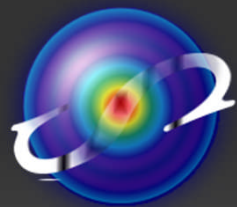
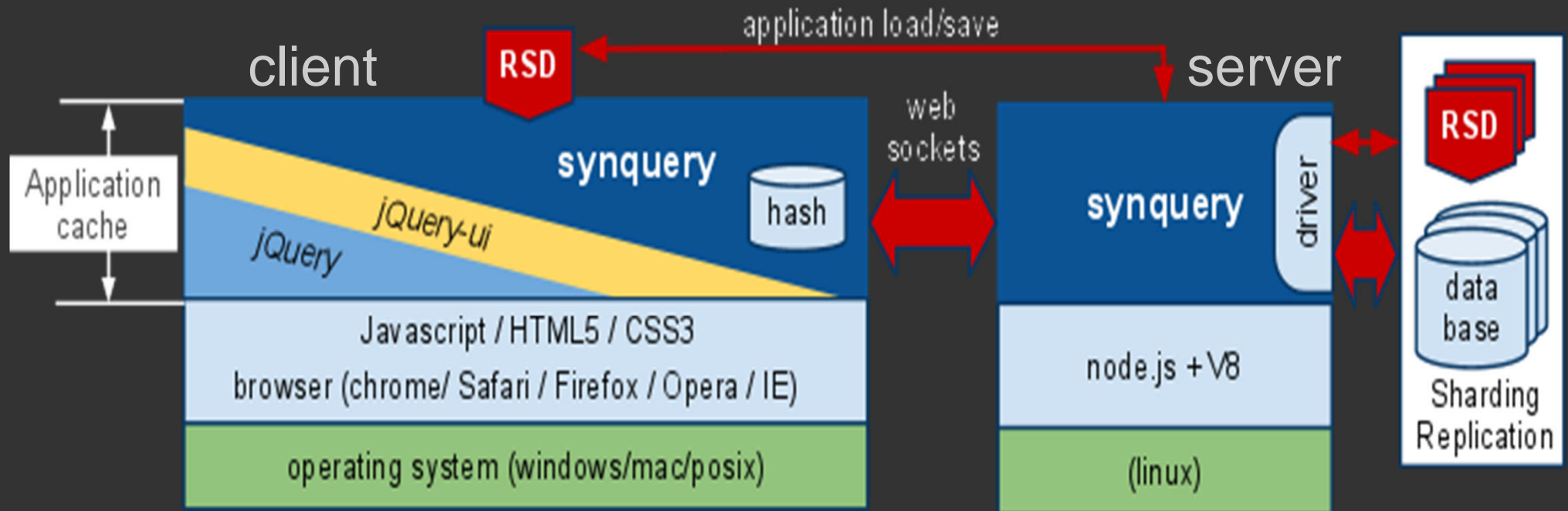


*synquery*TM web-platform

テクノロジー概要紹介



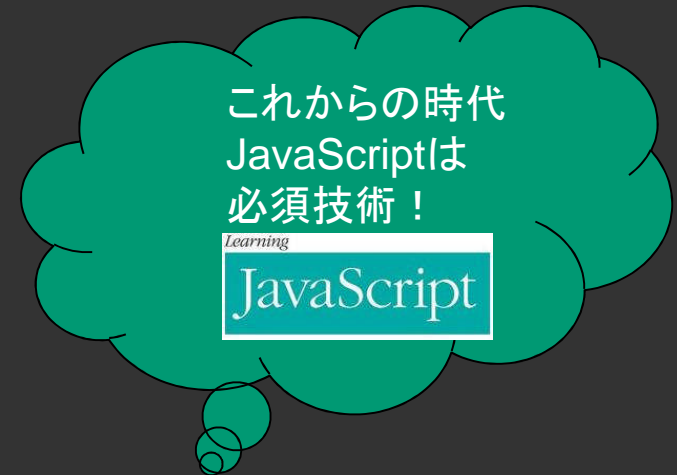
プラットフォームとしての *synquery*TM とは



- *synquery* はコンフィギュラブルなweb-system用プラットフォーム
- 簡単なスクリプト(**RSD**)をデプロイ= systemコンフィギュレーション
- *synquery* クライアントとサーバ間をイベントループ&WEBソケットでシームレスな接続
- *synquery*のコアシステム(RSDのインタープリタ)はブラウザー上に常駐
(アプリケーションキャッシュ内)
- サーバは単にデータとRSD を保持するだけなので、サーバ負荷がかからない
(すべての処理はクライアント上)

