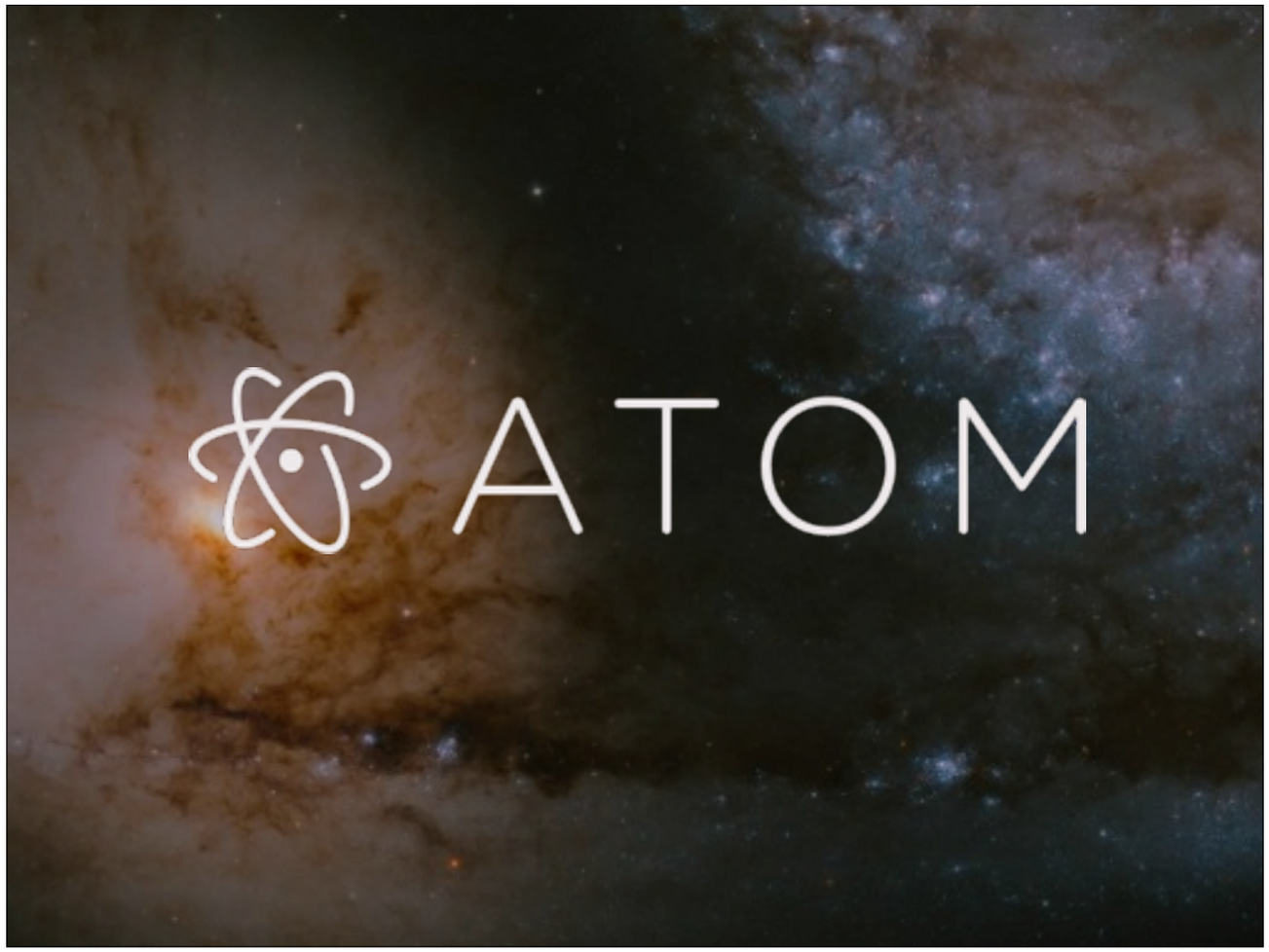
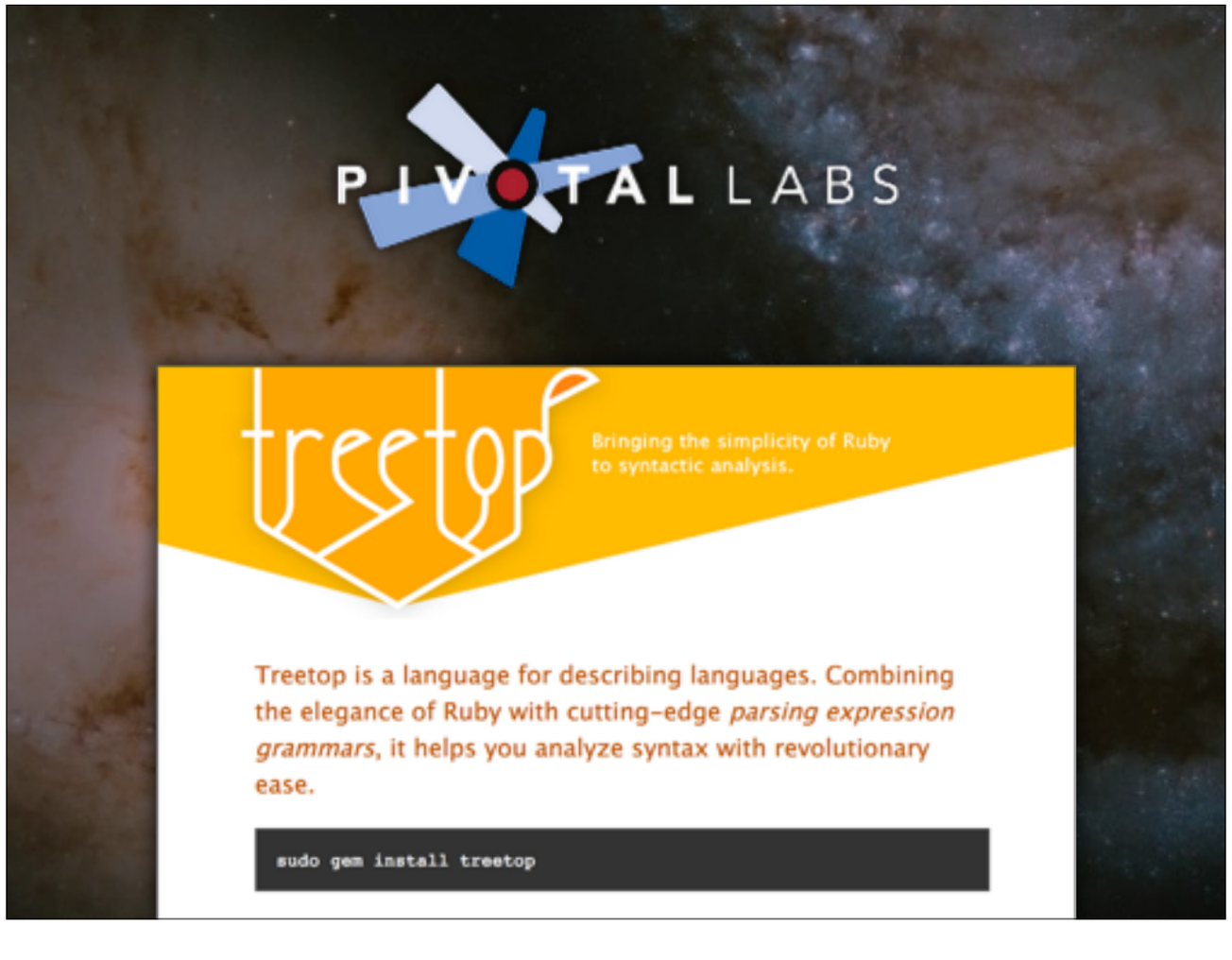
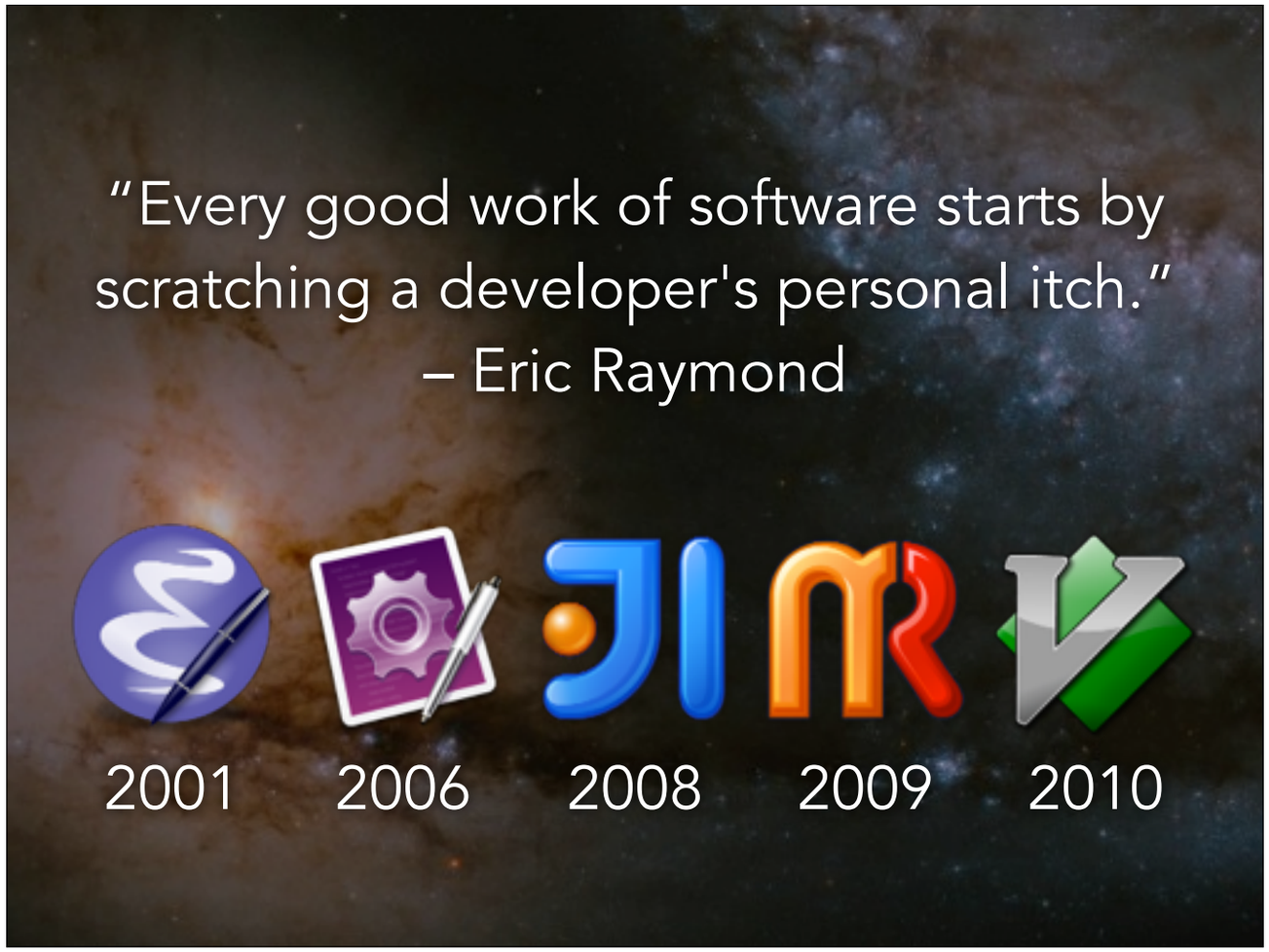

