FUEL

Firefox User Extension Library

FUELとは？

- Firefoxの拡張機能開発者のための JavaScriptライブラリ
- よく使う処理について、XPCOMの呼び出しなどの面倒な手順を省略可能にする
- Prototype.jsに近いイメージ
- 発音は「フエル」？「フューエル」？

比較

- FUEL使用前

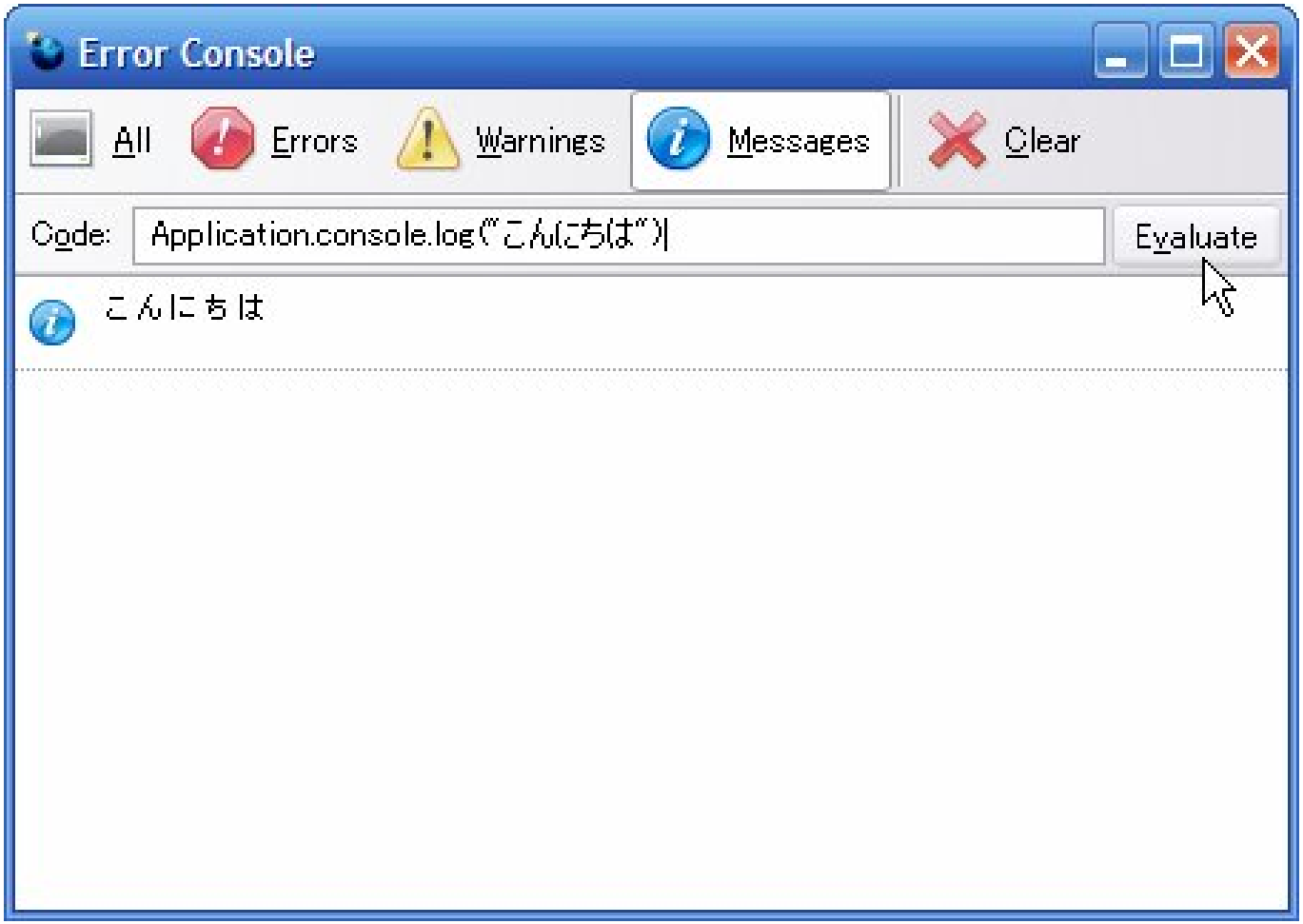
```
var cs = Components.classes["@mozilla.org/consoleservice;1"]
            .getService(Components.interfaces.nsIConsoleService);
cs.logStringMessage("こんにちは");
```

- FUEL使用後

```
Application.console.log(“こんにちは”);
```

FUELを試すには？

- Firefox 3.0a5が必要となります



FUELの実体

- JavaScript製XPCOMコンポーネント
- componentsフォルダ内のfuelApplication.js
- グローバルなオブジェクト「Application」によってどこのXULからでも呼び出せる
- ただし、JS製XPCOMから呼び出すには…
  - Components.classes["@mozilla.org/fuel/application;1"]
    .getService(Components.interfaces.fuelIApplication)

FUELのバージョン

- FUEL 0.1
  - すでにFirefox 3.0a5にて使用可能
- FUEL 0.2
  - 5月末に向けて作業中
- それ以降
  - MozillaWikiで現在検討中
- 将来的にはFirefox 1.5～3.0で使えるようになる予定？
  - Firefox 1.5のサポートはそろそろ終わる頃では？

FUEL 0.1の紹介

紙面の都合上、ここから先は以下のように略させていただきます

- Cc = Components.classes
- Ci = Components.interfaces

Application

- Ci.fuelIApplicationインタフェースを実装
- 内部的にはCi.nsIXULAppInfo
- アプリケーション、すなわちFirefoxに関する基本的な情報を得ることができる