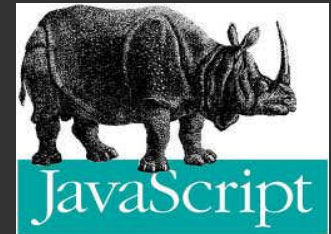
RSDとは

- **RSD (Relational Structure Description)** は *synquery* のコアエンジンが認識する記述
 - 言語はJavaScript (jQuery)
- RSDで記載される内容としては、
 - 各テーブル、フィールドの定義
 - 画面レイアウト
 - 実行(イベント)処理(必要に応じて)
- テキストエディタ、ブラウザ上で編集可能
 - RSDをcopy & pasteすることで再利用が可能



JavaScript?

- WEBブラウザのネイティブ言語
- 厳格、強固でオブジェクト指向でありながら、フレキシブル
apply() / call() / eval() etc
- コンパイラが最速 (内部ブラウザをコンパイル)
- サーバーサポート(同じ言語の利用可能)



jQuery?

- JavaScript 言語では業界標準
- 短い記述で高い機能を実現
- ブラウザとOS間の互換性が高い
- モバイルをサポート (jQuery-mobile)



synquery™ による技術者のパラダイムシフト

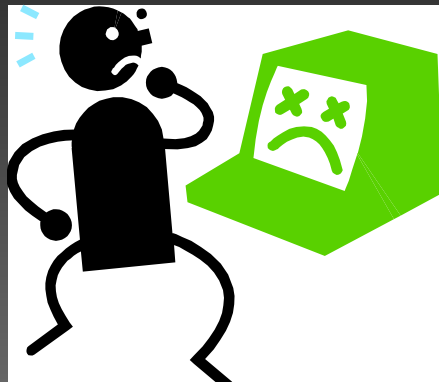
インフラ準備・・・
うわー高い



仕様書作成・・・面倒くさー
コーディング・・・やっときたか、、、
テスト・・・自分が作ってないところもテスト？



DB設計・・・大変、時間がかかる



- インフラ準備→ 不要
- 画面設計、DB設計、仕様書は自動生成
- だから、画面の表現や構成は開発者の思うまま



paradigm shift

従来のシステム開発手法との比較

- ・ 従来
 - インフラ準備 (DB、言語決定、HW)
 - DB設計
 - 仕様書作成
 - コーディング
 - テスト
 - 運用 (保守)
- ・ synquery
 - インフラ準備 → 不要
 - RSDの記述・画面設計、DB設計、仕様書は自動生成
 - ・ 画面の表現や構成は開発者の思うまま
 - ・ プレゼンテーション層
 - コーディング (特別なロジックの追加) ・任意
 - テスト ・自分の作った部分のみ!