ATOM

- Hello



- Before GitHub, spent the most time at Pivotal Labs
    - GitHubの前は、Pivotal Labsに長い間所属していました。
- You may know me for my Treetop parsing framework
    - Parse frameworkのTreetopの開発をしていました。

- Been working on Atom for a long time
  - Atomは長い間開発していました。
- Really my story with Atom goes back much further
  - ただ私とAtomの物語はもっと昔にさかのぼります。
- Treetop was supposed to be a text editor
  - Treetopは元々テキスト・エディターになるはずでした。



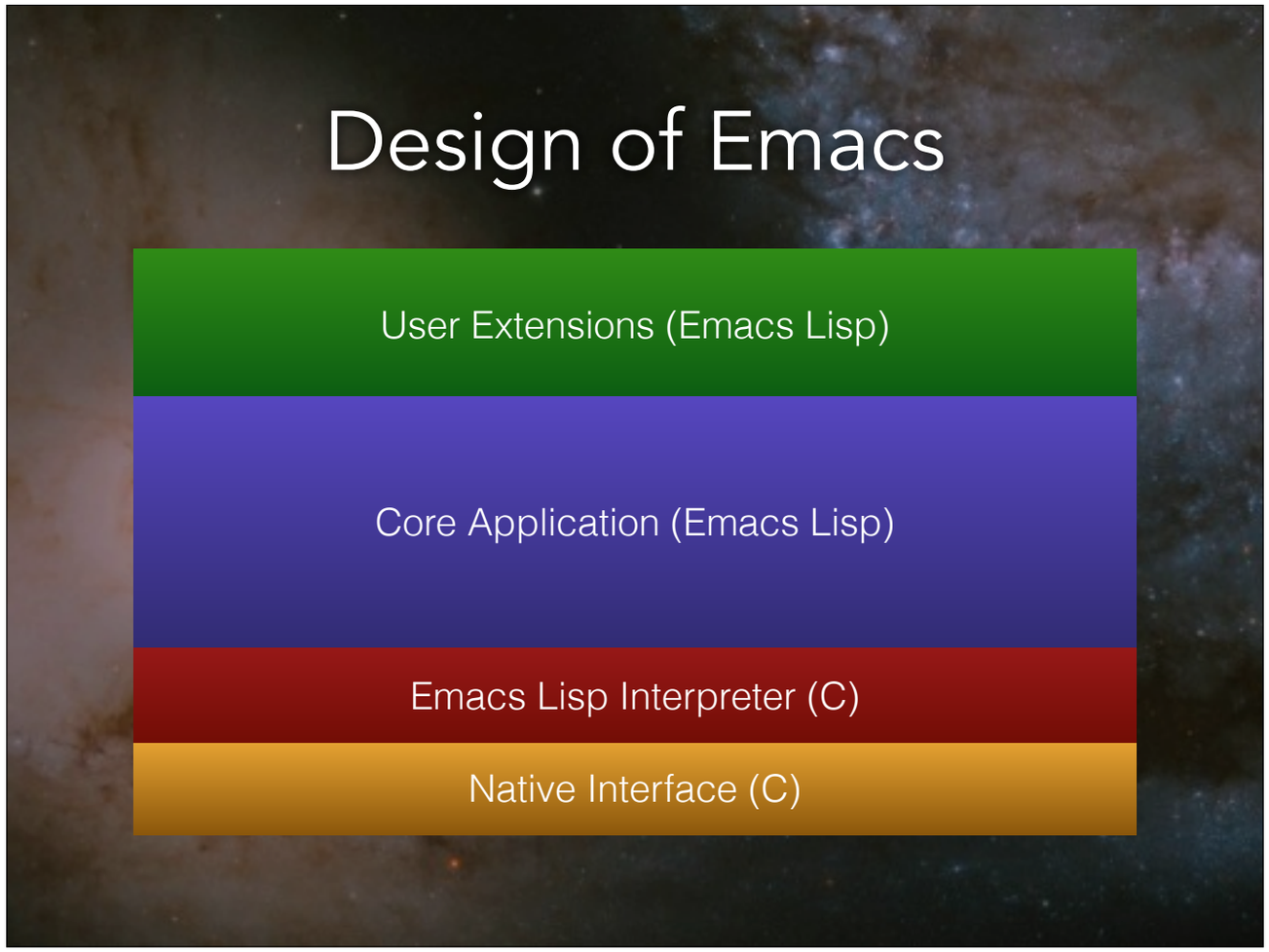
- Story of Atom starts long before GitHub
  - Atomの物語はGitHubの歴史より前に始まります。
- Eric Raymond says we build software by scratching an itch
  - Eric Raymond曰く我々は自分たちで気になる事を改善するためにソフトウエアを開発する。
- Used a lot of text editors over my career, never been quite satisfied, wanted to build my own tool.
  - 今まで様々なテキスト・エディターを使ってきましたが、満足は出来ませんでした、自分のツールを開発したいと思っていました。
- Pitched Chris on the idea in Summer of 2011 and started building Atom.
  - Chris (@defunkt)に2011年の夏にアイデアをピッチしてAtomの開発を始めました。
- Before we dive into Atom, I wanted to go back to the two editors that have had the biggest influence on me.
  - Atomを紹介する前に自分に今まで一番影響を与えた2つのエディターを紹介します。

- First editor I ever used in college
  - 大学で初めて使ったエディター。
- Deeply programmable
  - 深部まで開発が出来る。
- Doesn’t just expose an API. It’s an open system. No fundamental distinction between code that ships with Emacs and code written by users.
  - APIを公開しているだけでない。オープンなシステムである。Emacsと同封されているコードとユーザーが開発しているコードには基本的に違いは無い。
- Also really old. Been evolving since the mid 70s and doesn’t work like any modern desktop applications.
  - 非常に古い。70年台から進化をしているが、近代のデスクトップアプリと同じようには使えない。
- Requires lots of setup just to be productive.
  - 生産性を生むまでに様々なセットアップが必要。

- Jumped on the bandwagon after watching the DHH Rails video
  - DHHのRailsビデオを見てからTextMateのファンになった。
- First editor I really used in a professional capacity
  - プロになってから初めて本当に使い込んだエディター。
- Attractive, at home in OS X
  - 美しくてOSXネーティブである。
- Simple, provided convenient features out of the box for a productive workflow
  - シンプルであり、インストールした時から生産性が高いワークフローを可能にする便利な機能を持っている。
- But fundamentally limited
  - しかし、基本的なところで制限がいっぱいある。

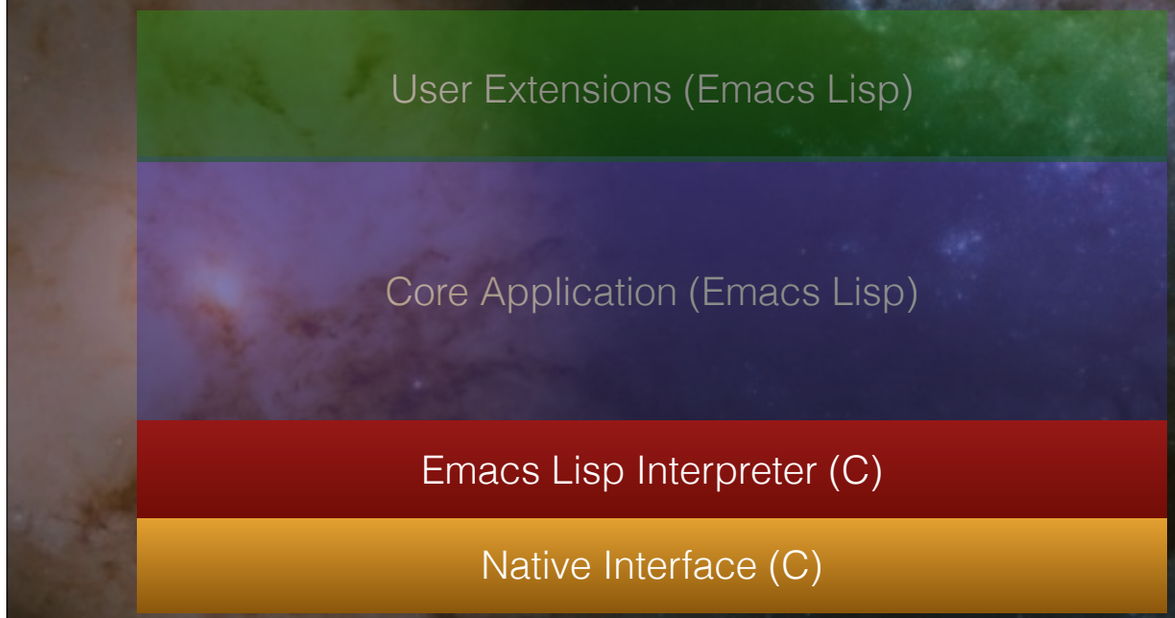
- As soon as I started outgrowing TextMate, I really wanted an editor that provided what I liked about both editors
- TextMateでは間に合わなくなってきた頃から、このエディター両方の好きなところを含んでいるエディターが欲しくなっていた。
- I wanted to build a modern Emacs that had the intuitive experience and convenient features of TextMate but the same hackable spirit
    - TextMateの直感的な使い方と便利の機能を持ちつつ、Hackも出来る近代的なEmacsを開発したかった。

