Flapjax: A Programming Language for Ajax Applications

Zagadnienia Programowania Obiektowego

Mateusz Kopeć

Wydział Matematyki, Informatyki i Mechaniki
Uniwersytet Warszawski

10.5.2010

Temat prezentacji

Temat:

- Flapjax: A Programming Language for Ajax Applications
- Flapjax - język programowania do zastosowań w aplikacjach AJAX-owych.

Autorzy:

- Leo A. Meyerovich - University of California, Berkeley
- Gregory H. Cooper - Google
- Jacob Baskin - Google
- Arjun Guha - Brown University
- Shriram Krishnamurthi - Brown University
- Michael Greennberg - University of Pennsylvania
- Aleks Bromfield - Microsoft

Temat prezentacji

Temat:

- Flapjax: A Programming Language for Ajax Applications
- Flapjax - język programowania do zastosowań w aplikacjach AJAX-owych.

Autorzy:

- Leo A. Meyerovich - University of California, Berkeley
- Gregory H. Cooper - Google
- Jacob Baskin - Google
- Arjun Guha - Brown University
- Shriram Krishnamurthi - Brown University
- Michael Greennberg - University of Pennsylvania
- Aleks Bromfield - Microsoft

Spis treści

Spis treści

Czym jest Flapjax?

1. Cechy współczesnych aplikacji internetowych:
   - bogate, interaktywne interfejsy
   - intensywna komunikacja z wieloma serwerami
2. Cechy Flapjaxa:
   - Abstrakcja komunikacji wewnątrz programu i z zewnętrznymi serwisami
   - Wysoka reaktywność - wykrywanie zależności
3. Dwa możliwe podejścia:
   - Język programowania, kompilowany do JavaScriptu
   - Biblioteka JavaScriptowa

Czym jest Flapjax?

1. Cechy współczesnych aplikacji internetowych:
   - bogate, interaktywne interfejsy
   - intensywna komunikacja z wieloma serwerami
2. Cechy Flapjaxa:
   - Abstrakcja komunikacji wewnątrz programu i z zewnętrznymi serwisami
   - Wysoka reaktywność - wykrywanie zależności
3. Dwa możliwe podejścia:
   - Język programowania, kompilowany do JavaScriptu
   - Biblioteka JavaScriptowa

Czym jest Flapjax?

1. Cechy współczesnych aplikacji internetowych:
   - bogate, interaktywne interfejsy
   - intensywna komunikacja z wieloma serwerami
2. Cechy Flapjaxa:
   - Abstrakcja komunikacji wewnątrz programu i z zewnętrznymi serwisami
   - Wysoka reaktywność - wykrywanie zależności
3. Dwa możliwe podejścia:
   - Język programowania, kompilowany do JavaScriptu
   - Biblioteka JavaScriptowa

Decyzje projektowe

- Cechy Javascriptu:
  - Dostępność, popularność,
  - DOM (Document Object Model),
  - XMLHttpRequest.
- Kompilator - Haskell, OpenSource

Decyzje projektowe

- Cechy Javascriptu:
  - Dostępność, popularność,
  - DOM (Document Object Model),
  - XMLHttpRequest.
- Kompilator - Haskell, OpenSource

Zmiany, zmiany, zmiany...

1. Źródła zmian na stronie:
   - startowe wartości z serwerów
   - akcje użytkownika (kliknięcia, ruchy myszy)
   - zmiany w strumieniach danych, jak kanały rss
   - zmiany wprowadzone w równoległej sesji
   - odpowiedzi z serwerów (np. na żądanie zapisu)
   - zmiany w polityce kontroli dostępu
2. Flapjaxowe typy danych:
   - Behavior (zachowanie)
   - Event Stream (strumień zdarzeń)
3. Przejdźmy do przykładu...

Zmiany, zmiany, zmiany...

1. Źródła zmian na stronie:
   - startowe wartości z serwerów
   - akcje użytkownika (kliknięcia, ruchy myszy)
   - zmiany w strumieniach danych, jak kanały rss
   - zmiany wprowadzone w równoległej sesji
   - odpowiedzi z serwerów (np. na żądanie zapisu)
   - zmiany w polityce kontroli dostępu
2. Flapjaxowe typy danych:
   - Behavior (zachowanie)
   - Event Stream (strumień zdarzeń)
3. Przejdźmy do przykładu...

Zmiany, zmiany, zmiany...

1. Źródła zmian na stronie:
   - startowe wartości z serwerów
   - akcje użytkownika (kliknięcia, ruchy myszy)
   - zmiany w strumieniach danych, jak kanały rss
   - zmiany wprowadzone w równoległej sesji
   - odpowiedzi z serwerów (np. na żądanie zapisu)
   - zmiany w polityce kontroli dostępu
2. Flapjaxowe typy danych:
   - Behavior (zachowanie)
   - Event Stream (strumień zdarzeń)
3. Przejdźmy do przykładu...

