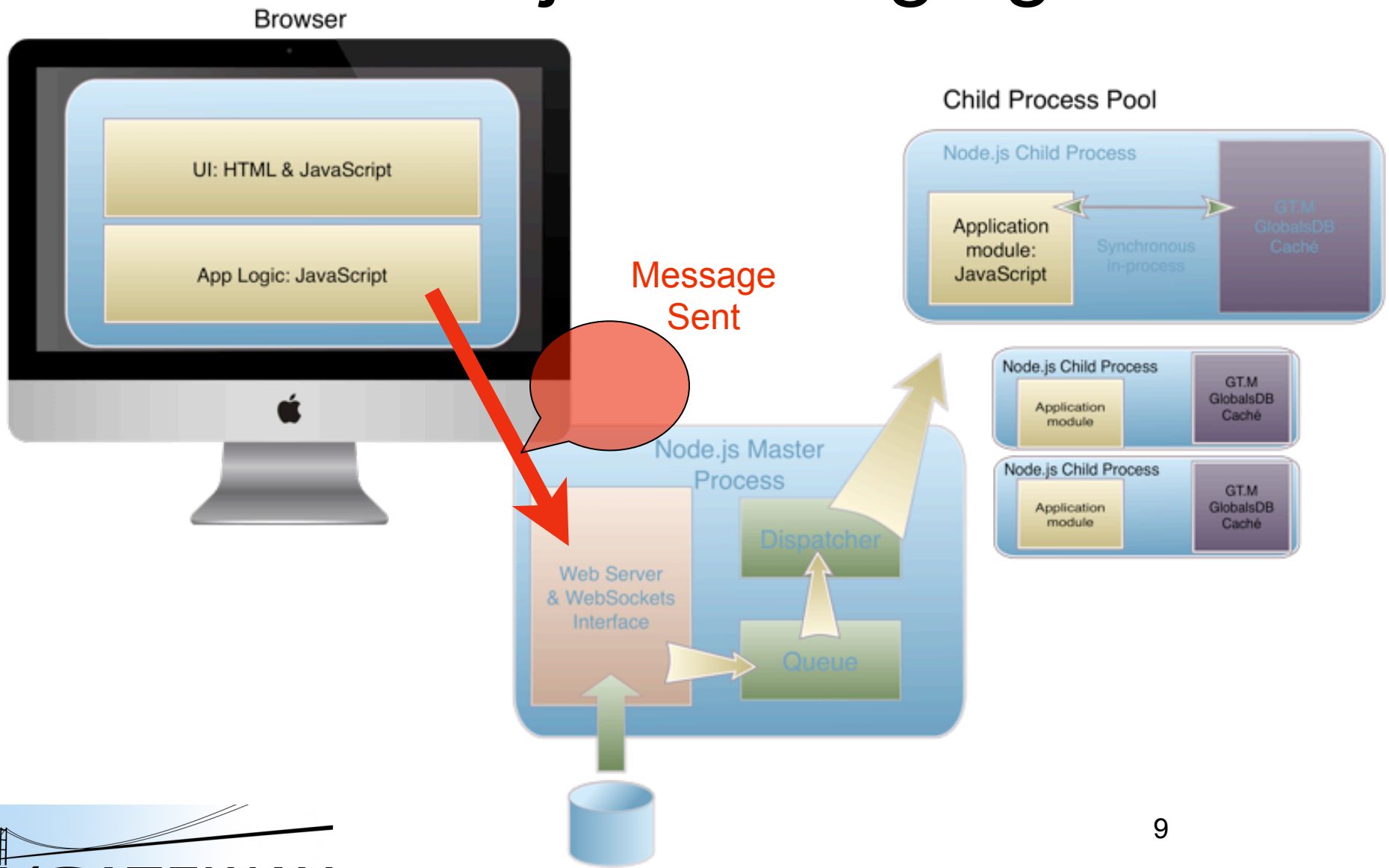
EWD.js Messaging



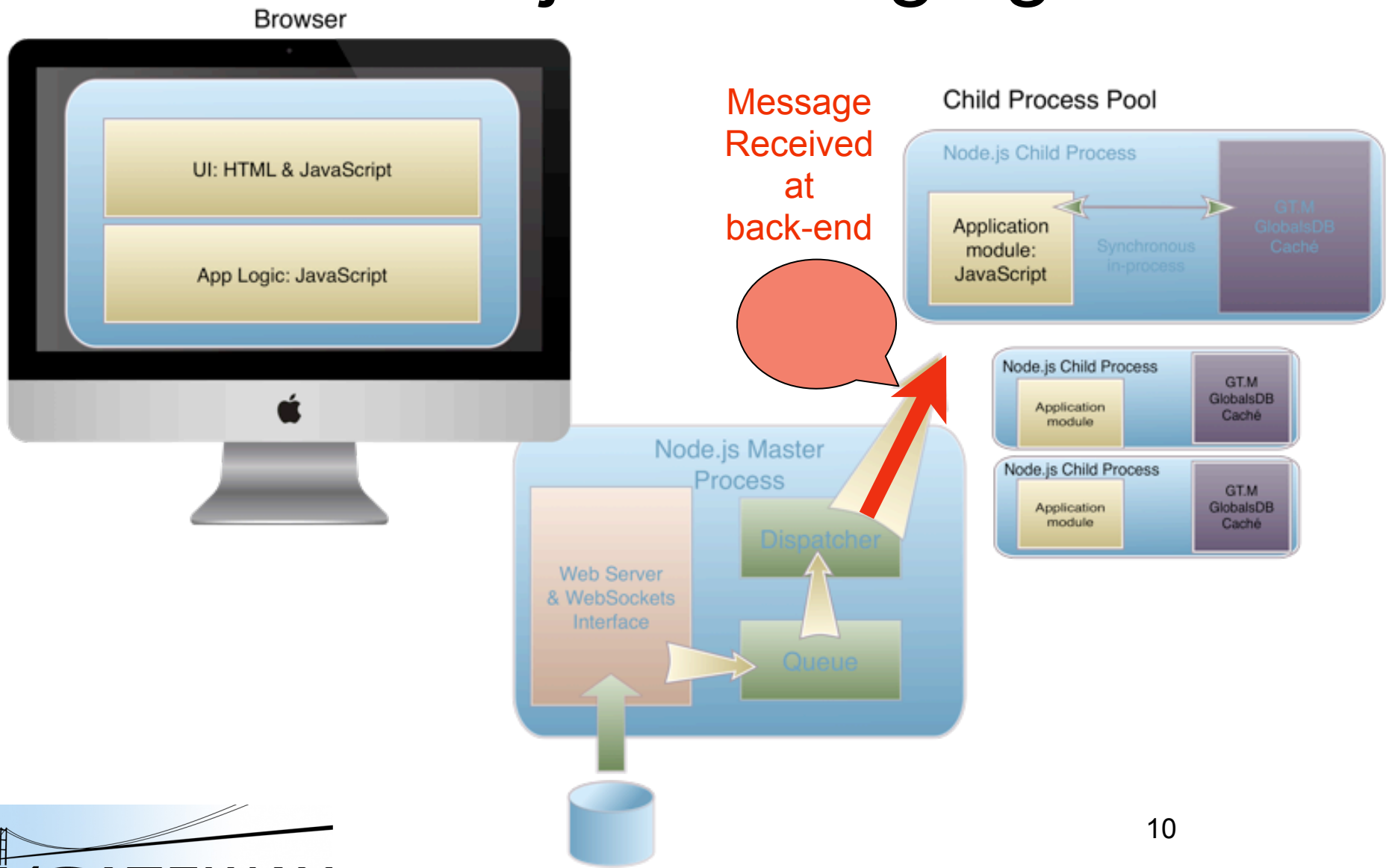
EWD.js Messaging



EWD.js Messaging

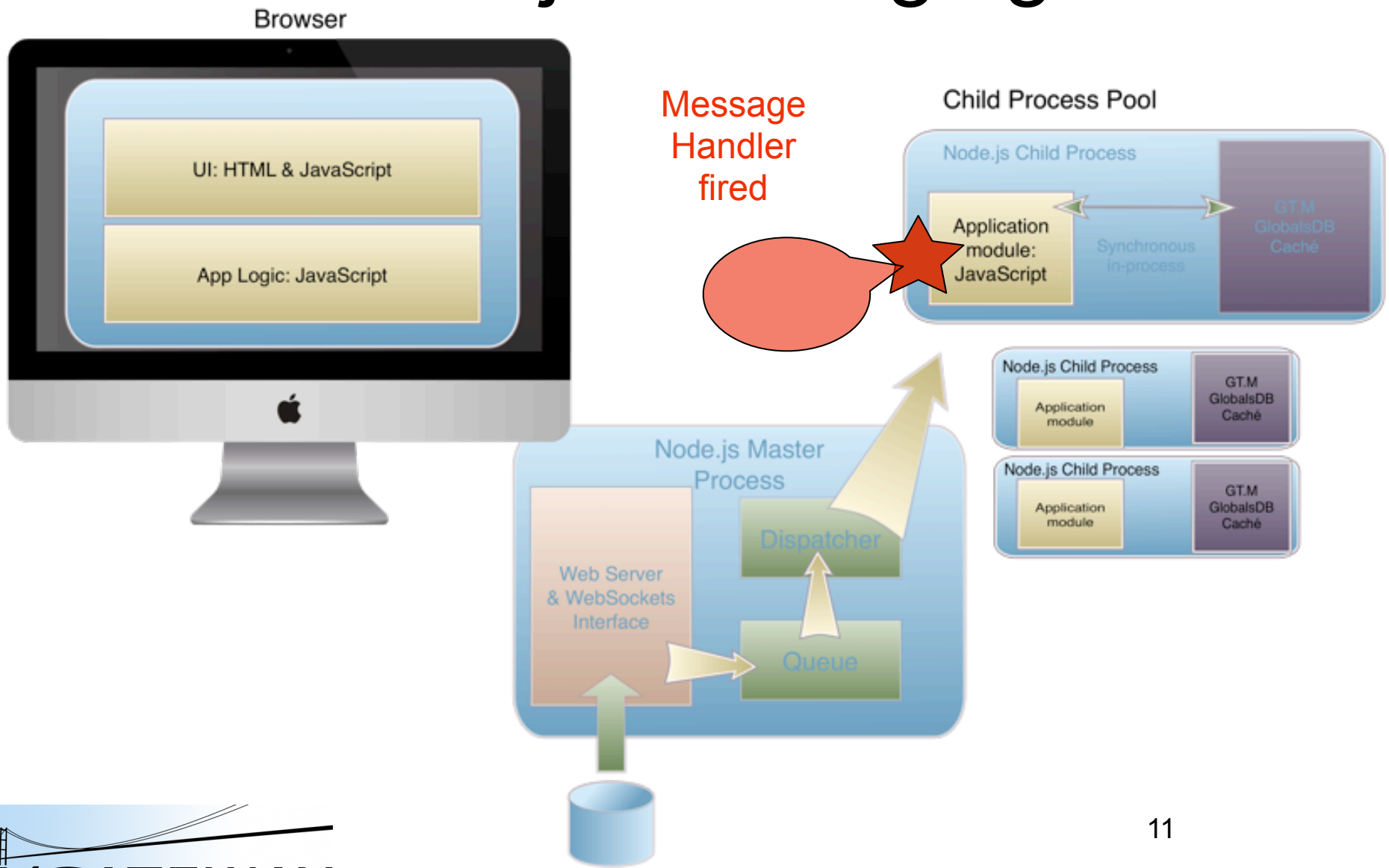


EWD.js Messaging



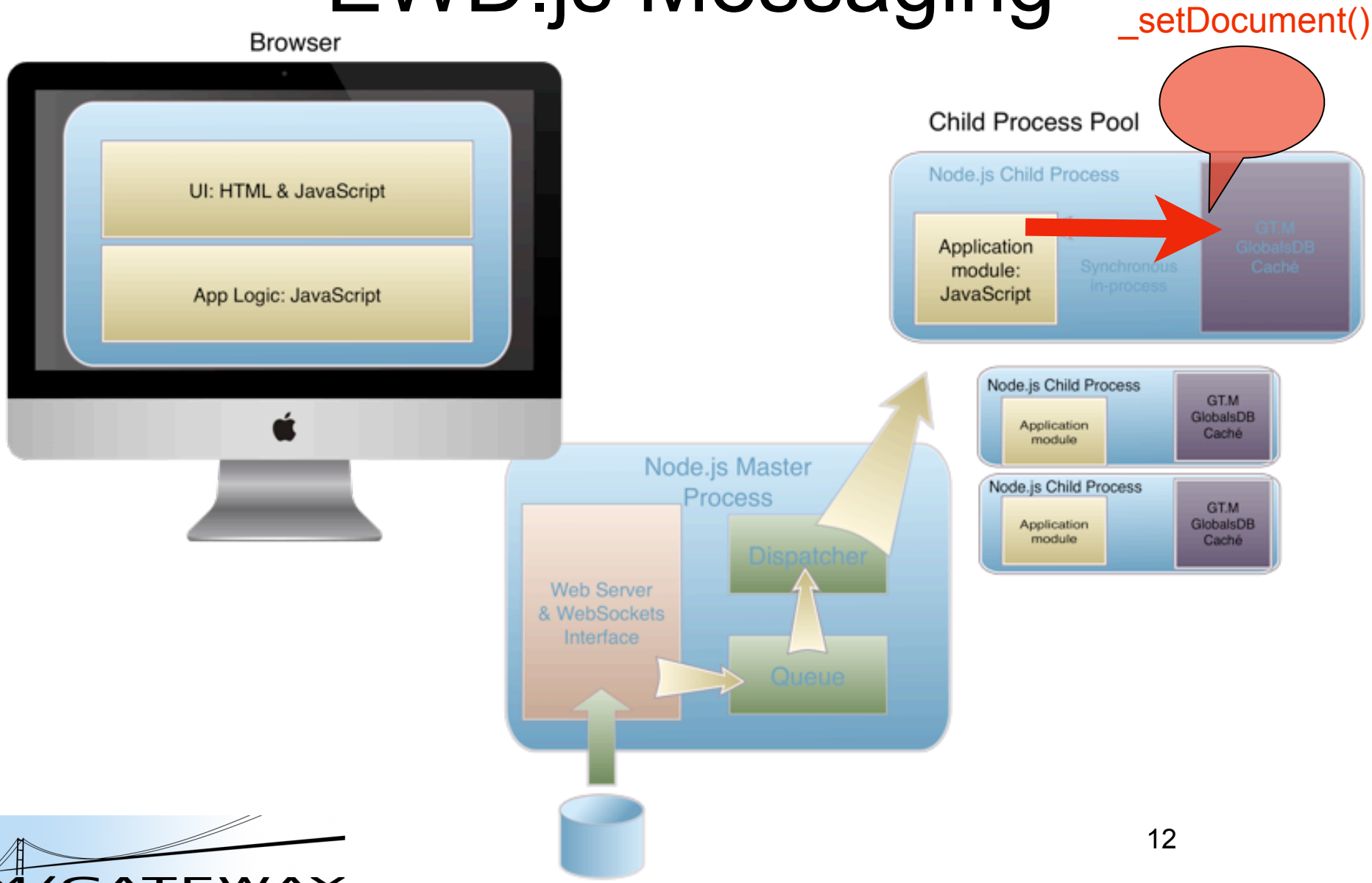