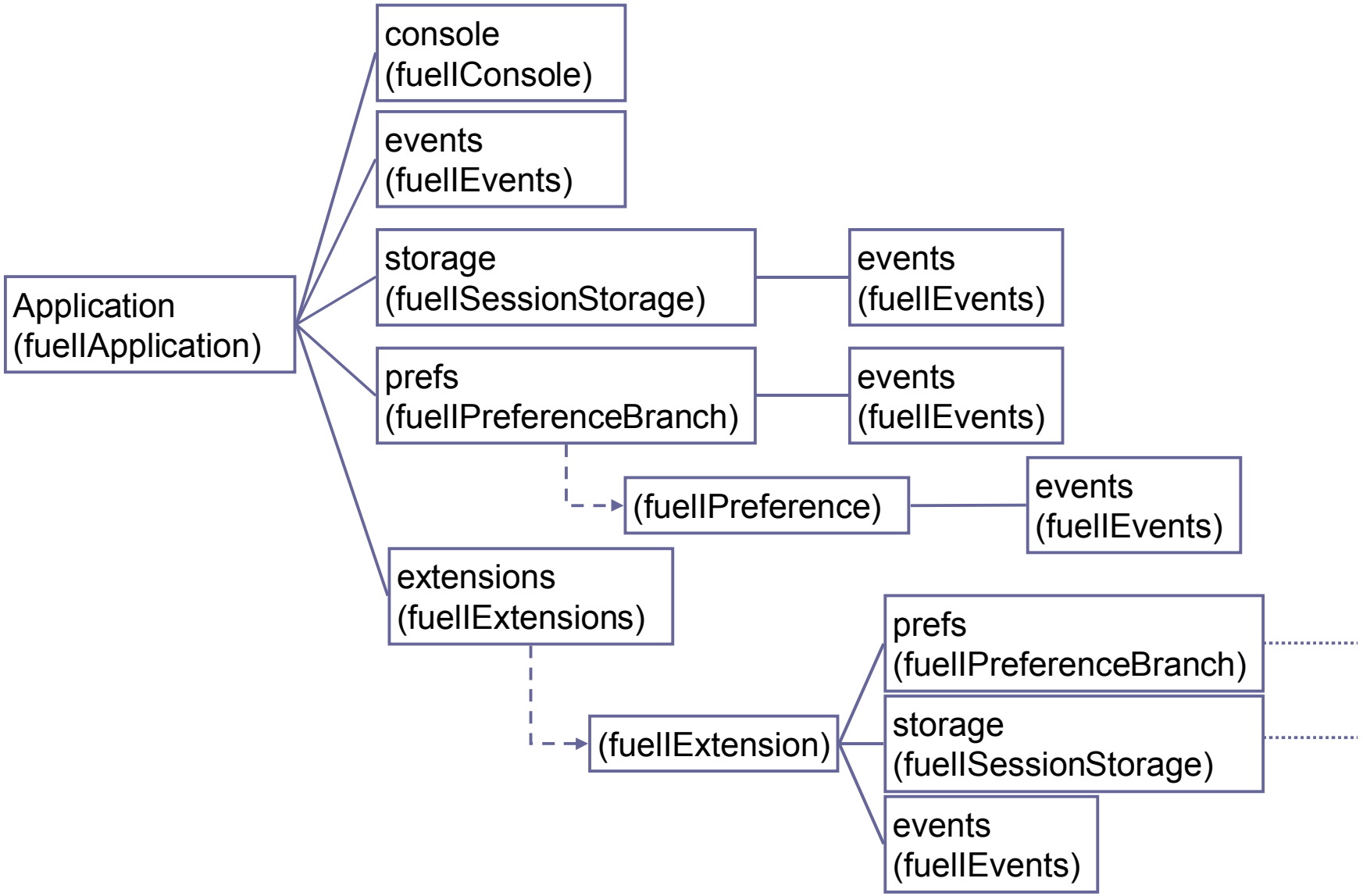
Application使用例

- FirefoxのGUIDを調べたい
  - Application.id
  - {ec8030f7-c20a-464f-9b0e-13a3a9e97384}
- Firefoxのアプリケーション名を調べたい
  - Application.name
  - Firefox
- Firefoxのバージョンを調べたい
  - Application.version
  - 3.0a5pre