Spis treści

Przykład 1. - zegar

Wersja JavaScript

```
var timerID = null; var elapsedTime = 0;
function doEverySecond() {
  elapsedTime += 1;
  document.getElementById("curTime").innerHTML =
    = elapsedTime;
}
function startTimer() {
  timerId = setInterval("doEverySecond()", 1000);
}
function resetElapsed() { elapsedTime = 0; }
<body onload="startTimer()">
 <input id="reset" type="button" value="Reset"
    onclick="resetElapsed()"/>
 <div id="curTime"></div></body>
```

Przykład 1. - zegar

Wersja Flapjax

```
var nowB = timerB(1000);
var startTm = nowB.valueNow();
var clickTmsB = $E("reset","click").snapshotE(nowB).
  .startsWith(startTm);
var elapsedB = nowB - clickTmsB;

insertValueB(elapsedB, "curTime", "innerHTML");

<body onload="loader()">
  <input id="reset" type="button" value="Reset"/>
  <div id="curTime"> </div>
</body>
```

Przykład 2. - Gmail

Pierwsze podejście

```
function mkSaveBox(t) {
  var draftBox = // make a <textarea>
  setInterval("...XMLHttpRequest...", t*1000);
  return draftBox;
}
```

Drugie podejście

```
function mkSaveBox(t, btn) {
  var draftBox = // make a <textarea>
  // zapis co t sekund lub po wduszeniu przycisku
  return draftBox;
}
```

Przykład 2. - Gmail

Pierwsze podejście

```
function mkSaveBox(t) {
  var draftBox = // make a <textarea>
  setInterval("...XMLHttpRequest...", t*1000);
  return draftBox;
}
```

Drugie podejście

```
function mkSaveBox(t, btn) {
  var draftBox = // make a <textarea>
  // zapis co t sekund lub po wduszeniu przycisku
  return draftBox;
}
```

Przykład 2. - Gmail

Wersja Flapjax

```
function mkSaveBox(whenE) {
  var draftBox = // make a <textarea>
  // zapis po zdarzeniu na strumieniu whenE
  return draftBox;
}
```

Przykłady

```
mkSaveBox(timerE(60000))
mkSaveBox($E(btn, "click"))
mkSaveBox(mergeE(timerE(60000), $E(btn, "click")))
```

Przykład 2. - Gmail

Wersja Flapjax

```
function mkSaveBox(whenE) {
  var draftBox = // make a <textarea>
  // zapis po zdarzeniu na strumieniu whenE
  return draftBox;
}
```

Przykłady

```
mkSaveBox(timerE(60000))
mkSaveBox($E(btn, "click"))
mkSaveBox(mergeE(timerE(60000), $E(btn, "click")))
```

Przykład 2. - Gmail

Uzupełniona wersja

```
function mkSaveBox(whenE) {
  var draftBox = TEXTAREA();
  var requestsE = whenE.snapshotE($B(draftBox)).
    .mapE(makeRequest);
  var savedE = getWebServiceObjectE(requestE);
  return draftBox;
}
```

Potrzebna funkcja

```
function makeRequest(v) {
  return { url: "/saveValue", fields: {value: v},
           request: "post" }; }
```

Przykład 2. - Gmail

Uzupełniona wersja

```
function mkSaveBox(whenE) {
  var draftBox = TEXTAREA();
  var requestsE = whenE.snapshotE($B(draftBox)).
    .mapE(makeRequest);
  var savedE = getWebServiceObjectE(requestE);
  return draftBox;
}
```

Potrzebna funkcja

```
function makeRequest(v) {
  return { url: "/saveValue", fields: {value: v},
           request: "post" }; }
```

Przykład 2. - Gmail

Style CSS

```
var changedE = $B(draftBox).changes();

styleE = mergeE( changedE.constantE("unsaved"),
                 savedE.constantE("saved") );

styleB = styleE.startsWith("saved");

insertValueB(styleB, draftBox, "className");
```

Przykład 3. - Flickr

Flickr

```
// flickrSearchRequest :: String -> Request
function flickrSearchRequest(req) { ... }

// flickrSearchResponse :: Response -> Listof(Url)
function flickrSearchResponse(resp) { ... }

// makeImg :: Url -> Element
function makeImg(url) { return IMG({ src: url }); }

var queryE=$B("search").changes().calmE(1000);
var requestE=queryE.mapE(flickrSearchRequest);
var responseE=getForeignWebServiceObjectE(requestE).
  .mapE(flickrSearchReponse);
var imgs=DIV(map(makeImg,responseE.startsWith([])));
```