M/GATEWAY DEVELOPMENTS

EWD.js Messaging



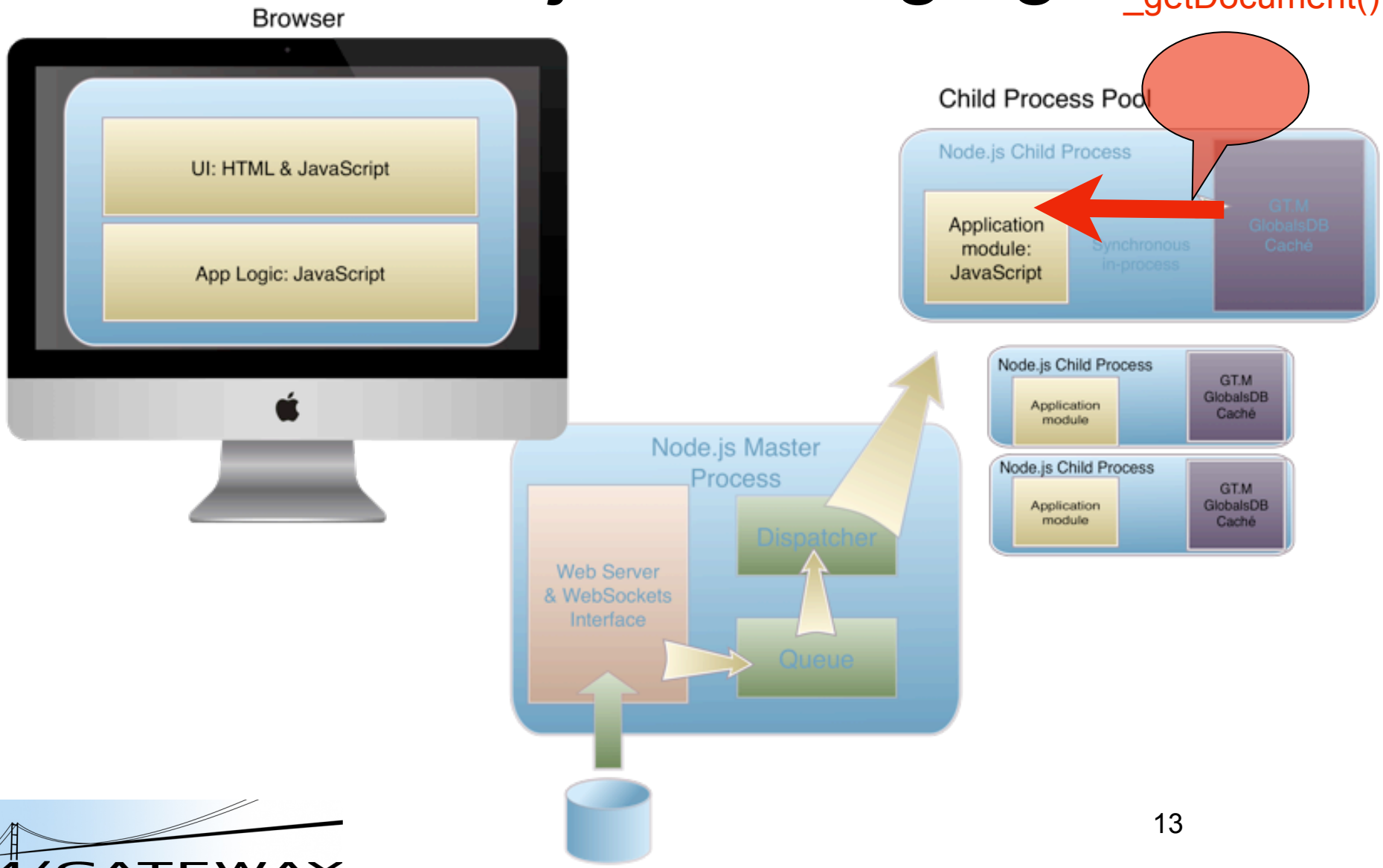
M/GATEWAY DEVELOPMENTS

EWD.js Messaging



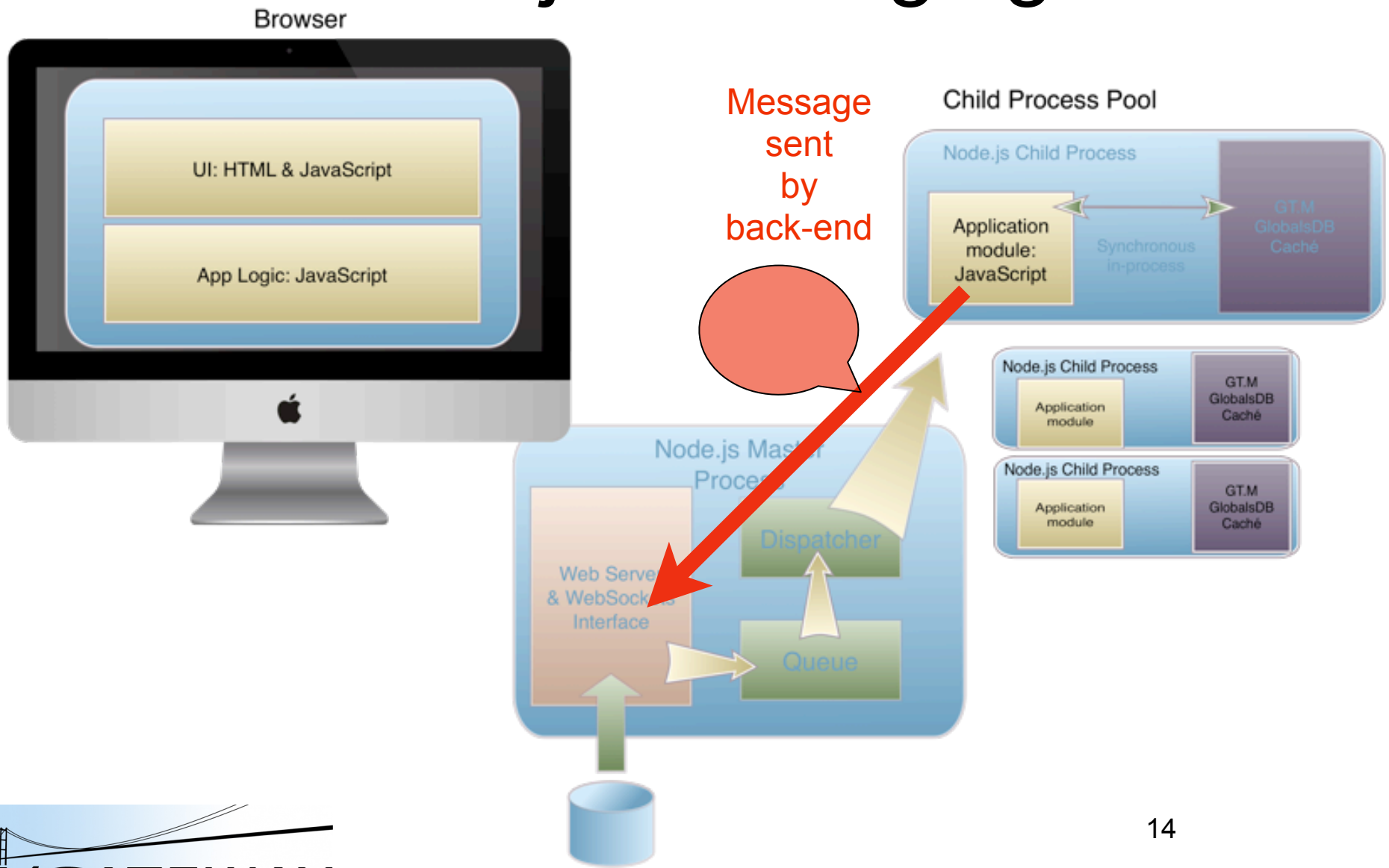
M/GATEWAY DEVELOPMENTS

EWD.js Messaging



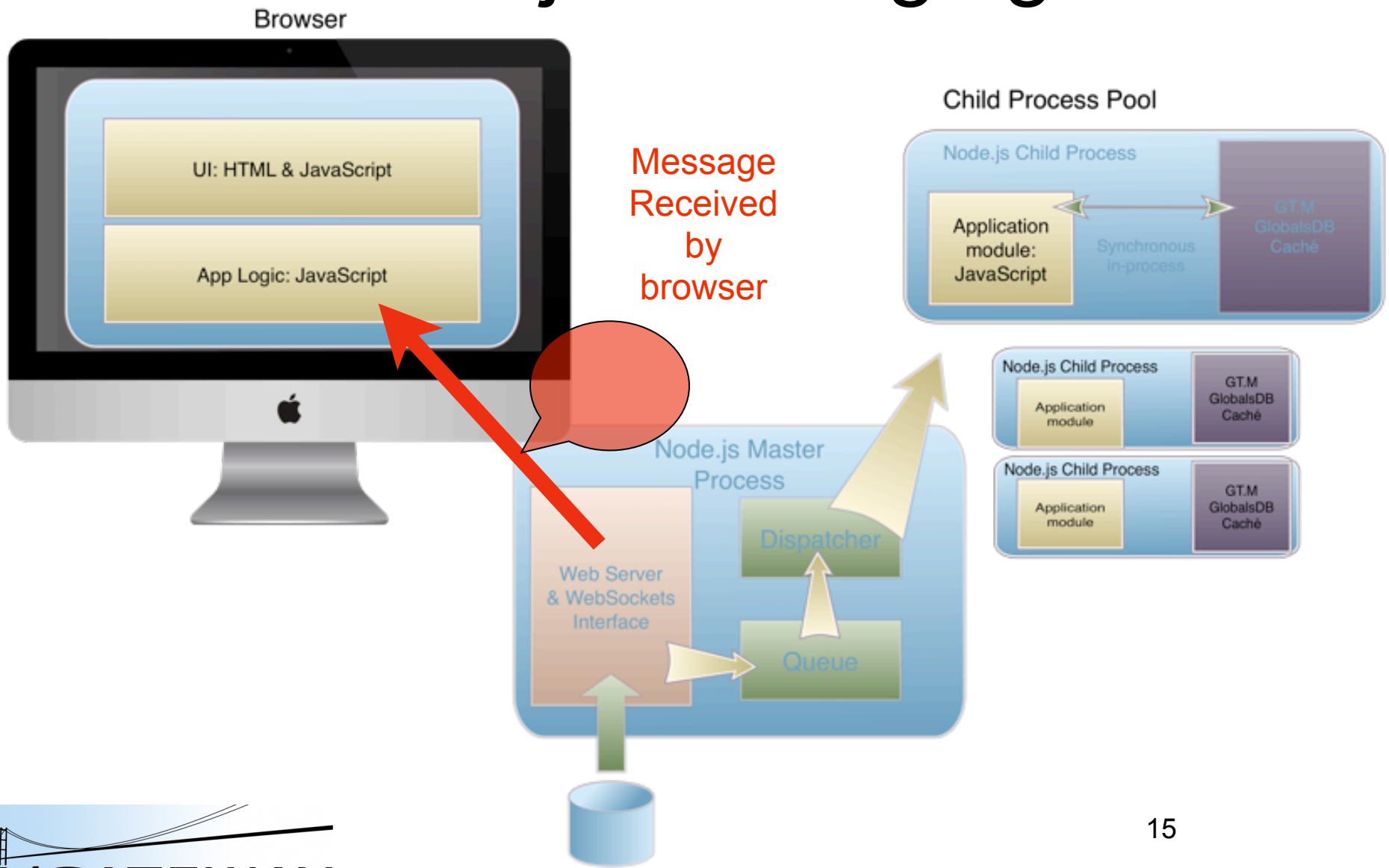