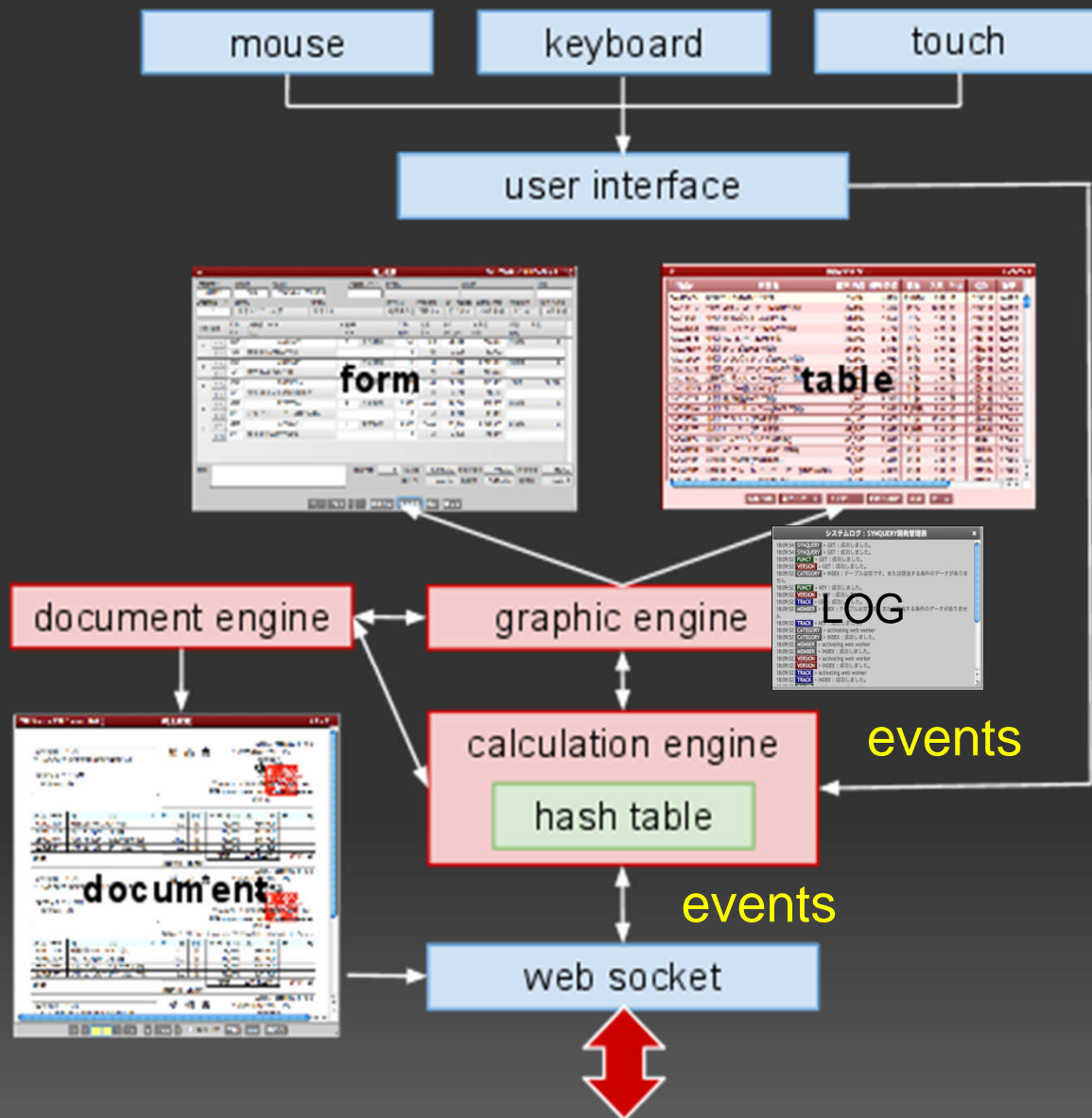
システム開発期間の大幅な短縮

4ヶ月かかったものが、1ヶ月でできるようになれば、4倍の仕事ができる！ (再利用性を考えれば、さらに加速可能！)

synquery™ の機能

- NoSQL: データベース構造に拘束されないシステムアーキテクチャ
 - SQLがないと不安ですか？
 - データの構造さえ、理解していればデータの取得はより簡単に (Key-Valueで取得できる)制御できます！
 - アプリケーションはサーバに同期されたクライアント上のHashを参照する
 - データの変更は、即時に全クライアントにブロードキャストされる
- RSDの記述に基づき画面を自動生成
- RSDの記述に基づきデータベースを自動生成
- 格納されたデータは帳票として出力可能 (PDF/SVG/HTML等).
- i18n コンプライアンス準拠 (マルチ言語対応)
- 設計者は短い**RSD**スクリプトを記述するだけでアプリケーションを開発できる

synquery™ Architecture of client system



サーバー & 他のクライアントへ



- event driven architecture
- internal hash orient
- strong engines
 - calculation
 - graphic
 - documentation
- web socket interface
- improved connectivity
- broadcasting