Design of Emacs



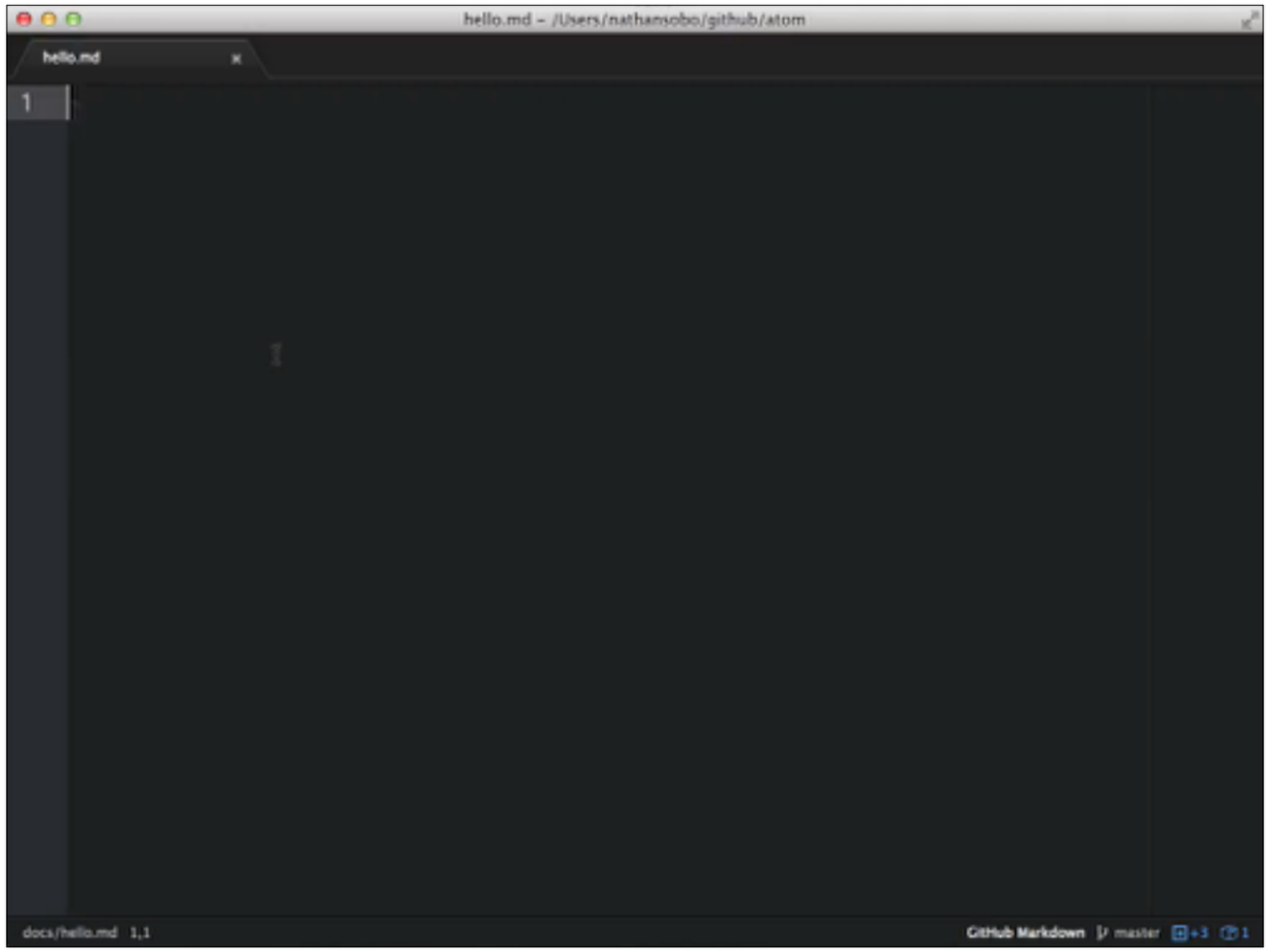
- Thought experiment: How would you design Emacs using modern technology.
  - ここで質問：近代的なテクノロジーを活用してEmacsを構築する場合はどう設計しますか?
- Let's look at how Emacs was designed to begin with.
  - Emacsが元々どうデザインされたかを見てみましょう。
- Emacs has its own programming language.
  - Emacsは独自の開発言語を持っている。
- So really it's not just a text editor, it's a programming language implementation and runtime environment.
- なので、テキスト・エディターだけではなく、開発言語とランタイム環境も実装している。
- As much of Emacs is possible is written at a much higher level of abstraction.
  - Emacsは出来る限りもっと上位層で書いてある。
- Because it's mostly written in an interpreted language, there's a very low barrier to being modified by users in that same language.
  - インタープリタ型言語なので、同じ言語を活用してユーザーが簡単に変更出来る。

Design of Emacs



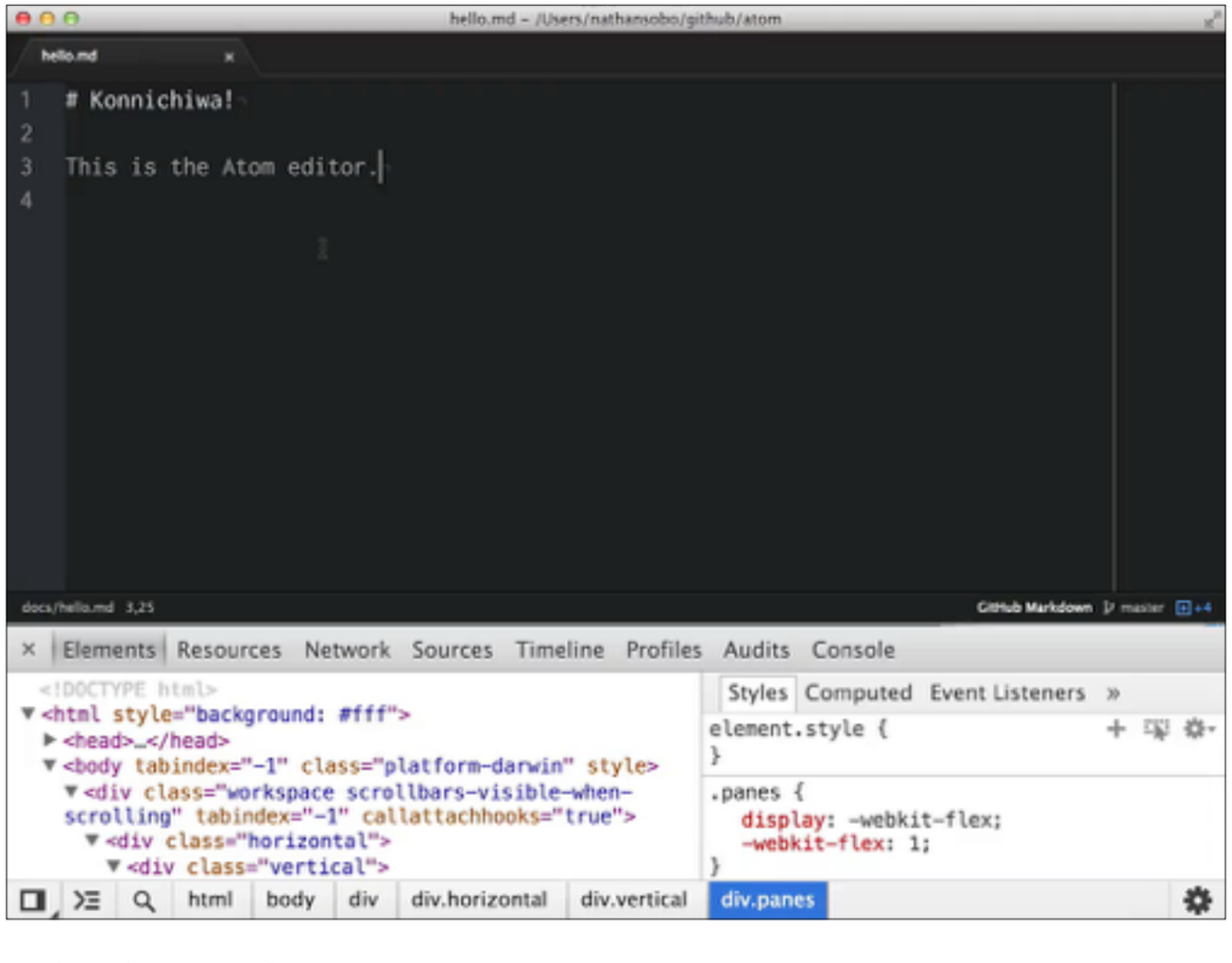
- So if we're going to build a modern Emacs, we first need to focus on the native foundation layer.
  - なので、近代的なEmacsを構築する場合は、ネーティブの基礎層を最初に考える必要がある。
- How long would it take us to build that layer ourselves, make it performant, cross platform.
  - この階層をパフォーマンスを高く保ちながらクロスプラットフォームで使えるように開発を自分達で行った場合はどれくらい時間がかかるか。
- We would want a familiar scripting language and a cross platform UI layer
  - 一般的に知られているスクリプト言語とクロスプラットフォームなUI階層を持っていて欲しい。
- Luckily, such a system already exists
  - 実はすでにそれを可能とするシステムが存在していました。