M/GATEWAY DEVELOPMENTS

EWD.js Messaging



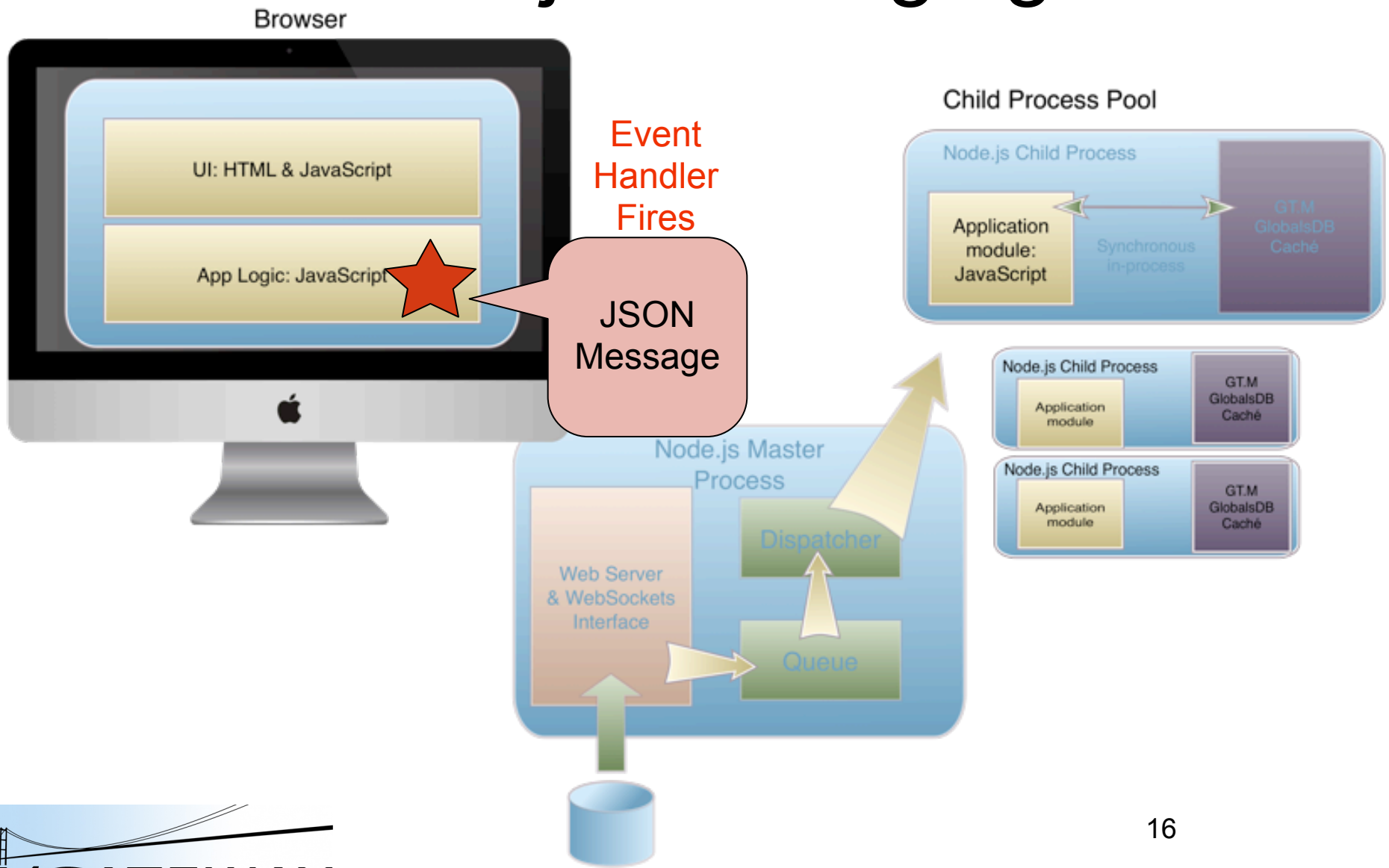
M/GATEWAY DEVELOPMENTS

EWD.js Messaging



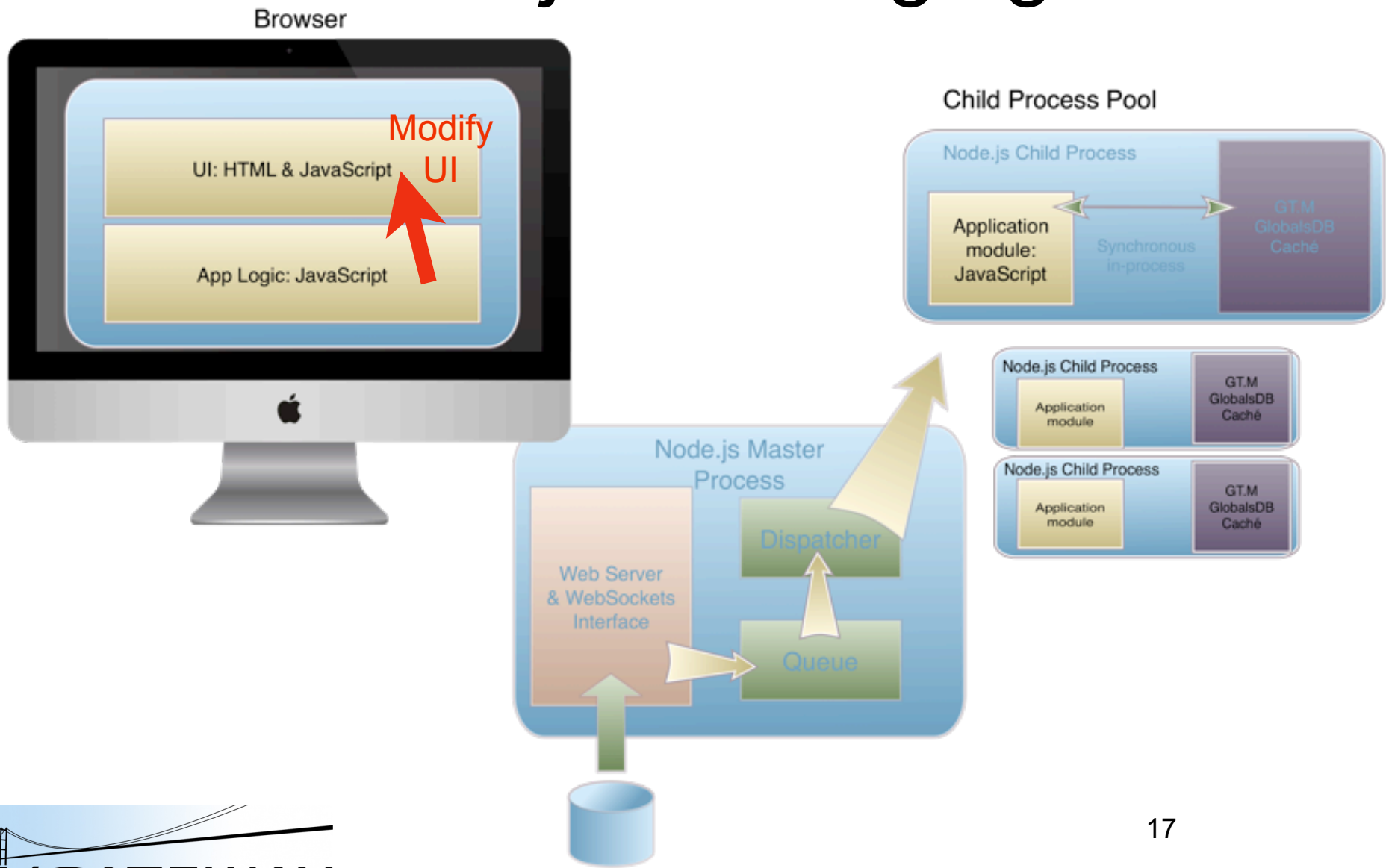
M/GATEWAY DEVELOPMENTS

EWD.js Messaging



M/GATEWAY DEVELOPMENTS

EWD.js Messaging



M/GATEWAY DEVELOPMENTS

Send message from browser

```
EWD.sockets.sendMessage({
  type: "sendHelloWorld",
  params: {
    text: 'Hello World!',
    sender: 'Rob',
    date: new Date().toUTCString()
  }
});
```

Back-end Module

```
module.exports = {
  onMessage:  {
    sendHelloWorld: function(payload, ewd) {
      // do whatever is required with payload
      return {received: true}; // or any JSON
    }
  }
};
```

Back-end Module

```
module.exports = {
  onMessage:  {
    sendHelloWorld: function(payload, ewd) {
      // eg save payload to JSON store
      var store = new ewd.mumps.GlobalNode('myStore', [1]);
      store._setDocument(payload);
      return {savedInto: 'myStore'};
    }
  }
};
```

Browser-side Handler

```
onMessage {
  sendHelloWorld: function(msgObject) {
    // do whatever you need to do with the incoming message, eg:
    var text = 'Your message was successfully saved into ' +
       messageObj.message.savedInto;
    toastr.clear();
    toastr.info(text);
  }
};
```

No Polling!