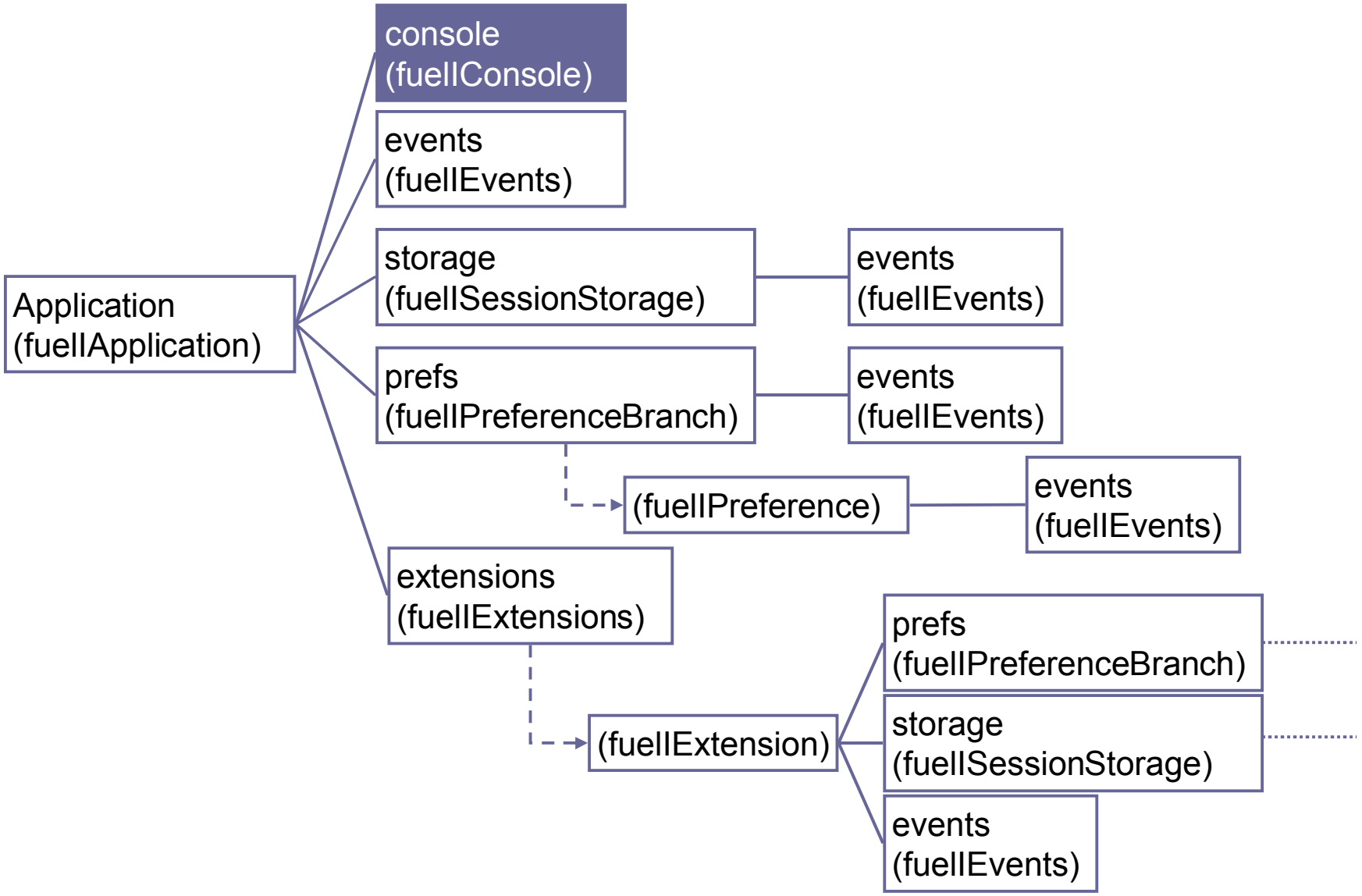
Application

- id, name, versionプロパティはおまけ
- 実際に機能を有するのは以下のプロパティ
  - Application.console
  - Application.events
  - Application.storage
  - Application.prefs
  - Application.extensions

FUEL 0.1系統図


Application
(fuelIApplication)
console
(fuelIConsole)
events
(fuelIEvents)
storage
(fuelISessionStorage)
events
(fuelIEvents)
prefs
(fuelIPreferenceBranch)
events
(fuelIEvents)
(fuelIPreference)
events
(fuelIEvents)
extensions
(fuelIExtensions)
(fuelIExtension)
prefs
(fuelIPreferenceBranch)
storage
(fuelISessionStorage)
events
(fuelIEvents)