- We call it a web browser
  - 我々はウエブブラウザーと呼んでいるものです。
- Chromium was designed to browse web pages
  - Chromiumはウエブページをブラウズするために設計されました。
- But the distinction between web page and application is rapidly blurring as web technology matures
  - ただし、ウエブテクノロジーが成熟化していくのと一緒にウエブページとアプリケーションの境目は急速に薄れてきました。



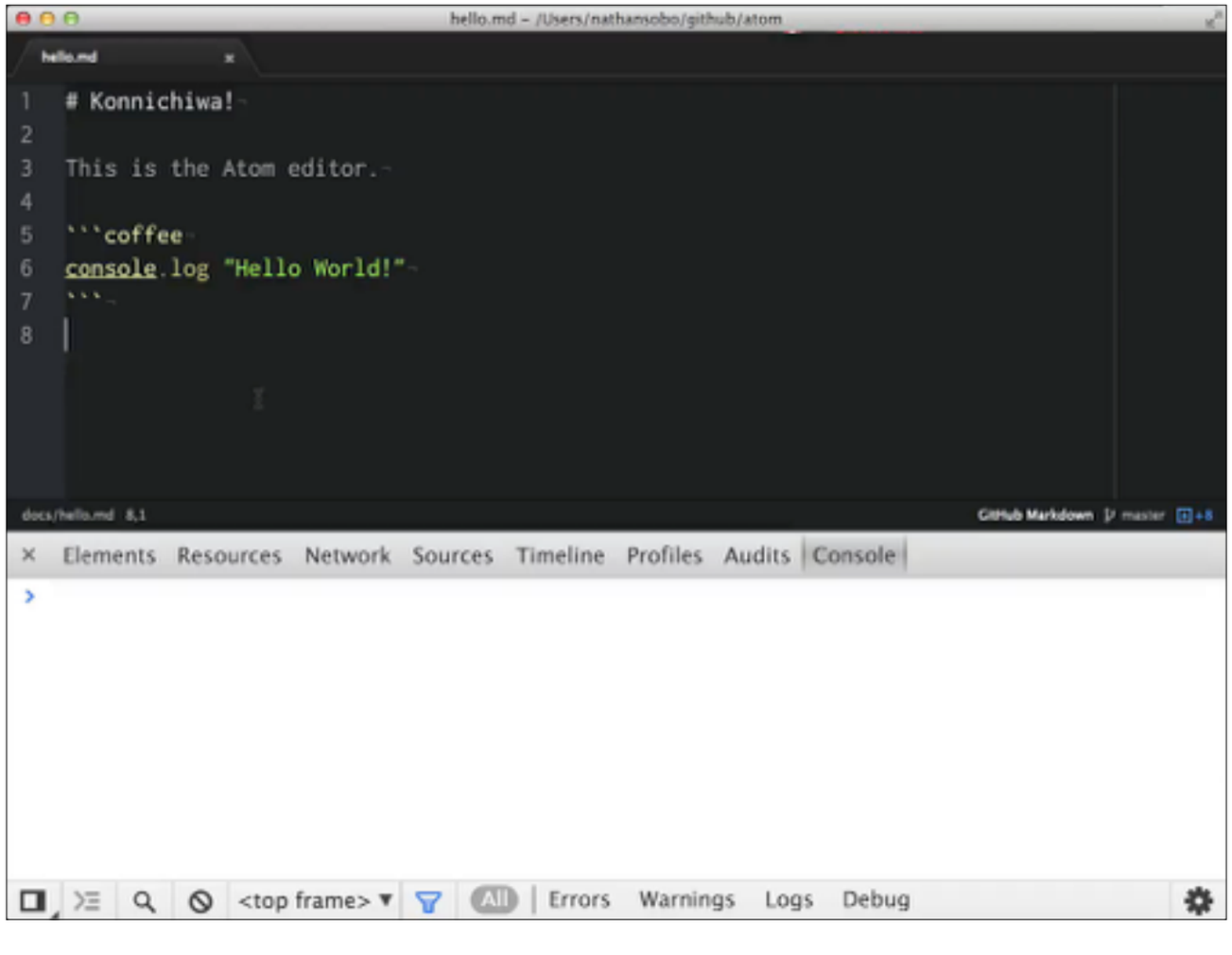
- So this is Atom
    - これがAtomです。
- As you can see, at first glance it feels a lot like TextMate
    - 見ての通り、一見ではTextMateni非常に似ています。
- It has an uncluttered interface that's focused on simple editing
- 簡単に文書を書けるように非常にシンプルなインて～フエイスを持っています。
- But it also has features like a tree view, a way to quickly find files, find and replace across the project, etc
    - ですが、トリービュー、素早くファイルを検索する方法、プロジエクト全体で使える検索と置換機能等など様々なものを保有しています。
- It also provides cool features like snippets, multiple cursors, etc
    - 更に、スニペット、マルチカーソル等の素晴らしい機能もあります。
- None of this is really ground-breaking, and we didn't intend it to be.
    - これはけして革新的なものではありませんが、我々は革新的なものを作る意図はありませんでした。
- We just wanted a simple and familiar text editing environment as our starting point
    - まずはシンプルでわかりやすいテキスト・エディター環境を作りたかったのです。
- The real magic is in how Atom is built
    - 本当のポイントはAtomがどうやって構築されているかです。
- Because it's based on Chromium, Atom is its own super flexible development environment with first class development tools and access to all the latest technologies

- First and foremost, it comes with a sophisticated and declarative UI framework
    - まずは、洗練された宣言型なUIフレームワークを同封しています。
- A huge number of developers and designers already know how to build sophisticated interfaces using this technology
    - 多数の開発者とデザインががすでにこのテクノロジーを活用して洗練されたUIを構築する方法を知っています。
- But HTML and CSS are great not just because they’re ubiquitous
    - ただし、HTMLとCSSの素晴らしさは何処ででも活用されているからだけではありません。
- CSS provides a really nice abstraction for allowing styling in a declarative way, which is great for theming and tweaking the look of the editor
    - CSSはスタイルを宣言型でデザインする素晴らしい方法をもっています、これはエディターのテーマを作ったり、見た目を修正するのには非常に便利です。
- The DOM certainly isn’t perfect, but being able to inspect it, manipulate it on the fly, and query it with selectors makes for a really malleable system.
    - DOMは当然最高ではありませんが、検査も出来たり、実行中に編集したり、セレクターもクエリーできるので、非常に柔軟なシステムです。
- If you want to achieve something in the editor, the flexibility of these technologies gives you a good chance of doing it even if there’s no API yet
    - エディターで何かをしたい場合はこのテクノロジーの柔軟性はAPIが存在しなくても色々な事を可能にします。



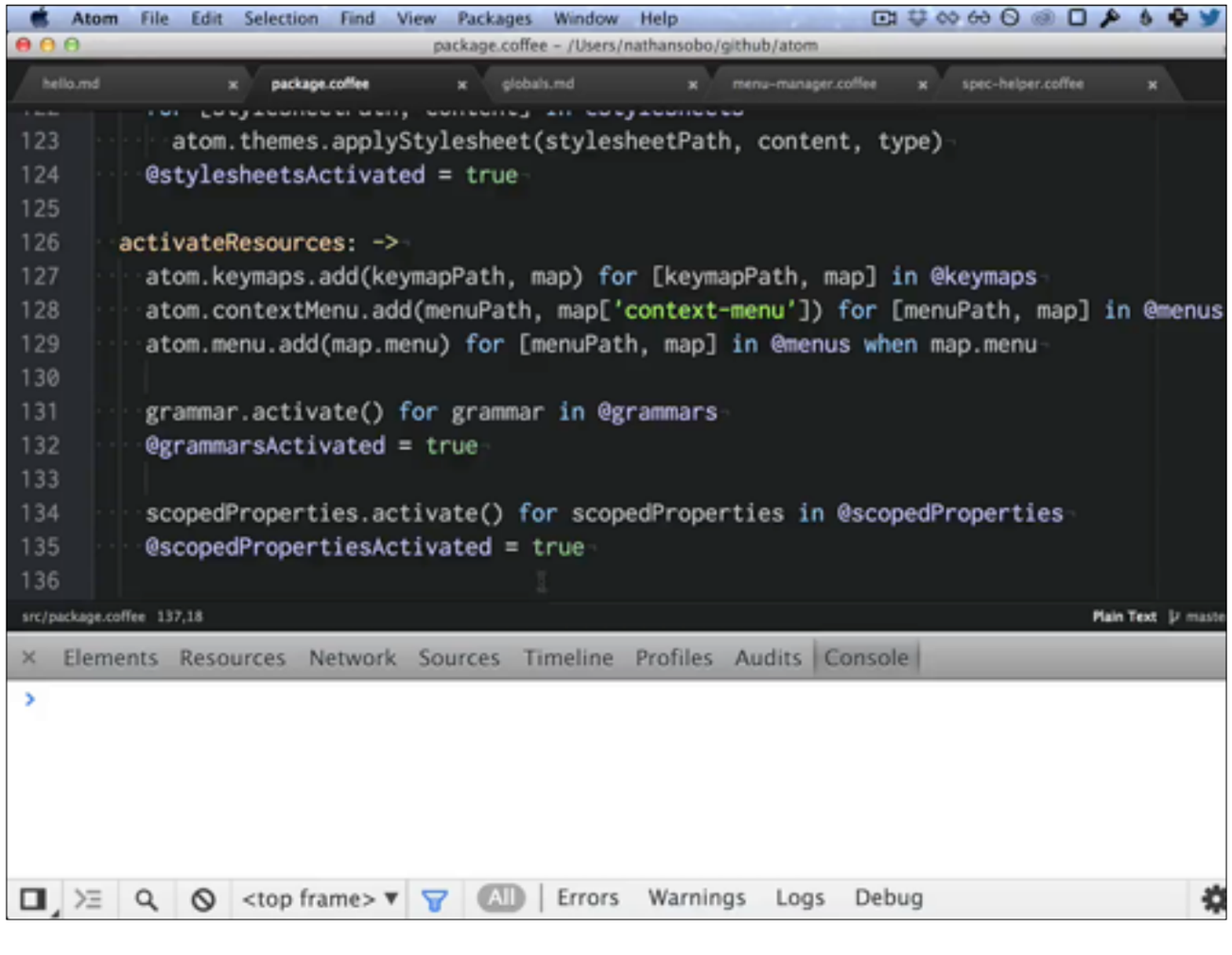
- So here you can see the power of everything being on the DOM
    - ではDOMで全てをアクセス出来る事がどれくらいすごいことかを紹介します。
- I can use the inspector to zoom in on a line I just typed and find out it's CSS style
    - インスペクターを使って、今書いた一行に行ってその行のCSSスタイルを確認できます。
- Then I open my user stylesheet and add a rule that matches that element, so I can change the color, and it's reloaded on the fly.
- ユーザースタイルシートを開き、そのelement用のルールを追加しました、なので色を変更すると、すぐに反映されます。
- Other editors might let you edit the theme, but in Atom everything can be styled this way. So for example, what if I want to change the look of the cursor?
- 他のエディターはテーマを編集できますが、Atomでは見えるもの全てがこのように編集できます。例えば、カーソルの見た目を変えたい場合は?