Przykład 4. - Google Geocoder

Google Geocoder

```
//     EventStream {data: a, loc: String }
// -> EventStream {data: a, point: Point or false}
function makeGoogleGeocoderE(requestE) {
 var geocoder = new google.maps.ClientGeocoder();
 var resultE = receiverE(); // primitive stream
 var callback = function(d) {
    return function(p) { resultE.sendEvent(
      {data:d,point:p}) }};

 requestE.mapE(function(req) {
   geocoder.getLatLng(req.loc,callback(req.data));});

 return resultE;
};
```

Przechowywanie danych na serwerze

Persistent Objects

```
writePersistentObject(saveE.snapshot($B('theText')),
                      {path:['draftFT']});

readPersistentObject({path:['draftFT'], initial:''});
```

Kontrola dostępu

```
permsB = readPermissionsB(["input"]);
INPUT({ type:"text",
        disabled:!(permsB.has("WRITE", true)) });
```

Przechowywanie danych na serwerze

Persistent Objects

```
writePersistentObject(saveE.snapshot($B('theText')),
                      {path:['draftFT']});

readPersistentObject({path:['draftFT'], initial:''});
```

Kontrola dostępu

```
permsB = readPermissionsB(["input"]);
INPUT({ type:"text",
        disabled:!(permsB.has("WRITE", true)) });
```

Spis treści

Grafy

- Grafy zależności - acykliczne, skierowane
- Węzeł - strumień zdarzeń (event stream)
- Schemat działania:
  - Otrzymanie zdarzenia e
  - Modyfikacja za pomocą funkcji f
  - Przesłanie do węzłów zależnych f(e)
- Węzeł ma 3 atrybuty:
  - sources :: listof(EventStream)
  - sinks :: listof(EventStream)
  - update :: a -> (b or StopValue)
- Zachowanie (behavior) = strumień zdarzeń + wartość początkowa + aktualna wartość
- Propagacja - porządek topologiczny

Grafy

- Grafy zależności - acykliczne, skierowane
- Węzeł - strumień zdarzeń (event stream)
- Schemat działania:
  - Otrzymanie zdarzenia e
  - Modyfikacja za pomocą funkcji f
  - Przesłanie do węzłów zależnych f(e)
- Węzeł ma 3 atrybuty:
  - sources :: listof(EventStream)
  - sinks :: listof(EventStream)
  - update :: a -> (b or StopValue)
- Zachowanie (behavior) = strumień zdarzeń + wartość początkowa + aktualna wartość
- Propagacja - porządek topologiczny

Grafy

- Grafy zależności - acykliczne, skierowane
- Węzeł - strumień zdarzeń (event stream)
- Schemat działania:
  - Otrzymanie zdarzenia e
  - Modyfikacja za pomocą funkcji f
  - Przesłanie do węzłów zależnych f(e)
- Węzeł ma 3 atrybuty:
  - sources :: listof(EventStream)
  - sinks :: listof(EventStream)
  - update :: a -> (b or StopValue)
- Zachowanie (behavior) = strumień zdarzeń + wartość początkowa + aktualna wartość
- Propagacja - porządek topologiczny

Grafy

- Grafy zależności - acykliczne, skierowane
- Węzeł - strumień zdarzeń (event stream)
- Schemat działania:
  - Otrzymanie zdarzenia e
  - Modyfikacja za pomocą funkcji f
  - Przesłanie do węzłów zależnych f(e)
- Węzeł ma 3 atrybuty:
  - sources :: listof(EventStream)
  - sinks :: listof(EventStream)
  - update :: a -> (b or StopValue)
- Zachowanie (behavior) = strumień zdarzeń + wartość początkowa + aktualna wartość
- Propagacja - porządek topologiczny

Grafy

- Grafy zależności - acykliczne, skierowane
- Węzeł - strumień zdarzeń (event stream)
- Schemat działania:
  - Otrzymanie zdarzenia e
  - Modyfikacja za pomocą funkcji f
  - Przesłanie do węzłów zależnych f(e)
- Węzeł ma 3 atrybuty:
  - sources :: listof(EventStream)
  - sinks :: listof(EventStream)
  - update :: a -> (b or StopValue)
- Zachowanie (behavior) = strumień zdarzeń + wartość początkowa + aktualna wartość
- Propagacja - porządek topologiczny

Kompilator

Z kompilatorem

```
TimerB(1000) + 1
```

Bez kompilatora

```
liftB(function(t) {return t+1;}, timerB(1000));
```

Poniżej 10 lift/KLOC...

Z kompilatorem

```
<input disabled={! !ccNumValidB !}/>
```

Kompilator

Z kompilatorem

```
TimerB(1000) + 1
```

Bez kompilatora

```
liftB(function(t) {return t+1;}, timerB(1000));
```

Poniżej 10 lift/KLOC...

