

このページでは D3.js 3.0 の新機能について解説します。2.x から 3.0 への移行方法については[3.0へのアップグレード](#)をご覧ください。

- [English](#)

参考: [地図投影法の一覧 - Wikipedia](#)

Geo

逆子午線切断 	三軸回転 	ヴィンケル図法の回転 	正距円筒図法の回転 
クリッピング 	可変型リサンプリング 	キャンバス・レンダリング 	衛星と経緯度線図 
投影法変化 (へんげ) 	統計地図 	ラスター再投影 	



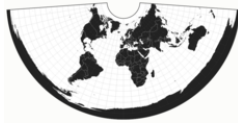




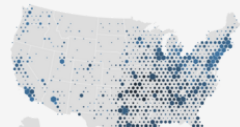



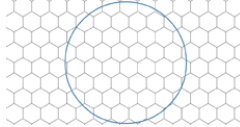
D3 に強力な新しい地理投影システムが追加されました！前のバージョンの D3 が投影を点関数として解釈するだけだったのに対し、D3 3.0 では投影を完全な幾何変換としてモデル化しています。これにより D3 は、直線を曲線に投影する際、線を細分化して不要な投影生成物を除去するために、設定可能な[可変型リサンプリング](#)を用いるようになりました。例として[ラリヴィ図法](#)や[ヴァン・デル・グリテン図法](#)でスムーズに描かれた南極大陸をご覧ください。線やポリゴンが逆子午線（訳注：中央にとった子午線の180度反対側の経度を通る子午線）を横切るとき、D3 は事前にシェイプファイルを切断するのではなく[ジオメトリを切断](#)します。また、すべての投影法が[小円](#)と[クリッピング](#)、それに[三軸回転](#)に対応するようになりました。

内部的には D3 はストリーミング幾何変換（`d3.geo.stream`）を用い、作業用オブジェクトの生成を避けることでメモリ使用量を削減しています。この設計で[キャンバス直接レンダリング](#)性能も劇的に向上しました。ストリームは、表示されている図形と、投影範囲や重心、境界線がずれてしまわないための計算にも使われ、さらに[縮尺に応じた図形の単純化](#)のための幾何計算も可能にしています。

`d3.geo` パッケージには便利なコンポーネントが多数追加されました。例えば[graticule \(経緯度線図\)](#) は地球の全体図をレンダリングするための線と球体の格子図を表示します。`d3.geo.circle` クラスは小円と大円レンダリングにも対応し、[テイソーの指示楕円](#)の近似計算が可能になりました。

新しい d3.geo.interpolate で大きな弧の球面座標補間も簡単にできるようになりました。さらに球面範囲、重心、境界を正確に計算するための新しい関数群も追加されています。

プラグイン

ペイルス・クインカン シャル図法 	ワグナー第6図法 	正距円錐図法 	モルワイデ図法 
Sinu・モルワイデ図法 	グード図法 	ヴァン・デル・グリテン図法 	d3.hexbin 
d3.geo.tile 	TopoJSON 	地図配色 	スムーズ・ズーミング 

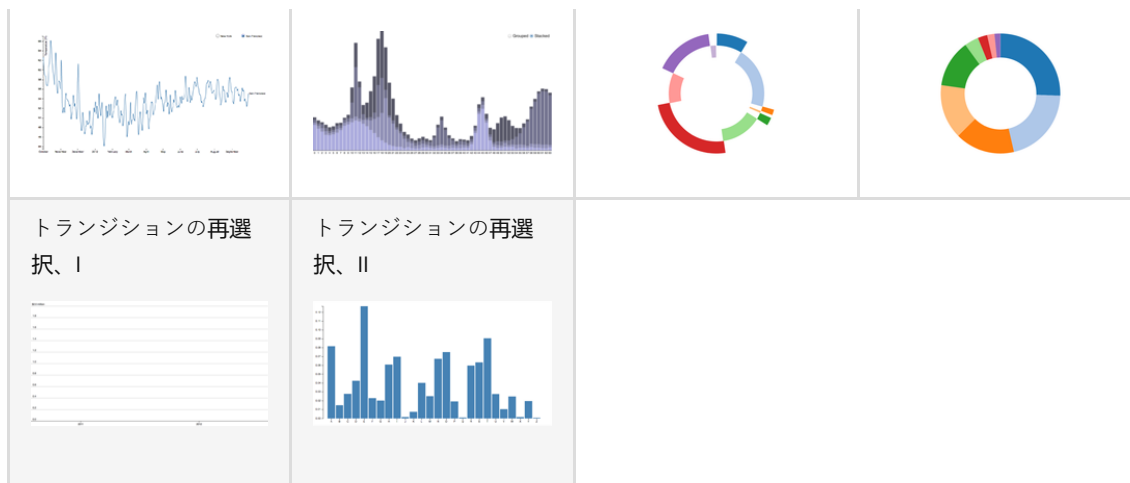
[地図投影法](#) プラグインは拡張され、対応する投影法は 40 を越えました。d3.geo.projection や d3.geo.projectionMutator を使って独自の投影法を作ったり、The [d3.geo.tile](#) プラグインを使って自分の地図のなかに簡単に256 x 256 のラスタータイルを挿入することもできます。

d3.behavior.zoom と組み合わせれば[滑らかに動く地図](#)もすばやく作れます。[d3.hexbin](#) と [d3.interpolateZoom](#) など [地図製作](#)に便利なプラグインです。