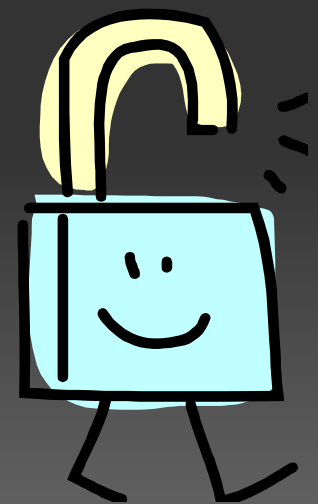
Summary: *synquery*の特長

■ パフォーマンスとクオリティ

- ✓ WEB上でリアルタイムにシステム開発/拡張
- ✓ 機能/データ追加しても現状のシステムを保持しながら拡張可能
- ✓ 従来のクライアント・サーバーシステムと同等な操作性
- ✓ 画面の描画や表現は開発者の思うままに作成可能

■ セキュリティ

- ✓ ターゲットポートに対してSSLによる通信インターフェースを用意
- ✓ クライアントのハードディスク内にデータが残らない
- ✓ データサイトは分散可能: 機密データは社内に保存



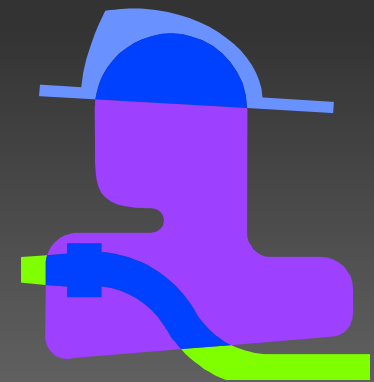
Summary: *synquery*の特長 つづき

■ Time to service / Time to market

- ✓ RSD記述による高い生産性
- ✓ 作成したRSDは再利用可能

■ 容易なメンテナンス性

- ✓ システムの仕様書を自動生成
- ✓ フィールドの追加および削除は瞬時に反映



最新のテクノロジーで最大の利益を

*synquery*開発には

JavaScript, jQuery,

HTML5, CSS3,

node.js, NoSQL, ...

を活用

node.js?

- イベントループ & web socket (send と broadcast)
- Google V8 エンジンで最速に強化されている
- 本システムと同じ言語

The logo for Node.js, featuring the word "node" in white and "JS" in a green-to-yellow gradient.

NoSQL = mongo?

- 最速の gets & puts 及び no-JOIN ステートメント (key-value and index)
- クラウドのような巨大システム上でスケーラビリティ & 信頼性 (sharding / replication)
- 適度な不可分性 (原子性) と一貫性
- map-reduce 機能をサポート

The logo for MongoDB, featuring a green leaf icon to the left of the text "mongoDB" in a gold-to-brown gradient. Below the text is the JSON document type notation: {name: "mongo", type: "db"}.

mongoDB における *synquery*TM の ACID

{name: "mongo", type: "db"}

Atomicity (原子性)

*synquery*TM はトランザクションにおいて commit と rollback の実行可能

Consistency (一貫性)

*synquery*TM はトランザクションにおいてエラーが生じた時、自動的に rollback

Isolation (独立性)