- As I said earlier, one of the great things about Emacs is that it has a lisp interpreter baked into it, and most of the application is written in that interpreted language.
  - Emacsはlisp interpreterを含んでいるので、アプリケーションのほとんどはその言語で開発されています。
- Building an efficient scripting language VM is a massive undertaking
  - 効率が良いスクリプト言語のVMを開発するのは大きなプロジェクトです。
- Luckily, Chromium already comes with one of the most sophisticated scripting language VMs in existence, V8.
  - Chromiumは現存する最も洗練されているスクリプト言語VMのV8を含んでいる。
- Again, while JS isn't perfect, it's a pretty awesome scripting language.
  - JSは完璧ではないが、かなり素晴らしいスクリプト言語である。
- It's the most popular programming language on GitHub–way more people know JS than Emacs Lisp or VimScript.
  - GitHub上でもっとも人気がある開発言語である - すなわちEmacs LispやVimScriptを知っている人よりJSを知っている人が圧倒的に多い。
- Fast, tons of libraries. And lots of other languages can compile to it if you don't want to use JS itself.
  - 速く、ライブラリーも非常に多い。他の言語もJSにコンパイルできるので、JSを直接使う必要も無い。

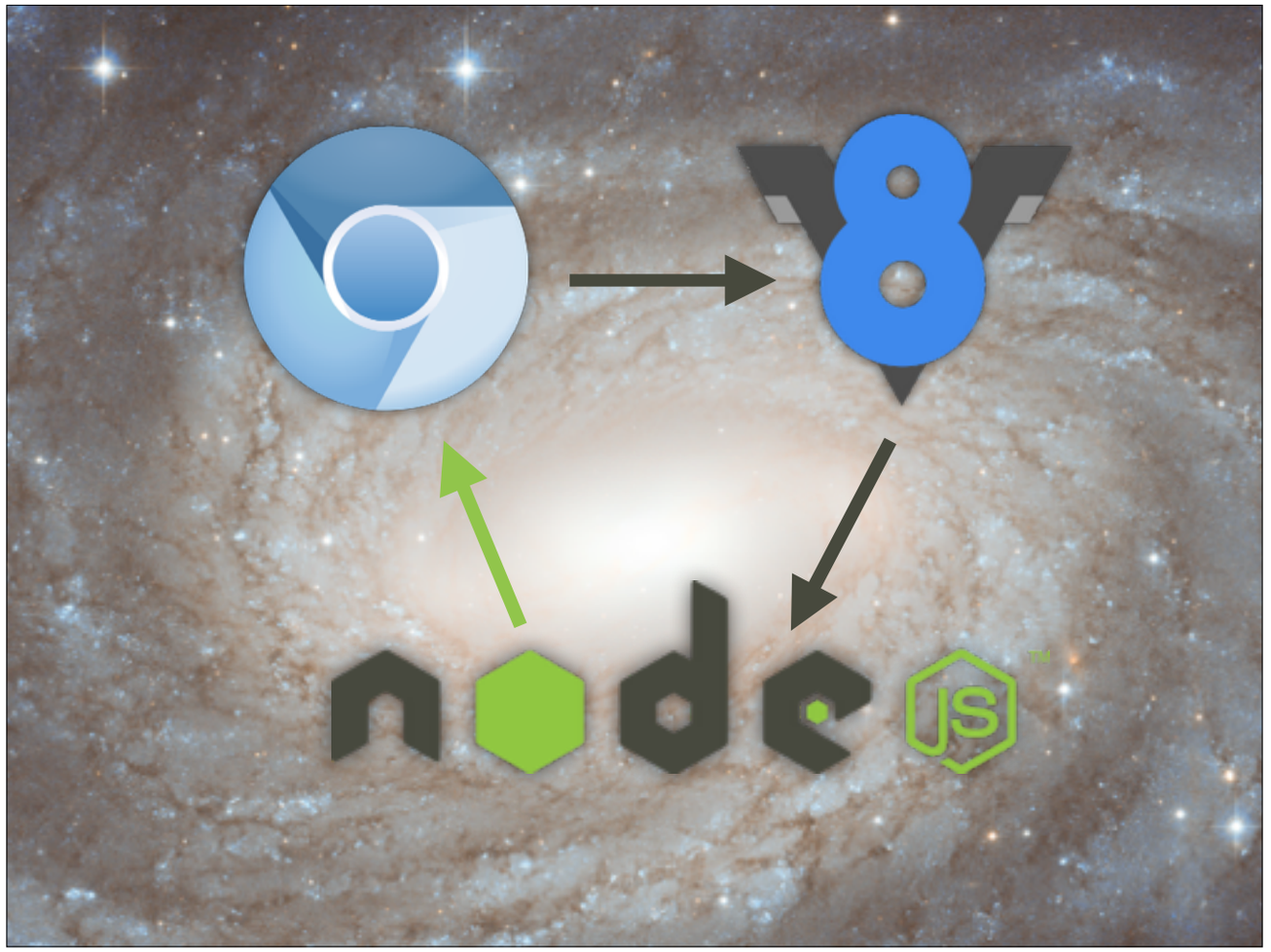


- So here's what it means to have JavaScript as our interpreter directly inside of our text editor.
    - それではテキスト・エディター内でJavaScriptが含まれていて実行出来るのはどのような事かをお見せしましょう。
- Anything we could do by interacting with the editor we can also script
    - エディターを触れてできることは全てスクリプトも出来る
- So we can split a pane
    - なので、ペインを割いたり
- Open a file
    - ファイルを開いたり
- Get the contents of an editor
    - エディターのコンテンツを取得したり。
- Or even set the contents of an editor
    - エディターのコンテンツを設定することも可能です。