- With WebSockets, the back-end can send a message *at any time* to:
  - a specific browser
  - all browsers running a specific EWD.js application
  - all currently-connected browsers

Back-end can send a message

```
var savedMsg = new ewd.mumps.GlobalNode('myMessage', []);

ewd.sendWebSocketMsg({
  type: 'savedMessage',
  message: savedMsg._getDocument()
});
```

Node.js, but Synchronous Code!

- Your back-end code runs in a Node.js Child Process
- An EWD.js Child Process only handles a single request at a time so it can afford to use blocking, synchronous I/O
- If all Child Processes are busy, incoming requests are queued in the fully asynchronous master Node.js process
- Child Process pool size is configurable to match demand

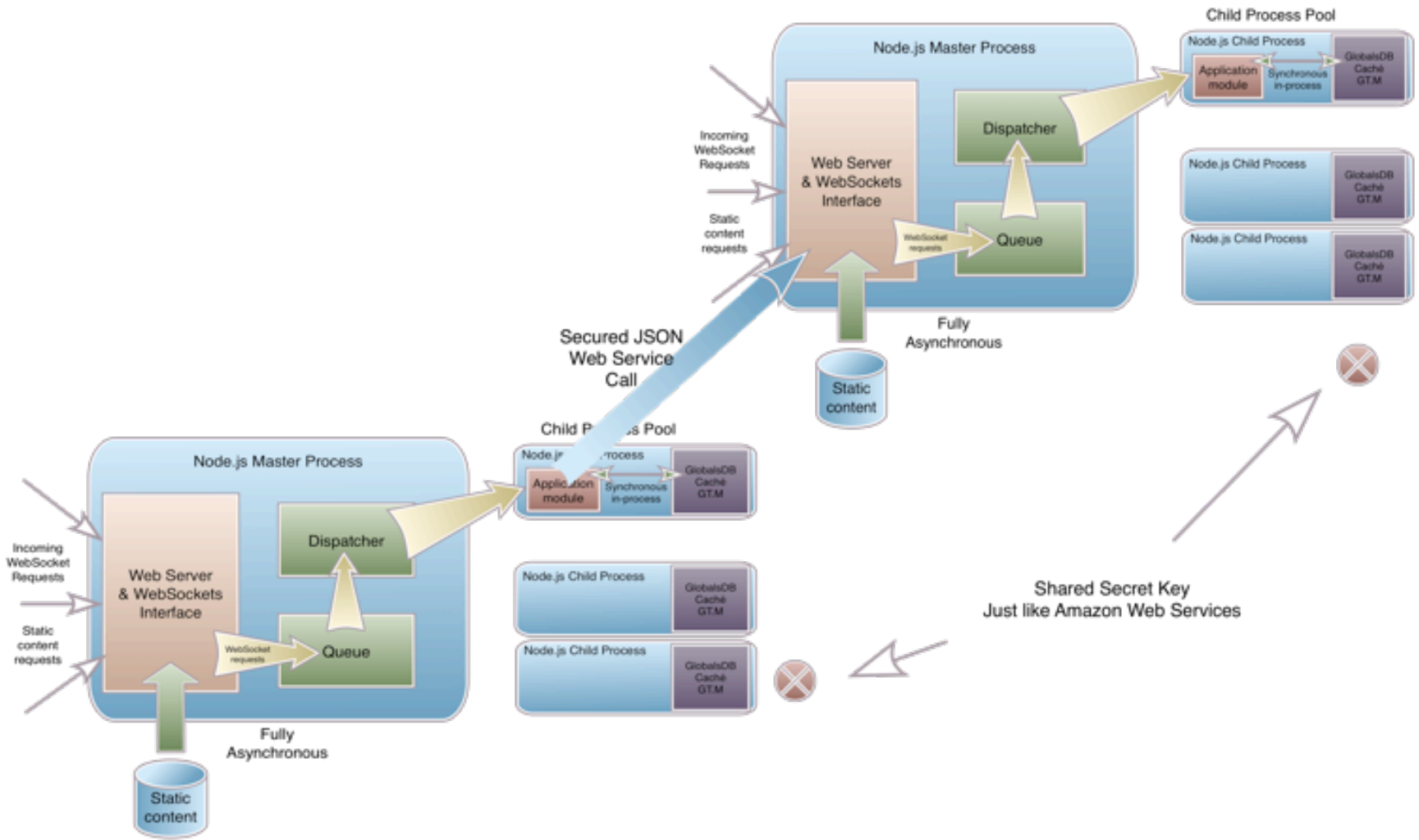
Node.js, but Synchronous Code!

- Database connector interfaces used by EWD.js are synchronous
- Your back-end JavaScript logic can be written in easy-to-maintain, intuitive synchronous style
- No “callback hell” or pseudo-synchronous syntax
- Result: faster development, maintainable JavaScript logic

Built-in secured Web Services

- Any back-end EWD.js JavaScript method can also be exposed as a JSON Web Service
- Access is automatically secured
  - HMAC-SHA256 digital signatures required for every HTTP request
  - The same security used by Amazon Web Services
- Lightweight peer-to-peer access between EWD.js systems

Secured Linked Systems



M/GATEWAY DEVELOPMENTS

EWD.js requirements

- Ideally browsers that support HTML5 WebSockets
  - however, EWD.js uses Node.js socket.io library
    - emulates websockets using other techniques if not available
    - even works with old versions of Internet Explorer!