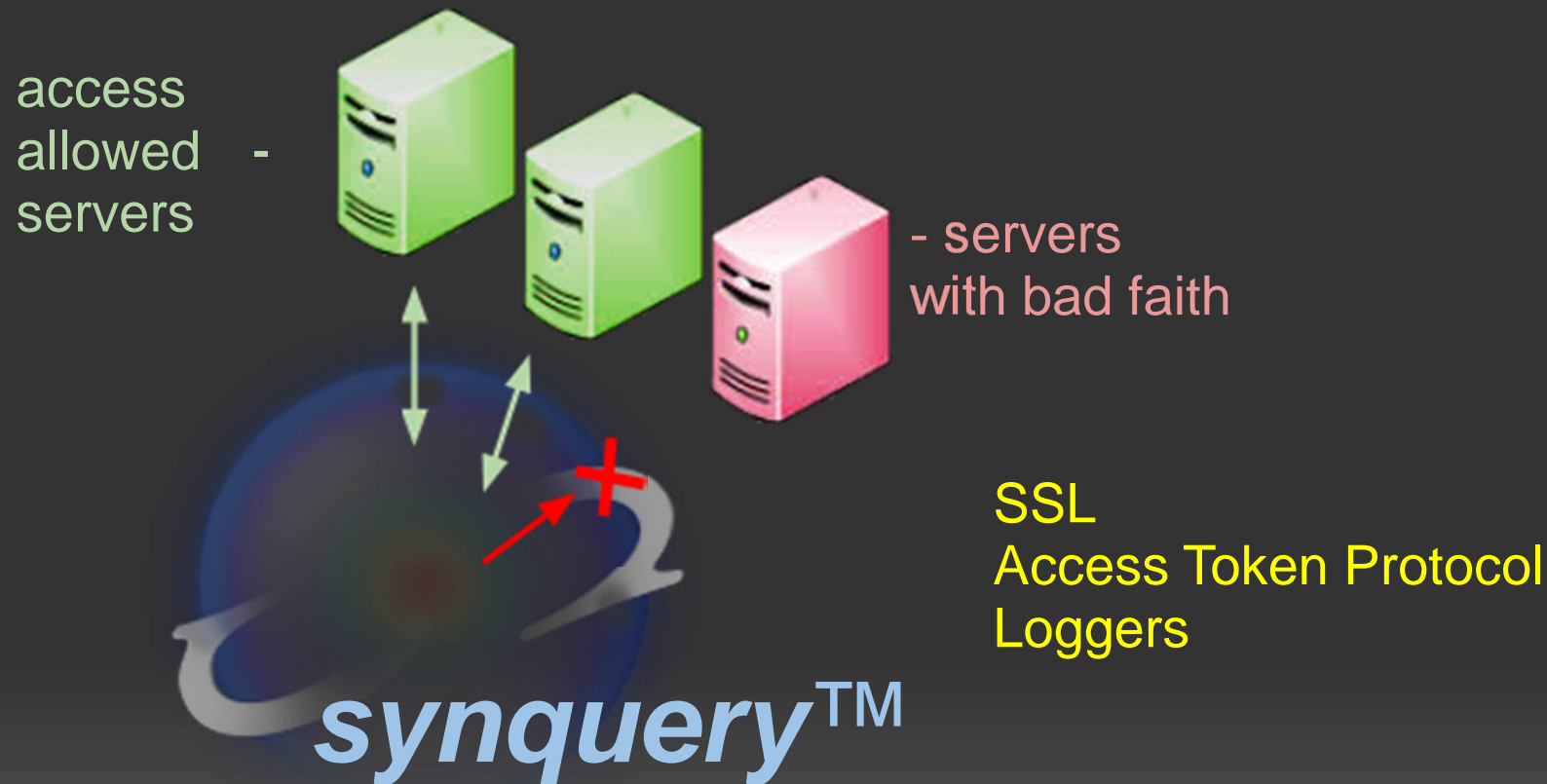
*synquery*TM はアイソレーションレベルでリアルタイムWEBコミュニケーションの **READ UNCOMMITTED** をサポート。加えて、分析計算の **SERIALIZABLE** をサポート。

Durability (耐久性)