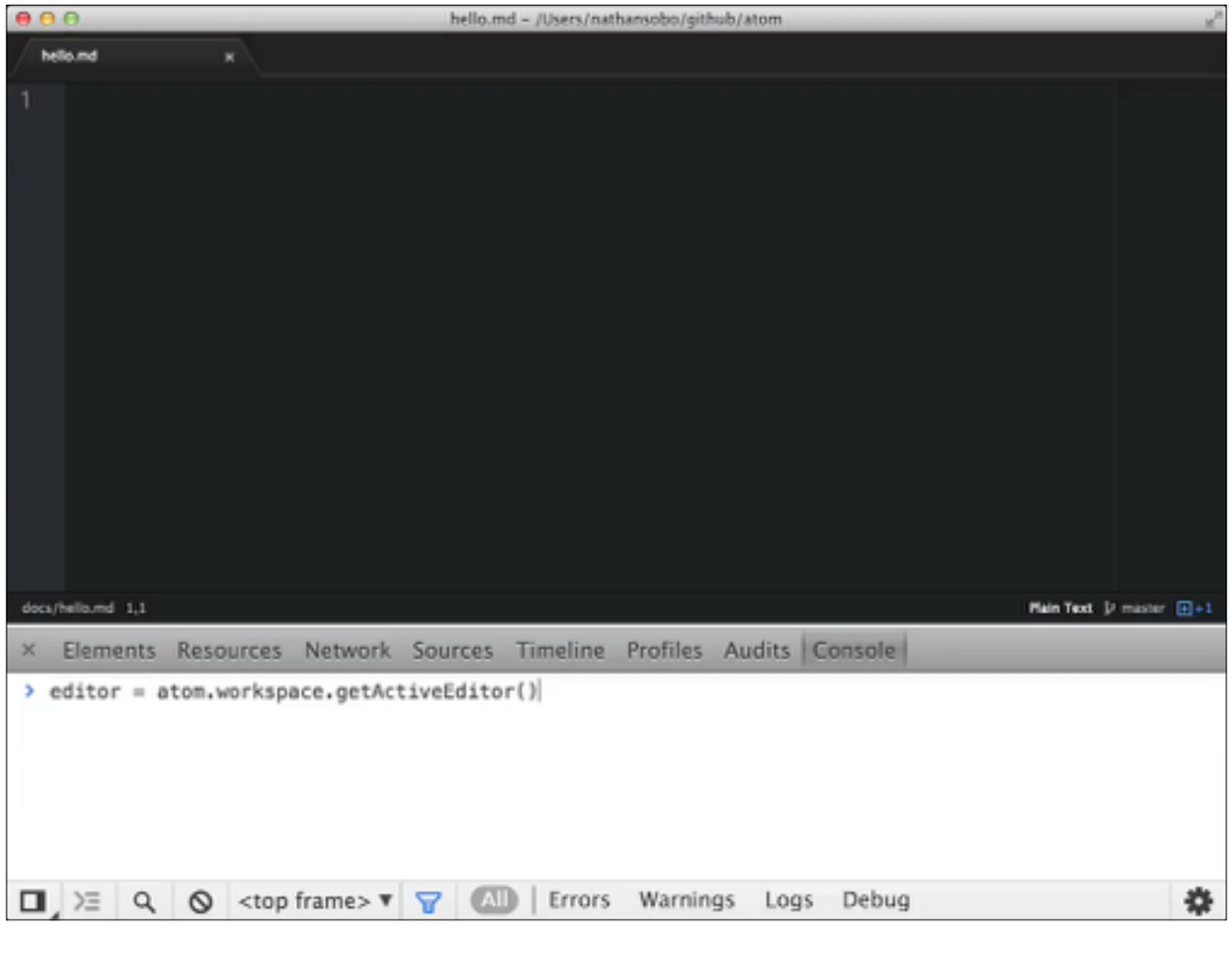
- But we didn’t want to build a text editor that runs in the browser
  - ただ、ブラウザー上で動くテキスト・エディターを作りたかったわけでもない。
- It was important to us that Atom be a real desktop application that integrates smoothly into the host OS and has full privileges
  - ホストOSに統合されていて、権限も全て保有する本当のデスクトップアプリケーションを開発したかったのです。
- That’s why we built Atom Shell, which is based on the Chromium content module. It’s designed to streamline the process of building desktop applications with web technology
  - なので、Chromimum content moduleをベースにしたAtom Shellを開発しました。ウエブテクノロジーを活用したデスクトップアプリケーションを開発する作業を簡素化するように設計されています。



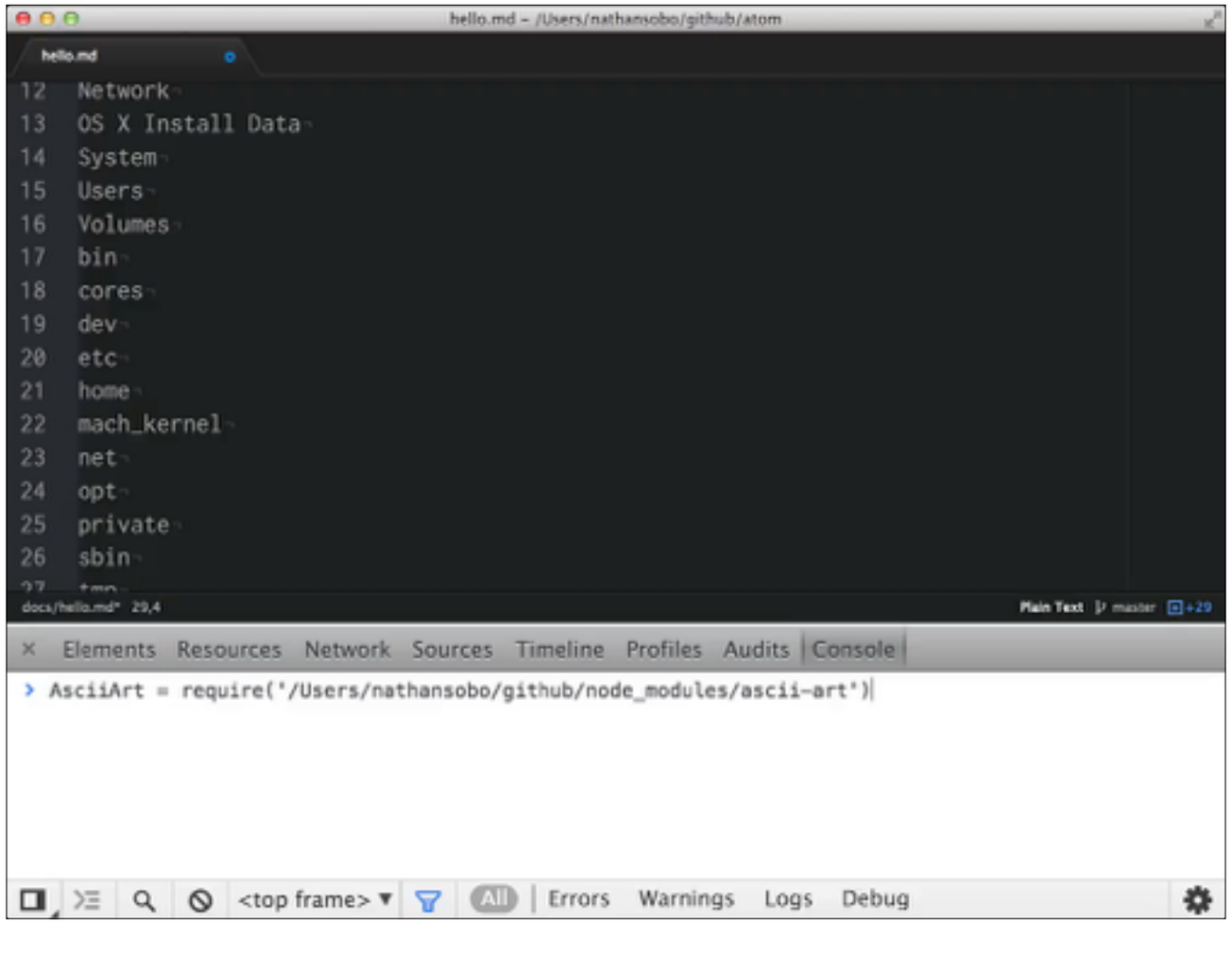
- So for example, one thing you'll find in Atom shell that you won't find in a web browser is the ability to add system menus
  - 例えばAtom Shellに含まれてウエブブラウザーに無いものの例はシステムメニューを追加する方法。
- Or open a native open dialog
  - ネーティブな「開く」ダイアログ



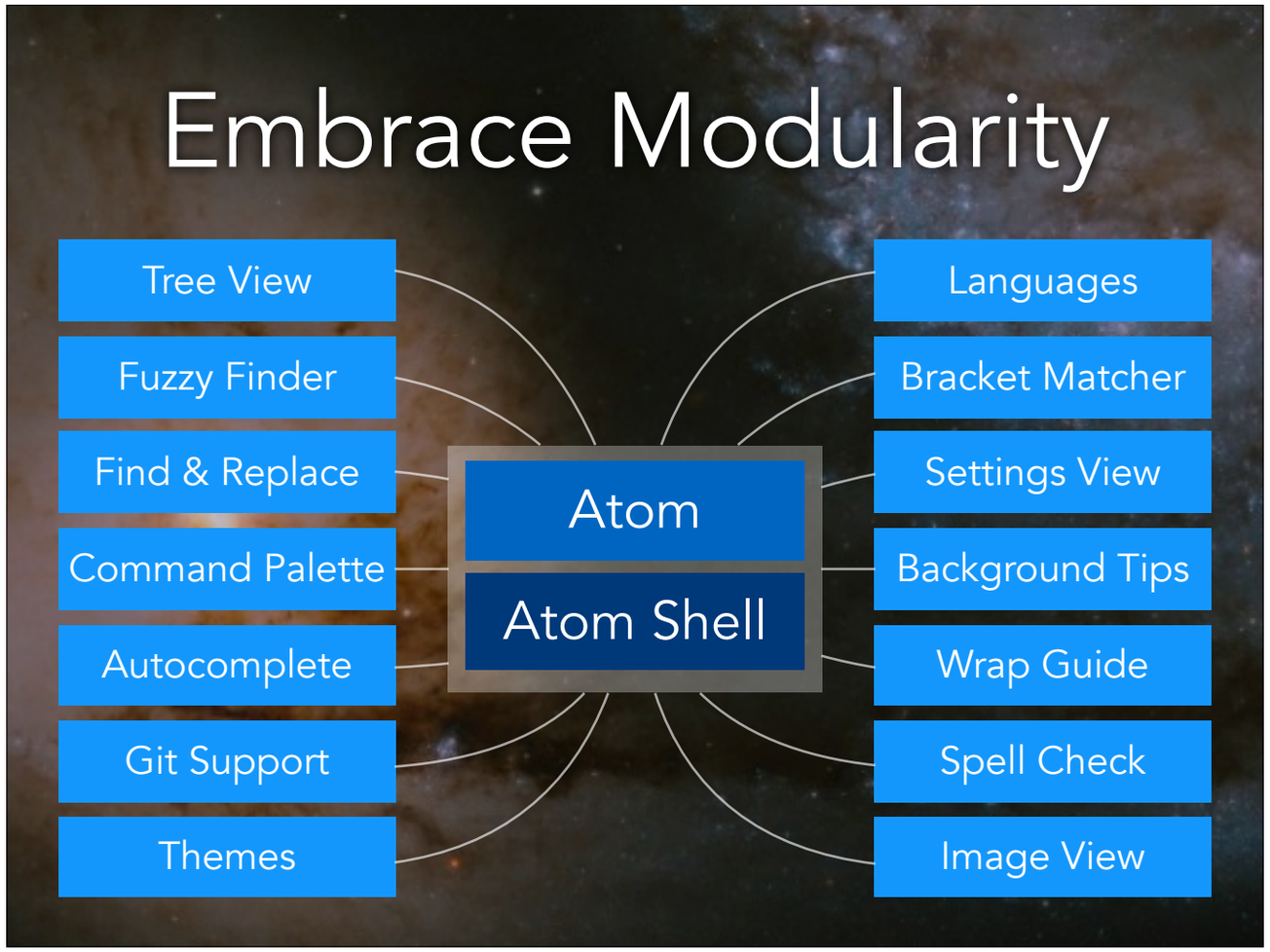
- But we need to go beyond giving the application a dock icon and a system menu
  - しかし、アプリケーションにドックのアイコンやシステムメニューを与えるだけでは足りません
- Serious desktop applications (like a text editor) need to have privileged access to the native system. We need to be able to read and write files, start subprocesses, or even run servers
  - テキスト・エディター等の本当のデスクトップアプリケーションはシステムに権限をもったアクセスする必要があります。ファイルを読んだり書き出したり、サブプロセスを開始したり、サーバーを起動したりも出来る必要があります。
- These are things that never developed within browsers because of security reasons
  - セキュリティー上の理由もあり、ウエブブラウザーにはこの機能は含まれていません。
- However, the Node.js project has made a lot of progress in making JavaScript useful on the server
  - Node.jsプロジェクトはJavaScriptをサーバー上で実行するために色々と開発を進めてきました。
- Node already has APIs for all the privileged things we want to do, plus other things we need like a module system
  - NodeはAtomが必要としている権限のAPIやモジュールシステム等をすでに保有しています。
- Atom shell injects Node.js into Chromium. It's like programming in the web browser with the ability to access Node APIs, or writing a Node application that has a window object.
  - Atom ShellはChromiumにNode.jsを注入します。Node APIにアクセス出来るウエブブラウザ内の開発や、Window objectを持ったNodeアプリケーションを開発するようなものです。