Application.console



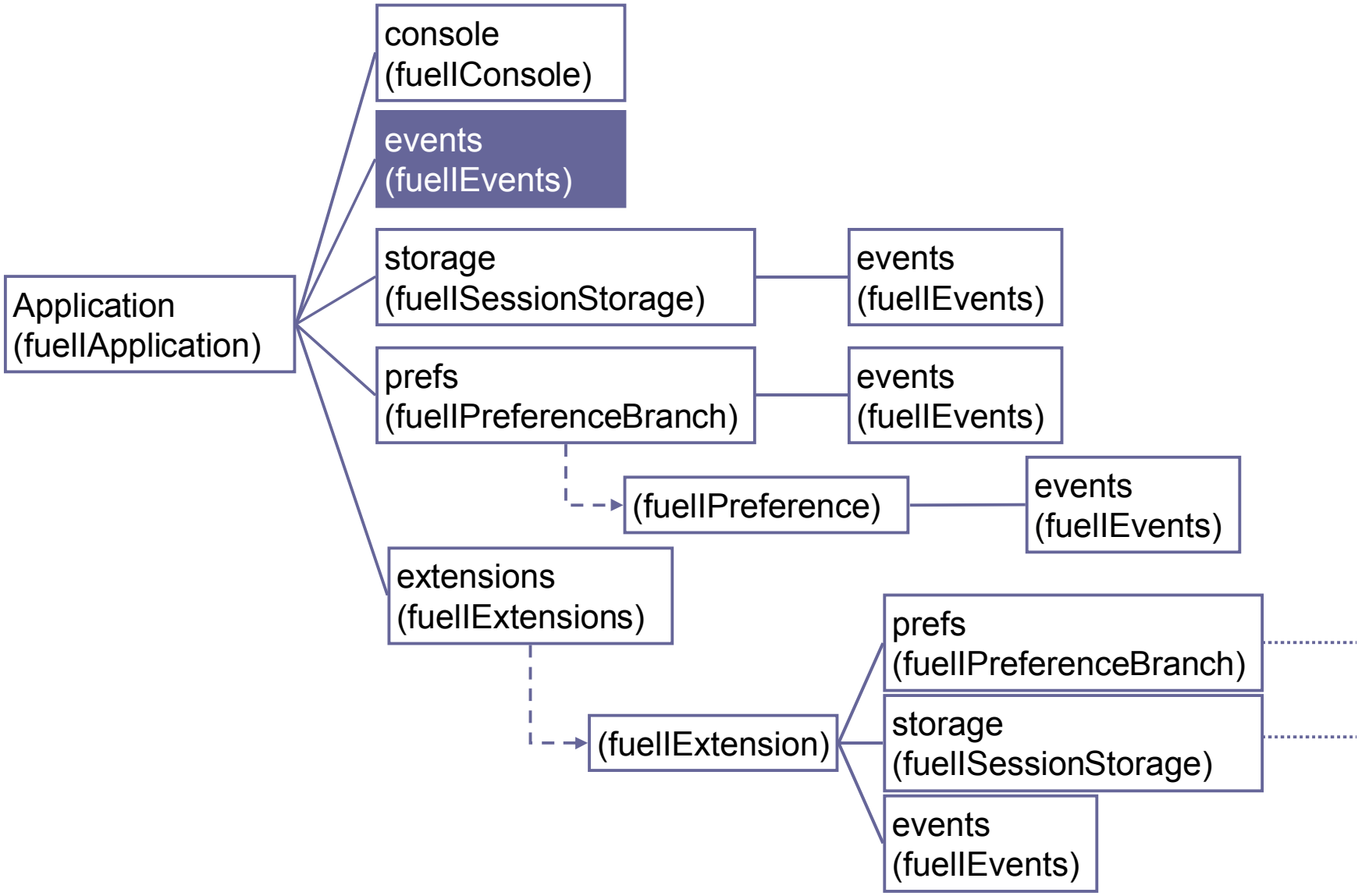
Application.console

- Ci.fuelIConsole型
- 内部的にはCi.nsIConsoleService

Application.consoleの使用例

- エラーコンソールへメッセージを出力する
  - Application.console.log(“こんにちは”);
- エラーコンソールを開く
  - Application.console.open();

Application.events



Application.events

- Ci.fuelIEvents型
- Firefoxの起動や終了を監視できる

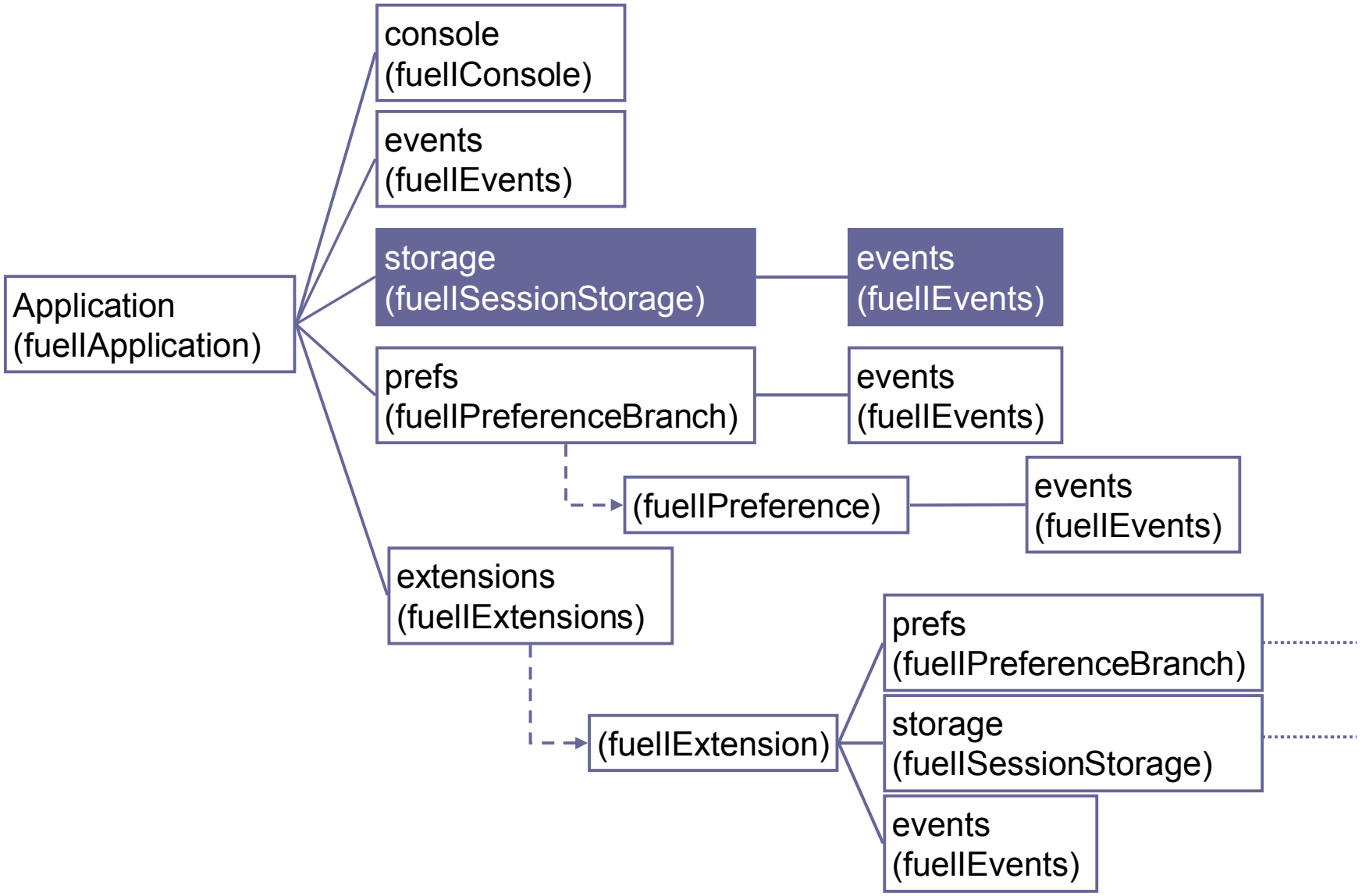
Application.events使用例①

- Firefoxの終了を監視する
  - var listener = function(aEventItem) {
    dump(aEventItem.type + “\n“);
    dump(aEventItem.data + "\n");
    };
    Application.events.addListener("quit", listener);
    Application.events.addListener("unload",listener);
  - aEventItem.type ： “quit” または “unload”
    aEventItem.data ： “application”
- quit : nsIObserverServiceで“quit-application-requested”を監視するのと同等
- unload : nsIObserverServiceで“xpcom-shutdown”を監視するのと同等