[TopoJSON](#) は GeoJSON を拡張したフォーマットです。典型的には 80-90% ファイルサイズが縮小します。また TopoJSON はトポロジーもエンコードしており、（ MapShaper のような）トポロジーを維持したままの図形簡略化、[地図配色](#)、[統計地図](#)、[境界メッシュ計算](#)その他多くのことが可能になりました。TopoJSON は自体は D3 のプラグインではなく、TopoJSON から GeoJSON へ変換するための単体の JavaScript ライブラリに過ぎません。しかし将来の d3.geo のリリースで TopoJSON にネイティブに対応するかもしれません。

トランジション（画面遷移）

チェインしたトランジ ション、I	チェインしたトランジ ション、II	チェインしたトランジ ション、III	チェインしたトランジ ション、IV

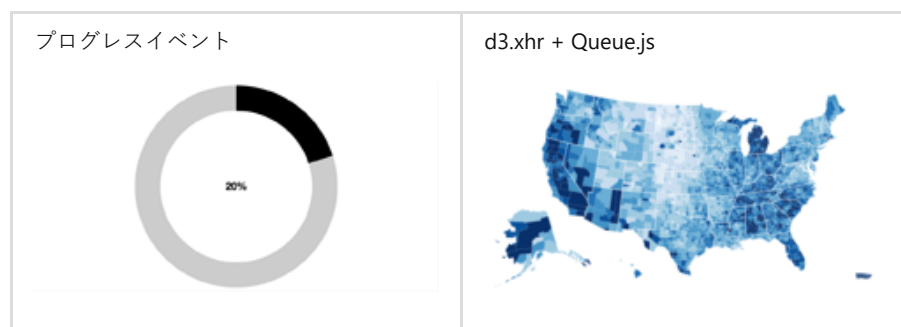


3.0 では、従来のデータとセレクションとの結合方法と似た形で、トゥイーンやその他のトランジションの状態を DOM と結合できるようになりました。これは結合させたその DOM から、定義されたトランジションを再選択し、その DOM を変更できるようになるということです。例えば[カスタマイズされた 軸](#)などがその例です。こうした DOM 要素はJavaScript コンソールを使って調べることができるため容易にデバッグできます。

トランジションを使いやすくするため、値関数は `transition.attr` と `transition.style` を使って即時評価されるようになりました。従来のようにトランジションの開始を待つ必要はありません。これにより、軸領域など外部の状態に依存する[チェーンされたトランジション](#)も大幅に単純化されました。また、`transition.transition` メソッドは、（同時実行するトランジションではなく）前のトランジション終了後に開始するトランジションを生成するようになりました。もう他のトランジションの終了イベントを待つことなく[トランジションをチェーンする](#)ことが可能になったわけです。

D3 におけるトランジションの詳しい仕組みについてはチュートリアル[「トランジションを使う」](#)や [[3.0へのアップグレード]]のトランジションの項をご覧ください。

リクエスト



3.0では、リクエスト発行のための D3 非同期メソッドが[d3.xhr](#)を返すようになりました。これによって従来よりずっと優れたリクエスト管理が可能になりました。たとえば[progress](#)、`load`、`error`の各イベントごとに別のリスナーを持たせるなど、複数のリスナー設定が可能となりました。[xhr.abort](#)を使って実行中のリクエストをキャンセルしたり、[xhr.header](#)を使ってリクエストヘッダを追加したり、[xhr.send](#)で HTTP メソッドや body データのカスタマイズも可能になりました。不本意ではありますが、マイクロソフトの非標準の `XMLHttpRequest` オブジェクトを使う方法で IE9 への CORS 互換性の対応も行いました。

また、非同期リクエストメソッドが Node.js のコールバック形式（コールバック関数への最初の引数が「エラー」、第2引数が「結果」という形式）に対応しました。これにより `d3.xhr` が[Queue.js](#)のような非同期 JavaScript ヘルパーライブラリとの互換性を持ち、複数のリソースの同時読み込みをずっと容易に行えるようになります。具体例は[[Upgrading to 3.0]]のリクエストのセクションをご参照ください。

その他

ここまで述べたことの他にも、3.0 では少数のバグフィックスと性能の改善点のほか、いくつかの新機能が加わっています。例えば新しく追加された `d3.shuffle` メソッドは [フィッシャー・イエーツ・シャッフル](#) を提供します。

`d3.format` クラスは `align` と `fill` に対応しました。`d3.format` と `d3.time.format` はいずれも実行時 [POSIX ロケール化](#) に対応するようになりました。

また、`d3.layout.treemap` は「スライスとダイス」アルゴリズムに対応しました。

最後に、ブラウジングとフォークをやすくするため、すべての D3 の事例集を bl.ocks.org と [GitHub Gist](#) に移転したことを書き加えておきます。