Z kompilatorem

```
<input disabled={! !ccNumValidB !}/>
```

Kompilator

Z kompilatorem

```
TimerB(1000) + 1
```

Bez kompilatora

```
liftB(function(t) {return t+1;}, timerB(1000));
```

Poniżej 10 lift/KLOC...

Z kompilatorem

```
<input disabled={! !ccNumValidB !}/>
```

Kompilator

Z kompilatorem

```
TimerB(1000) + 1
```

Bez kompilatora

```
liftB(function(t) {return t+1;}, timerB(1000));
```

Poniżej 10 lift/KLOC...

Z kompilatorem

```
<input disabled={! !ccNumValidB !}/>
```

Inne cechy języka

Funkcyjność

```
var name = calmE(changes($B("name")), 300);
```

Obiektowość

```
var name = $B("name").changes().calmE(300);
```

- Podnoszenie (lifting) stałych (np. `timerB(1000)`)
- Adresowanie przez id
- Modyfikacje w miejscu (np. `DIV (timerB(1000))`)

Inne cechy języka

Funkcyjność

```
var name = calmE(changes($B("name")), 300);
```

Obiektowość

```
var name = $B("name").changes().calmE(300);
```

- Podnoszenie (lifting) stałych (np. `timerB(1000)`)
- Adresowanie przez id
- Modyfikacje w miejscu (np. `DIV (timerB(1000))`)

Inne cechy języka

Funkcyjność

```
var name = calmE(changes($B("name")), 300);
```

Obiektowość

```
var name = $B("name").changes().calmE(300);
```

- Podnoszenie (lifting) stałych (np. `timerB(1000)`)
- Adresowanie przez id
- Modyfikacje w miejscu (np. `DIV (timerB(1000))`)

Inne cechy języka

Funkcyjność

```
var name = calmE(changes($B("name")), 300);
```

Obiektowość

```
var name = $B("name").changes().calmE(300);
```

- Podnoszenie (lifting) stałych (np. `timerB(1000)`)
- Adresowanie przez id
- Modyfikacje w miejscu (np. `DIV (timerB(1000))`)

Inne cechy języka

Funkcyjność

```
var name = calmE(changes($B("name")), 300);
```

Obiektowość

```
var name = $B("name").changes().calmE(300);
```

- Podnoszenie (lifting) stałych (np. `timerB(1000)`)
- Adresowanie przez id
- Modyfikacje w miejscu (np. `DIV (timerB(1000))`)

Spis treści

Więcej przykładów

- Resume `http://resume.cs.brown.edu/`
- Continue 2.0 `http://continue2.cs.brown.edu/`
- Strona projektu `http://flapjax-lang.org/`
  - Kompilator online
  - Dema
  - Dokumentacja

Więcej przykładów

- Resume `http://resume.cs.brown.edu/`
- Continue 2.0 `http://continue2.cs.brown.edu/`
- Strona projektu `http://flapjax-lang.org/`
  - Kompilator online
  - Dema
  - Dokumentacja

Spis treści

Główne cechy Flapjaxa

1. Spójność informacji jako aksjomat języka programowania
2. Możliwość algebraicznego wnioskowania o programach
3. Bezpieczeństwo – prostsza struktura aplikacji pozwala na lepszą jej analizę
4. Odpluskwianie – system typów
5. MVC z kontrolerem realizowanym przez język

Główne cechy Flapjaxa

1. Spójność informacji jako aksjomat języka programowania
2. Możliwość algebraicznego wnioskowania o programach
3. Bezpieczeństwo – prostsza struktura aplikacji pozwala na lepszą jej analizę
4. Odpluskwianie – system typów
5. MVC z kontrolerem realizowanym przez język

Główne cechy Flapjaxa

1. Spójność informacji jako aksjomat języka programowania
2. Możliwość algebraicznego wnioskowania o programach
3. Bezpieczeństwo – prostsza struktura aplikacji pozwala na lepszą jej analizę
4. Odpluskwianie – system typów
5. MVC z kontrolerem realizowanym przez język

Główne cechy Flapjaxa

1. Spójność informacji jako aksjomat języka programowania
2. Możliwość algebraicznego wnioskowania o programach
3. Bezpieczeństwo – prostsza struktura aplikacji pozwala na lepszą jej analizę
4. Odpluskwianie – system typów
5. MVC z kontrolerem realizowanym przez język

Główne cechy Flapjaxa

1. Spójność informacji jako aksjomat języka programowania
2. Możliwość algebraicznego wnioskowania o programach
3. Bezpieczeństwo – prostsza struktura aplikacji pozwala na lepszą jej analizę
4. Odpluskwianie – system typów
5. MVC z kontrolerem realizowanym przez język

Pytania?

Dziękuję.