Application.events使用例②

- Firefoxの起動を監視する
  - // JS製XPCOMの登録時とかに…
    var app = Cc["@mozilla.org/fuel/application;1"]
    .getService(Ci.fuelIApplication);
    var listener = function(aEventItem) {
    dump(aEventItem.type + "\n”);
    dump(aEventItem.data + "\n");
    };
    app.events.addListener("ready", listener);
  - aEventItem.type ： “ready”
    aEventItem.data ： “application”
- ready : nsIObserverServiceで“final-ui-startup”を監視するのと同等
- readyに対してloadというのもあるが…

Application.storage



Application.storage

- Ci.fuelISessionStorage型
- ≠mozStorage（SQLiteデータベース）
- 内部的にはただのJavaScriptオブジェクト {}
- キー／値によるシンプルな構造
- 永続性はなく、Firefox起動中のみ保持する
  グローバルなデータベースとして使える

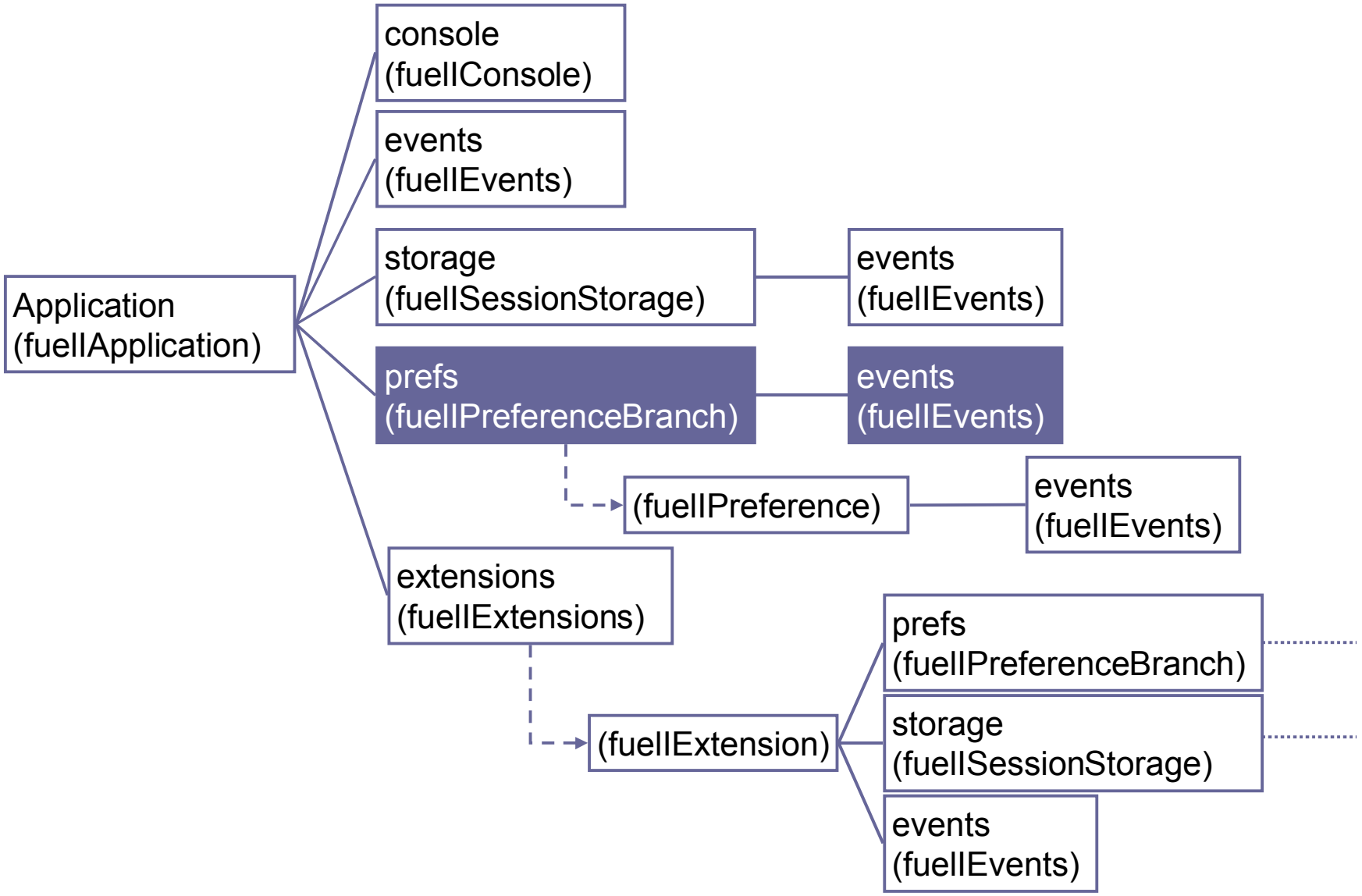
Application.storage使用例①

- 値のセット
  - Application.storage.set(“KEY”, “VALUE”);
  - 内部的には_storage[“KEY”] = “VALUE”
- 値のゲット
  - Application.storage.get(“KEY”, “DEFVAL”);
  - 内部的にはreturn _storage[“KEY”] || “DEFVAL”
- キーが存在するか調べる
  - Application.storage.has(“KEY”);
  - 内部的にはreturn _storage.hasOwnProperty(“KEY”)