追加の *synquery*TM を実行して、mongoDB の **'Journaling' 機能 (ファイル更新履歴の自動バックアップ)** を実現

synquery™のネットワークセキュリティ

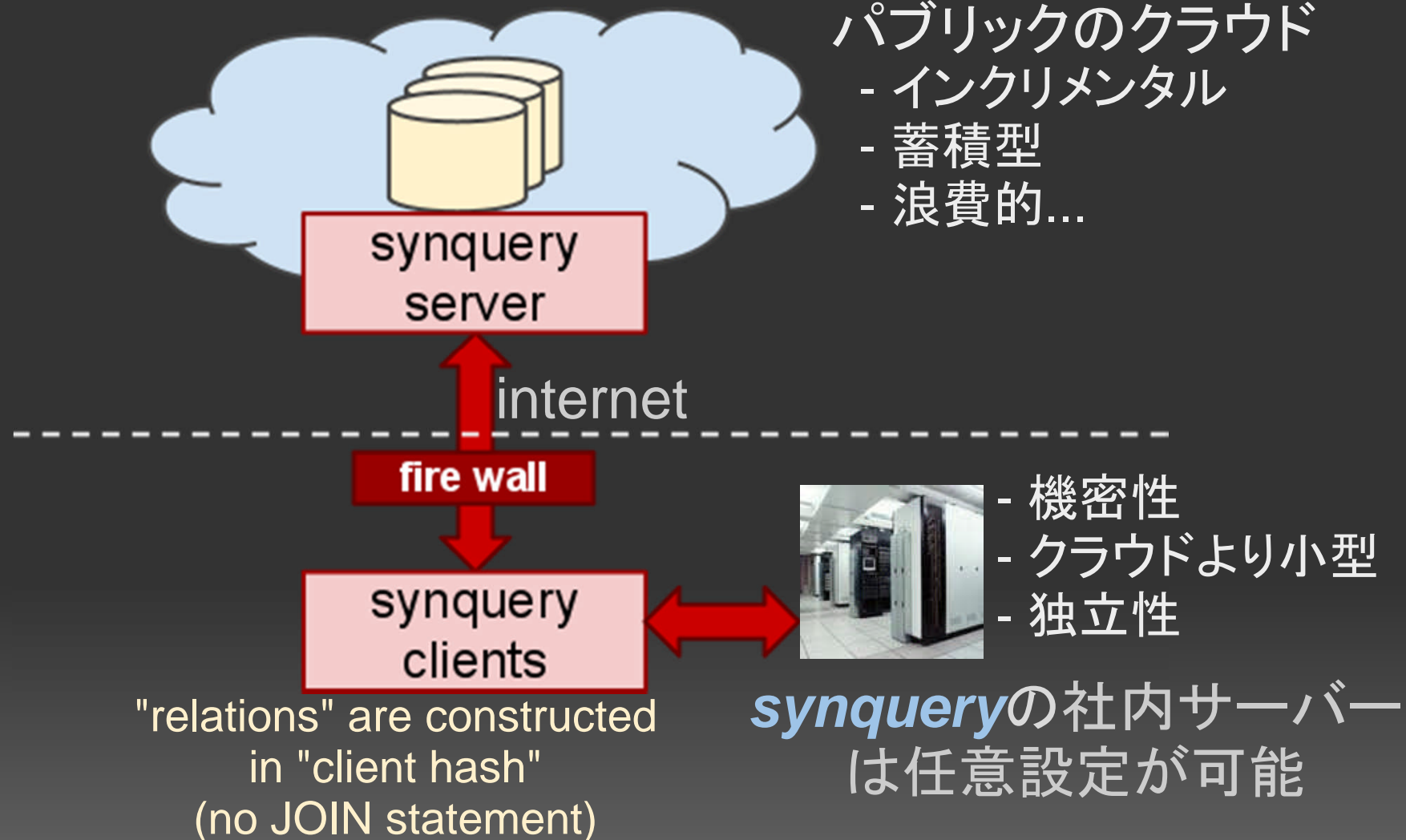
synquery™の‘JavaScriptのネイティブ・オブジェクト・ラッパー’機能はユーザと開発者の不正アクセスを防ぐ