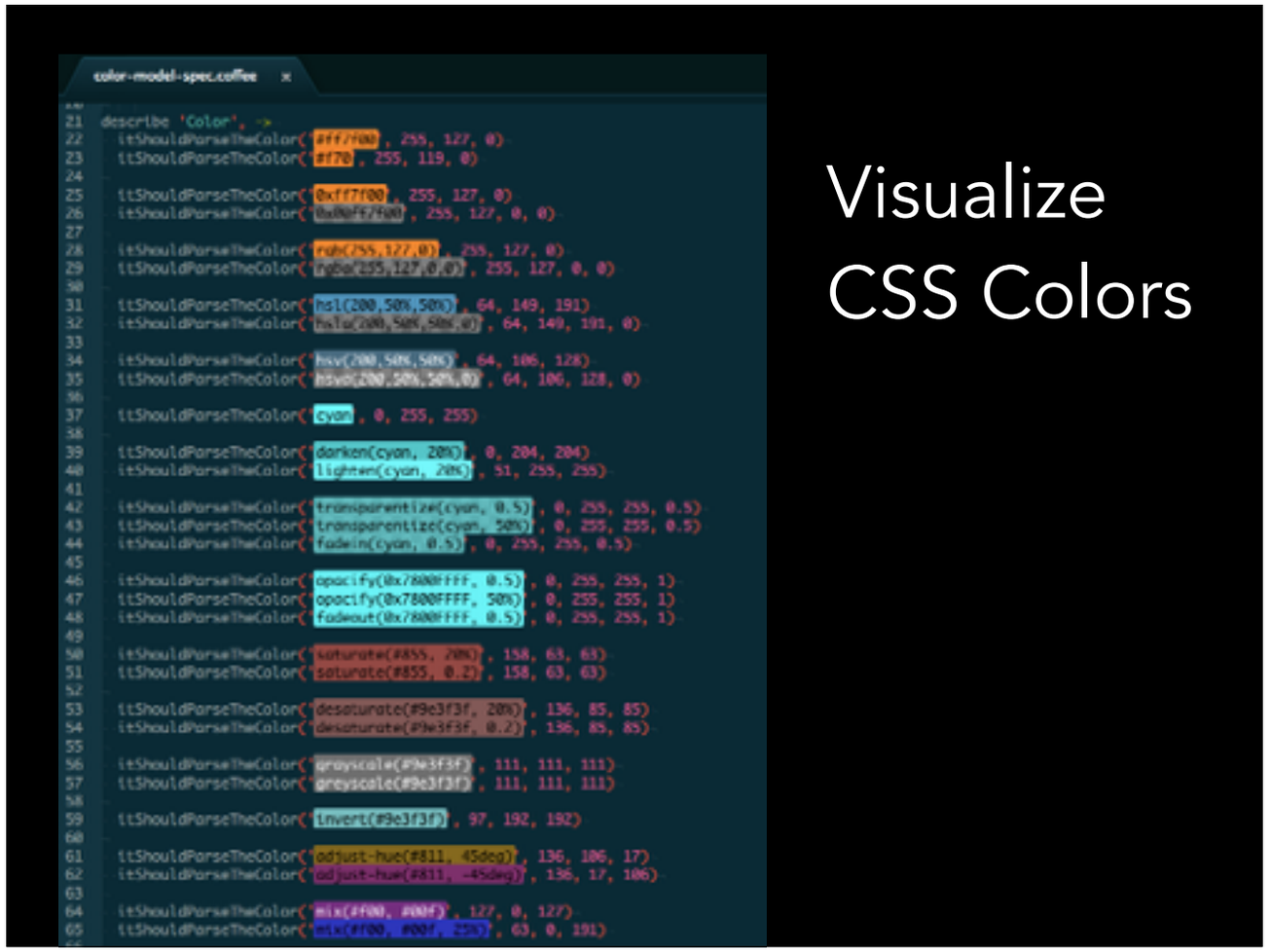
- So in this example, you see we can use require to load the `fs` module.
    - この例では'fs'モジュールをロードするためにrequireを活用しています。
- Then we can read from the file system just as we would in a node application. This wouldn't be possible in a web browser for security reasons.
    - その後にnodeのアプリケーションと同様、ファイルシステムから直接読み取りができます。ウエブブラウザーの場合はセキュリティー上の理由からこれは可能であありません。
- We can take the result and combine it with the Atom APIs.
    - 結果をAtom APIと統合が出来ます。



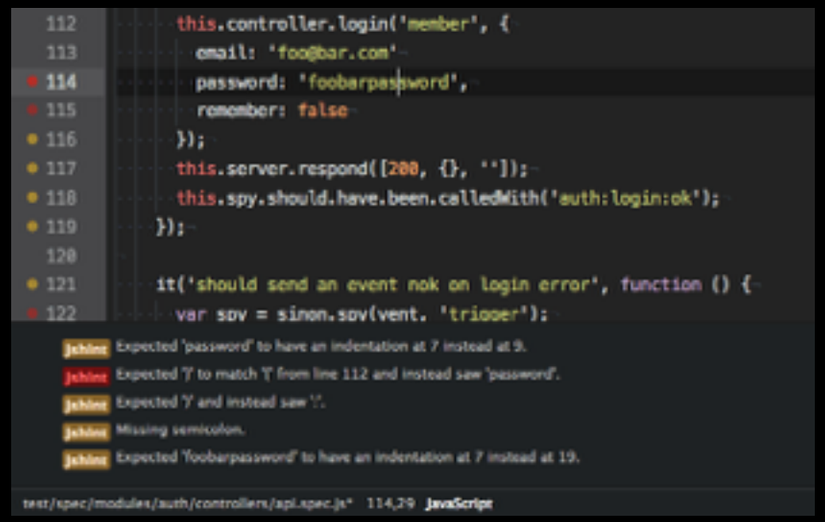
- Another great thing we can do is take advantage of npm modules. Here I'm loading the ASCII art module, and I can use it to stylize text.
    - さらに便利なのはnpmモジュールを活用できることです。ここではASCIIアートのモジュールをロードして、テキストのスタイルをします。
- Node integration also allows us to interface to code written in C and C++.
    - Node都の統合で、CやC++で開発されたコードにもアクセスが出来ます。
- We use this ability in a number of places, such as linking to the Oniguruma regular expression library to support TextMate grammars, using kqueue to watch the file system, and even using OS X API's to determine information about the scrollbar style.
    - これは幾つかの場所で活用しています、例えばTextMateの文法に対応するためにOniguruma regular expressionライブラリーにリンクしたり、ファイルシステムを監視するのにkqueueをつかったり、スクロールバーのスタイル情報を知るためにOSXのAPIも使っています。



- Using web technology will really help us maximize participation in Atom by ensuring lots of developers have a low barrier to contributing
  - ウエブテクノロジーを使うことによって、Atomへの参加のハードルを下げて、色々な開発者が貢献出来るようにしています。
- We want Atom to evolve, and making the system modular is a big part of that strategy
  - 我々はAtomに進化して欲しいので、システムがモジュール化されているのはその戦略の大きな部分です。
- We've tried to deliver each major piece of functionality in its own module, keeping the core as small as possible
  - 大きな機能は独立したモジュールで提供して、コアを可能な限り小さくしています。
- This means that over time, we'll be able to swap out pieces of the system. By keeping each component small, it's easier to maintain. It's also easier to replace.
  - この設計によって、将来的にはこのシステムのパーツを交換できます。各パーツを小さくすることによってメンテナンスも楽になります。交換もより簡単です。
- We also unleash the forces of competition.