Application.storage使用例②

- eventsプロパティはCi.fuelIEvents型
- 値の変更を監視する
  - var listener = function(aEventItem) {
    Application.console.log(aEventItem.type);
    Application.console.log(aEventItem.data);
    };
    Application.storage.events.addListener("change", listener);
  - aEventItem.type：“change”
    aEventItem.data：変更されたキー
- 監視をやめる
  - Application.storage.events.removeListener("change", listener);

Application.prefs



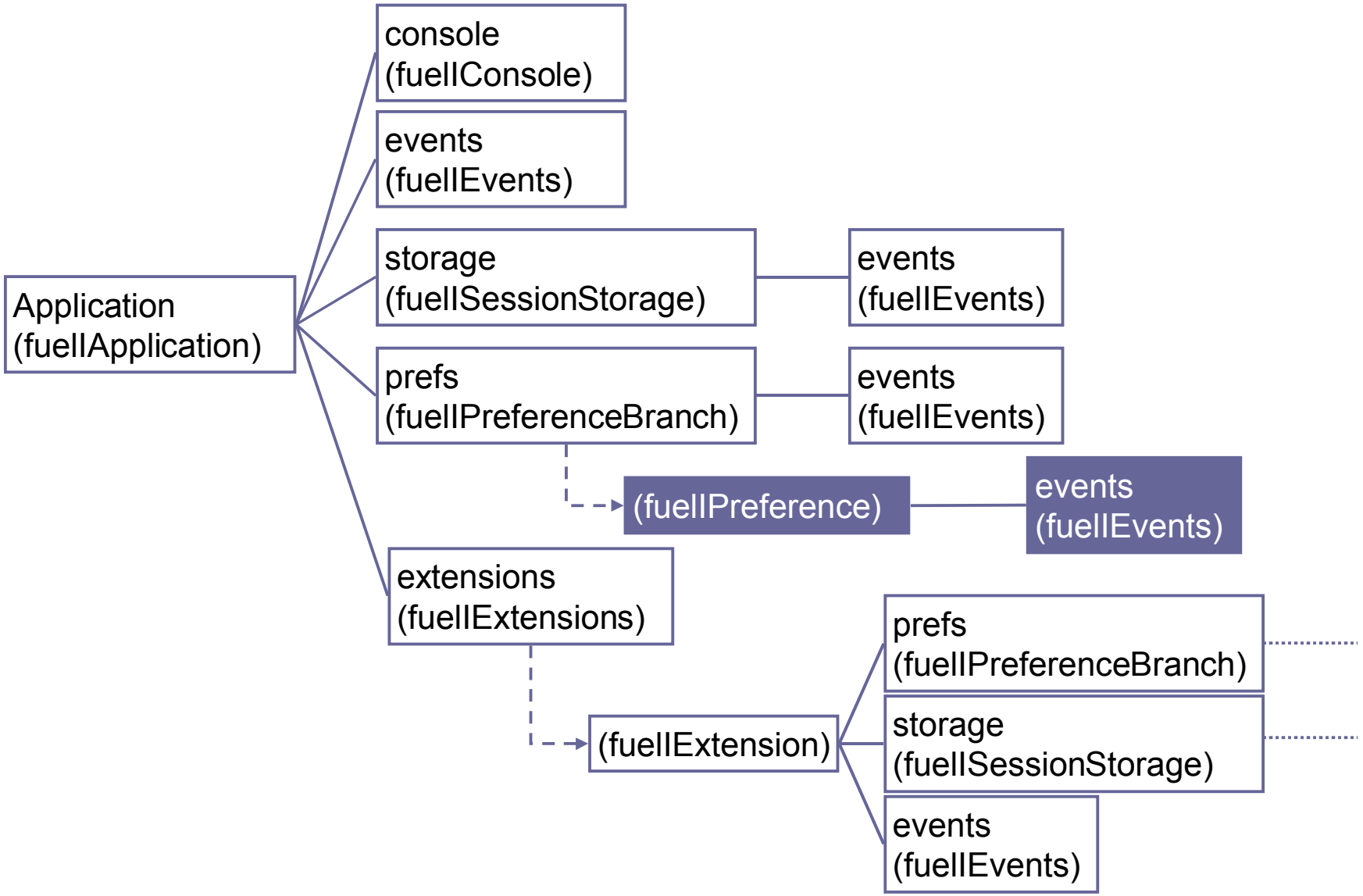
Application.prefs

- Ci.fuelIPreferenceBranch型
- 内部的にはCi.nsIPrefBranch2など

Application.prefs使用例①

- 設定値を取得する
  - Application.prefs.getValue("general.useragent.locale", "???")
  - en-US
- 設定値を変更する
  - Application.prefs.setValue("general.useragent.locale", "ja-JP")
- 個々の設定値の詳細を得る
  - var p = Application.prefs.get(“general.useragent.locale”);
  - pはCi.fuelIPreference型オブジェクト