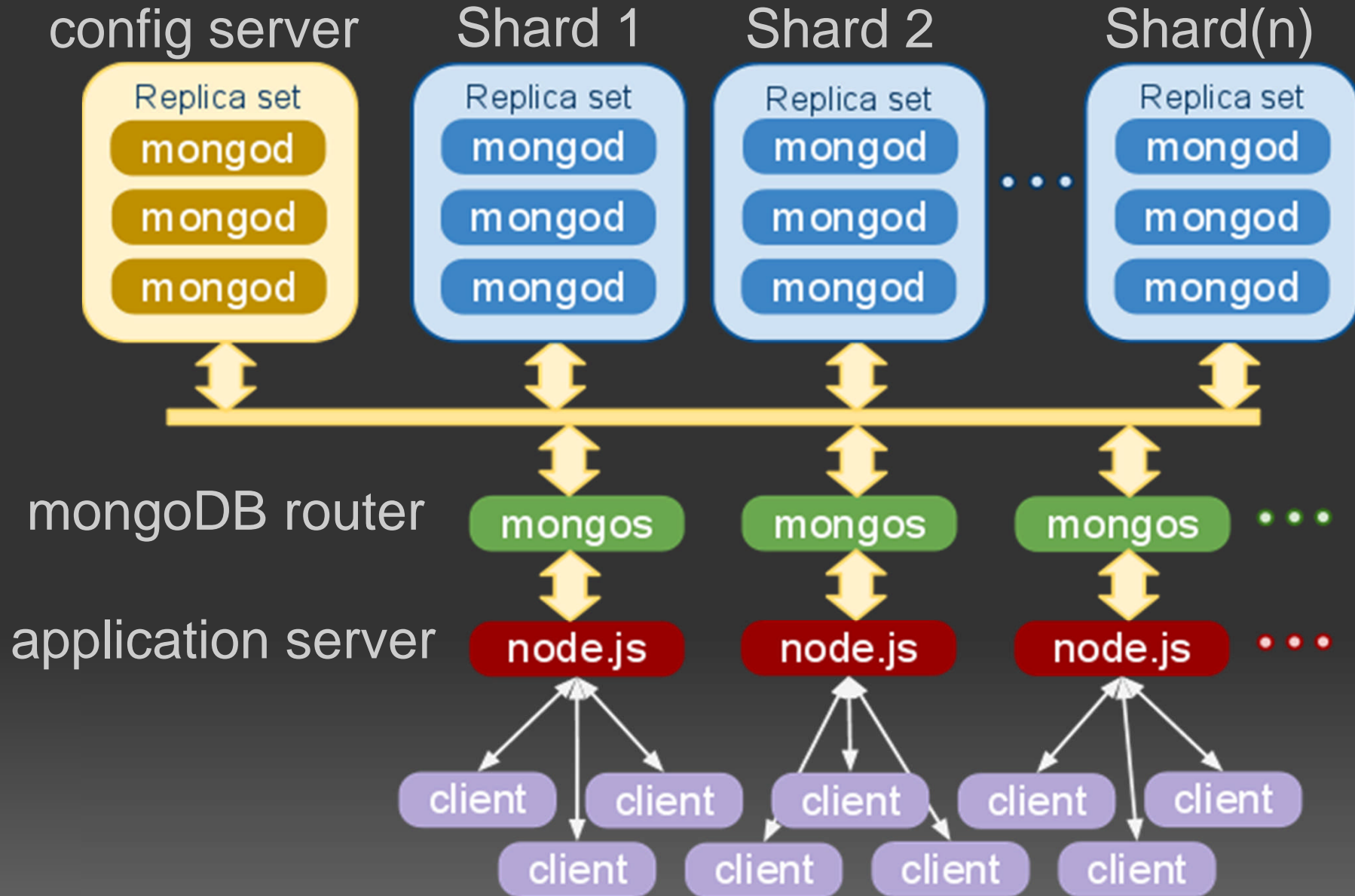
No secret deal with synquery™ through other servers.

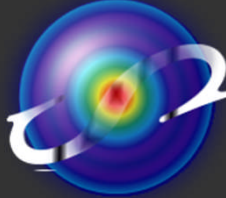
パブリックのクラウドか or プライベートのクラウドか?

=> *synquery* はカプセル化 (Key-Value) & 分散データのサイト・アーキテクチャにより“**Hybrid クラウド**” 環境を提供



スケールアウトとフェイルオーバー



Come on with  *synquery!*

WYSIWYG!

What You "Script" Is What You Get!