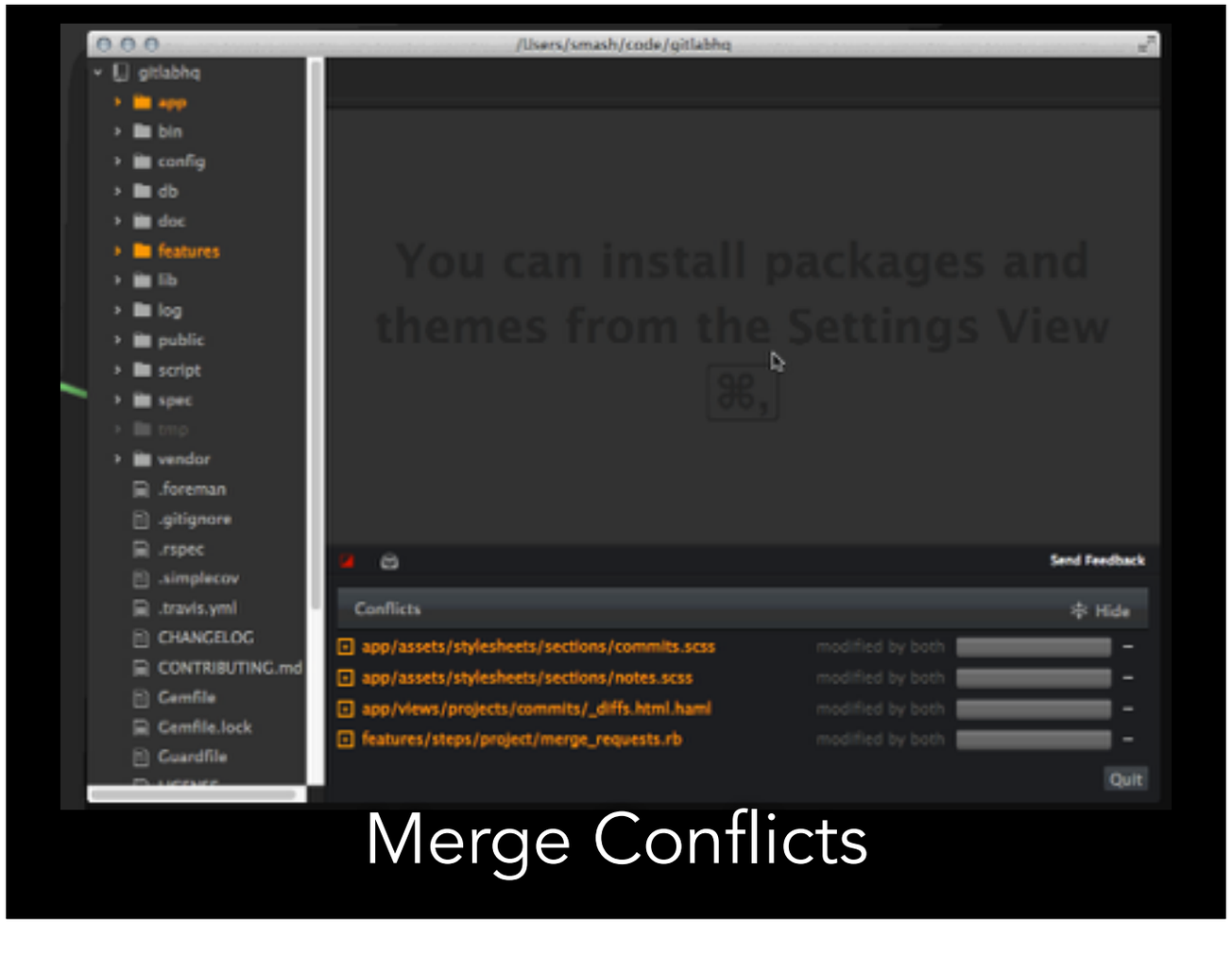


Visualize
CSS Colors

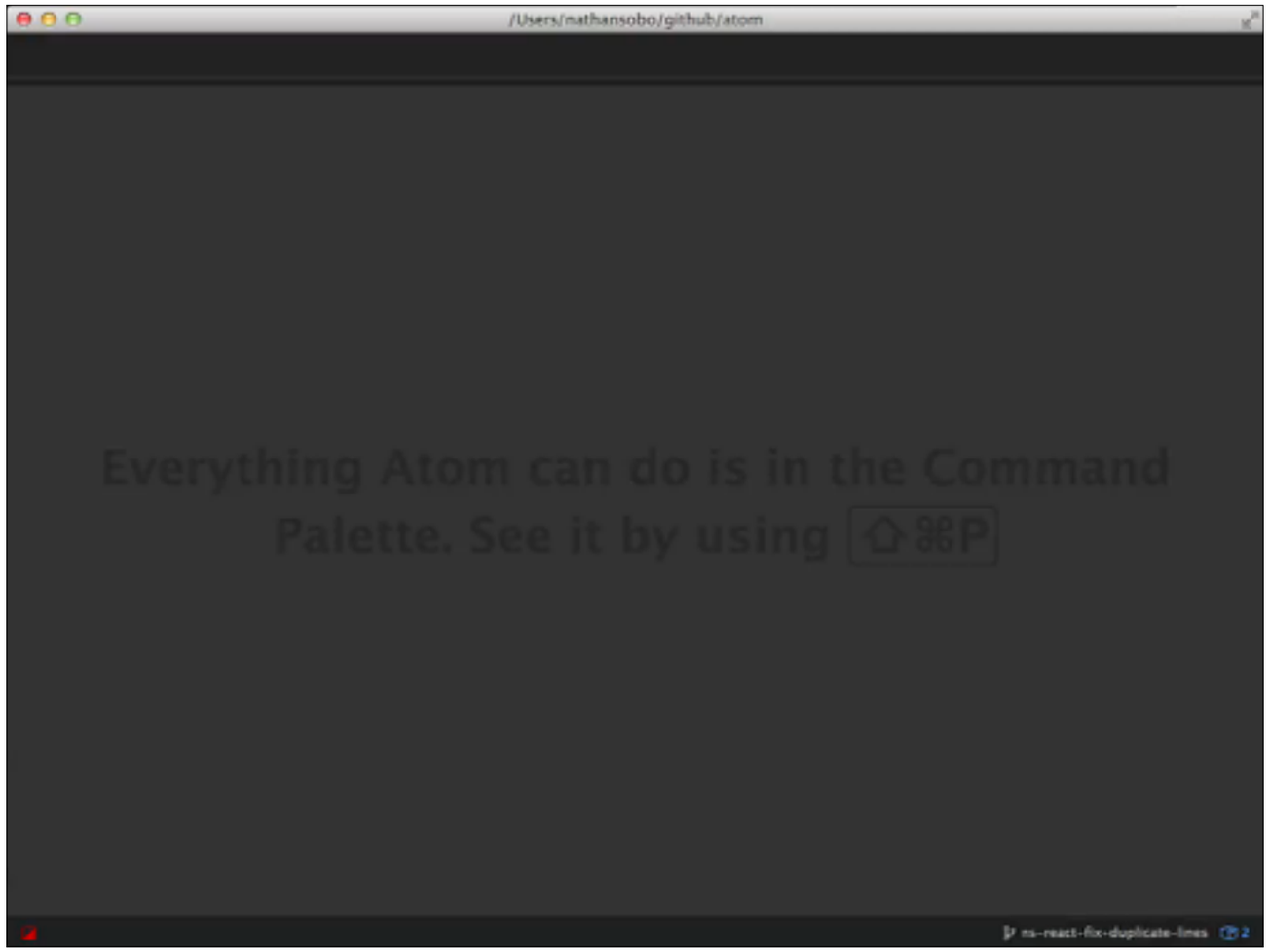

this.controller.login('member', {
email: 'foo@bar.com'
password: 'foobarpassword',
remember: false
});
this.server.respond([200, {}, '']);
this.spy.should.have.been.calledWith('auth:login:ok');
});
it('should send an event nok on login error', function () {
var spy = sinon.spy(vent, 'trigger');
jshint Expected 'password' to have an indentation at 7 instead at 9.
jshint Expected '}' to match '{' from line 112 and instead saw 'password'.
jshint Expected ';' and instead saw ':'.
jshint Missing semicolon.
jshint Expected 'foobarpassword' to have an indentation at 7 instead at 19.
test/spec/modules/auth/controllers/api.spec.js* 114,29 JavaScript

Code Linter

Bezier Curve Editor



Merge Conflicts



- Here’s how you can create your own simple Atom package in less then 3 minutes
    - 3分以内で簡単なAtomのパッケージを作成する方法を紹介します。
- This package will convert selected text to ASCII art
    - このパッケージは選択されたテキストをASCIIアートに変換します。
- This same example is used as a tutorial on atom.io
    - この例はatom.io上のチュートリアルに使用されています。

- Start using Atom
  - Atomを使ってください。
- Find something that annoys you. Submit a PR on core or a bundled package.
  - 改善したいところを探してください。コアやパッケージへのPull Requestを提出してください。
- Find an idea that inspires you. Build your own package.
  - インスパイアされるアイデアを探してください。そして、自分でパッケージを作ってみてください。
- We have an official build on OS X. You still need to build manually on Windows and Linux and could use help with that.
  - OSX用の公式ビルドはあります。ウィンドウズではまだマニュアルでビルドする必要があるので、そこを改善するお手伝いは募集しています。

- Lots of people are already collaborating with us. We want you!
  - 多くの人がすでに参加しています。みなさまも是非参加してください。

- Our Mission at GitHub is to build better software by working together
    - GitHubのミッションは一緒に働く事によってより良いソフトウエアを開発することです。
- We’ve designed Atom with that philosophy in mind
    - Atomはこのミッションをもとに設計されています。
- Atom will succeed based on the strength of its community
    - Atomの成功はコミュニティーによって決められます。


Questions?