Ci.fuelIPreferenceインタフェース



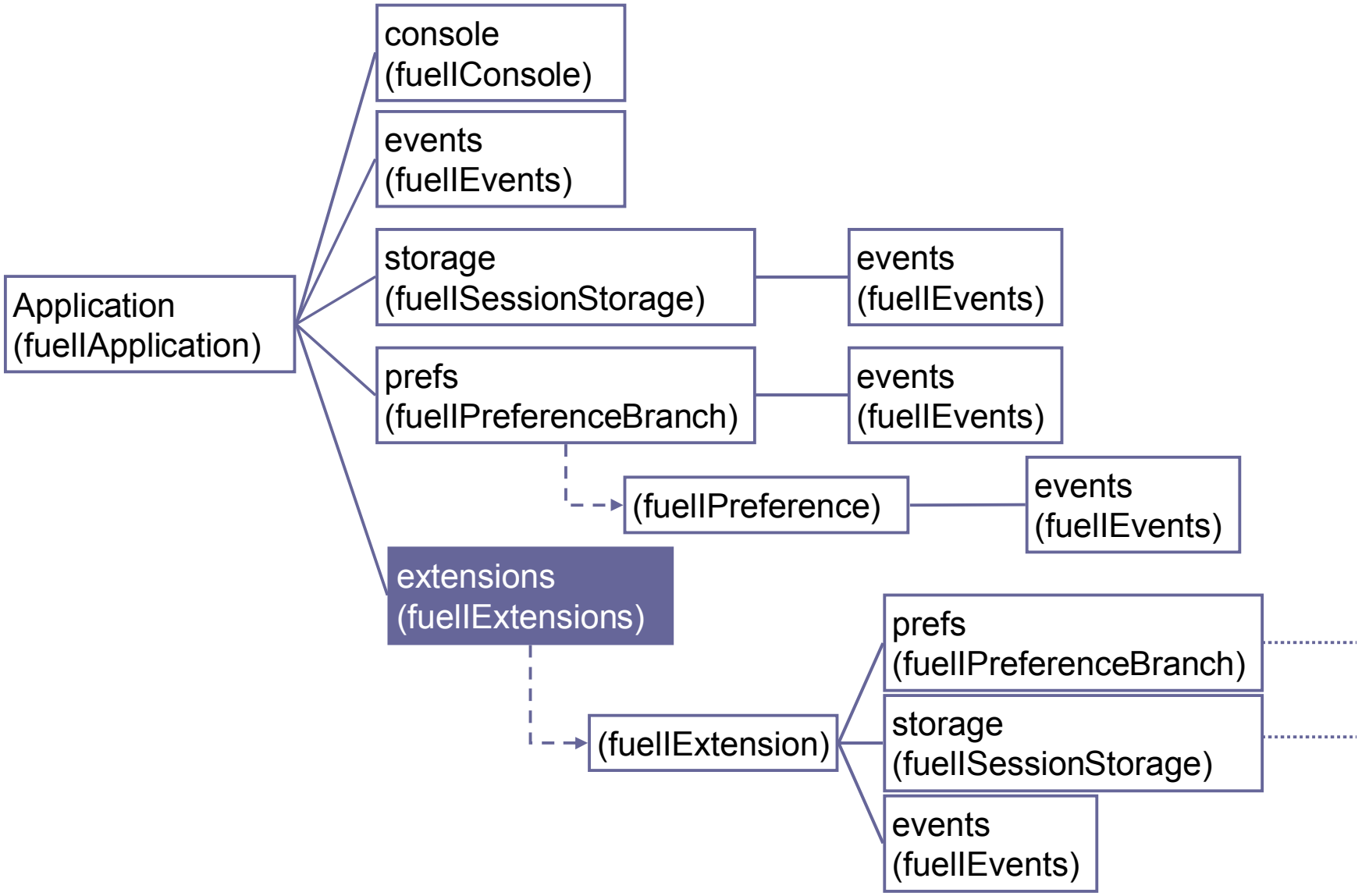
Ci.fuelIPreference使用例①

- 設定値を取得する
  - p.value
  - en-US
- 設定値が変更されているかどうか調べる
  - p.modified
  - true
- 設定値をデフォルト値へ戻す
  - p.reset()

Ci.fuelIPreference使用例②

- eventsプロパティはCi.fuelIEvents型
- 設定値の変更を監視する
  - var listener = function(aEventItem) {
    Application.console.log(aEventItem.type);
    Application.console.log(aEventItem.data);
    };
    p.events.addListener("change", listener);
  - aEventItem.type : “change”
    aEventItem.data : “general.useragent.locale”
- 監視をやめる
  - p.events.removeListener("change", listener);

Application.extensions



Application.extensions

- Ci.fuelIExtensions型
- 内部的にはCi.nsIExtensionManager
- 拡張機能全体

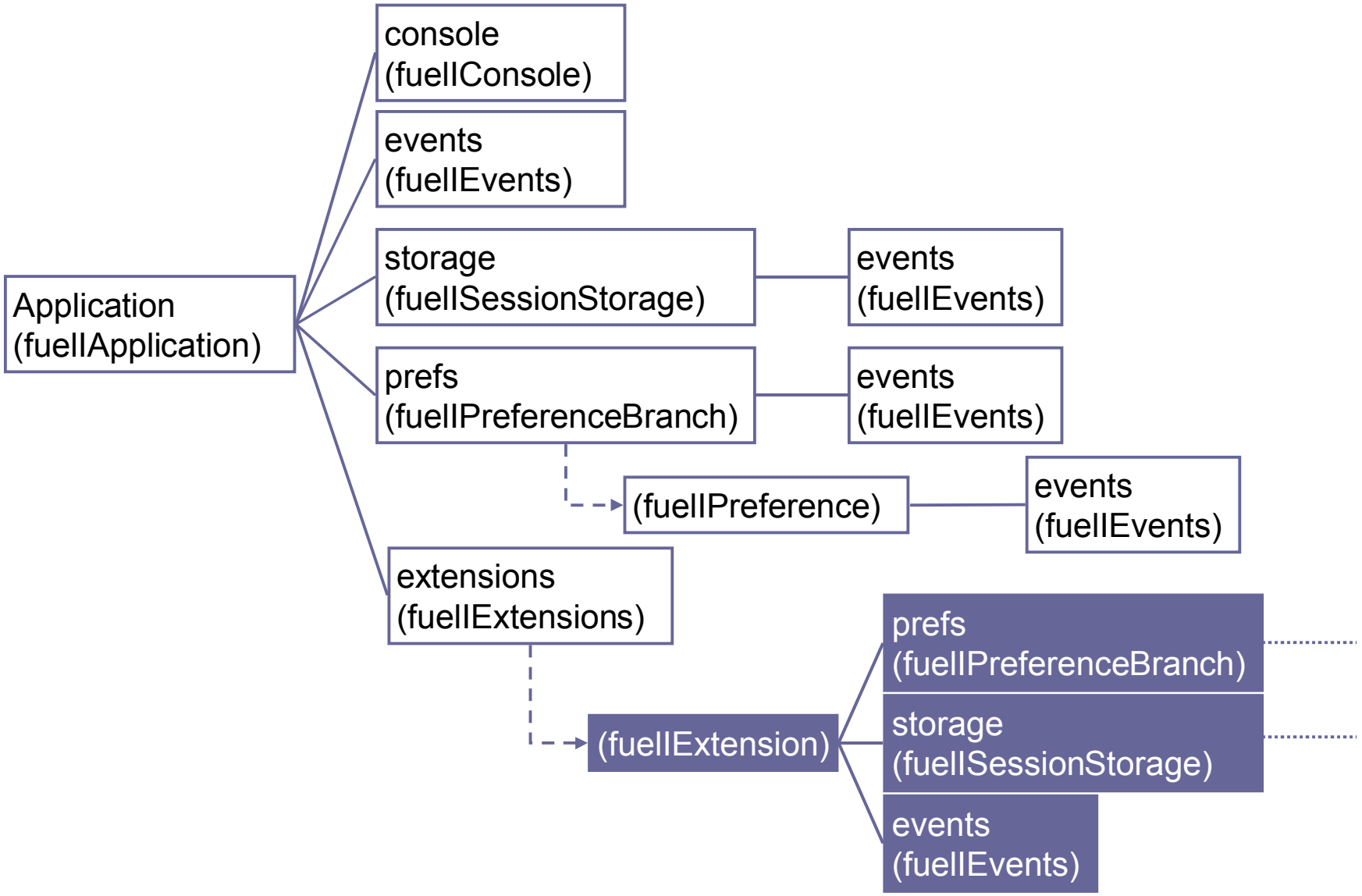
Application.extensions使用例①

- インストールされている拡張機能の一覧を調べたい
  - Application.extensions.all
  - Ci.fuelIExtensionオブジェクトの配列
- インストールされている拡張機能の名前をリストアップする
  - Application.extensions.all.forEach(function(ext) {
    Application.console.log(ext.name);
    });
  - DOM Inspector　Talkback　…

Application.extensions使用例②

- ある拡張機能がインストールされているかどうか調べたい
  - Application.extensions.has("inspector@mozilla.org")
  - true
- ある拡張機能について詳細な情報を得たい
  - Application.extensions.get("inspector@mozilla.org")
  - Ci.fuelIExtension型

Ci.fuelIExtensionインタフェース



Ci.fuelIExtensionインタフェース使用例①

- DOMインスペクタについて調べたい...
  - var ext = Application.extensions.get("inspector@mozilla.org");
- GUIDは？
  - ext.id
  - inspector@mozilla.org
- 名前は？
  - ext.name
  - DOM Inspector
- バージョンは？
  - ext.version
  - 1.9a5pre

Ci.fuelIExtensionインタフェース使用例②

- eventsプロパティはCi.fuelIEvents型
- 拡張機能のアンインストールを監視する
  - var ext = Application.extensions.get("inspector@mozilla.org");
    var listener = function(aEventItem) {
    Application.console.log(aEventItem.type);
    Application.console.log(aEventItem.data);
    };
    ext.events.addListener("uninstall", listener);
  - aEventItem.type : ”uninstall”
    aEventItem.data : inspector@mozilla.org
  - 厳密にはアンインストールしようとした時に発生

Ci.fuelIExtensionインタフェース使用例③

- prefsプロパティはCi.fuelIPreferenceBranch型
- 拡張機能固有のPreference Branchとして使える
- 拡張機能固有の設定値をリストアップ
  - var ext = Application.extensions.get("inspector@mozilla.org");
    ext.prefs.all.forEach(function(pref) {
    　Application.console.log(pref.name + ” = ” + pref.value);
    });
- あくまでも「extensions.*%GUID%*.」に対するPreference Branchであることに注意
  - ext.prefs.root
  - extensions.inspector@mozilla.org.

Ci.fuelIExtensionインタフェース使用例④

- storageプロパティはCi.fuelISessionStorage型
- 拡張機能固有のセッションストレージとして使える
  - var ext = Application.extensions.get("inspector@mozilla.org");
    ext.storage.set(”foo”, ”bar”);
    ext.storage.get(”foo”, null);

FUELの今後の展望

- FUEL 0.2 以降は実用性あり
  - タブブラウザ (Application.browser)
  - ブックマーク (Application.bookmarks)
- 私の要望
  - ファイル読み書き (Application.files?)
  - クリップボード (Application.clipboard?)
  - mozStorage (Application.database?)

FUELについての疑問

- 個人的には、Firefox 2でも使えるようになってくれないと意味がないと思う
- 今後、Firefox 2本体にFUELが搭載され、使えるようになるのでしょうか？Markさん教えてください！