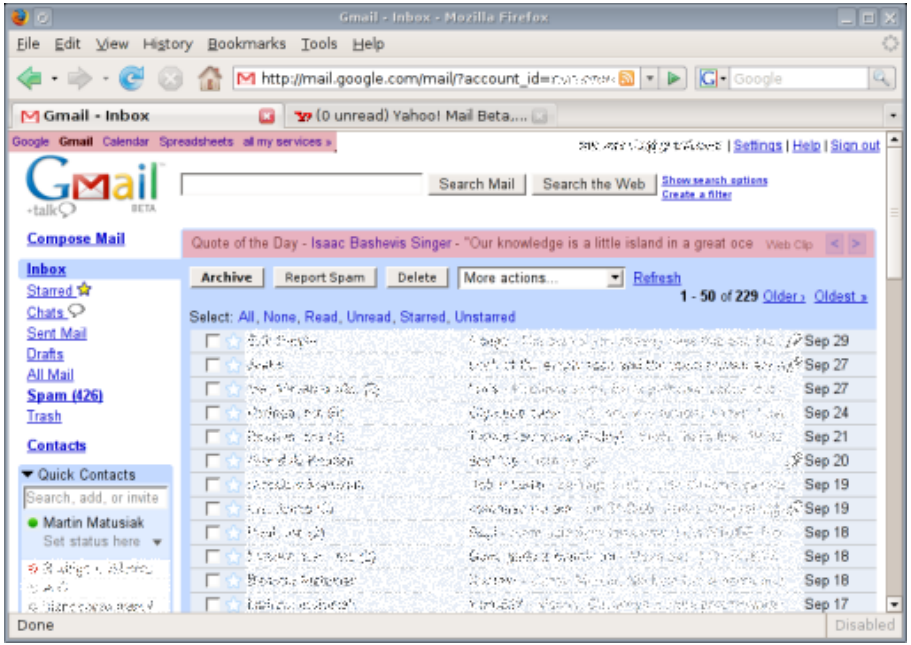
Генерация HTML на стороне сервера

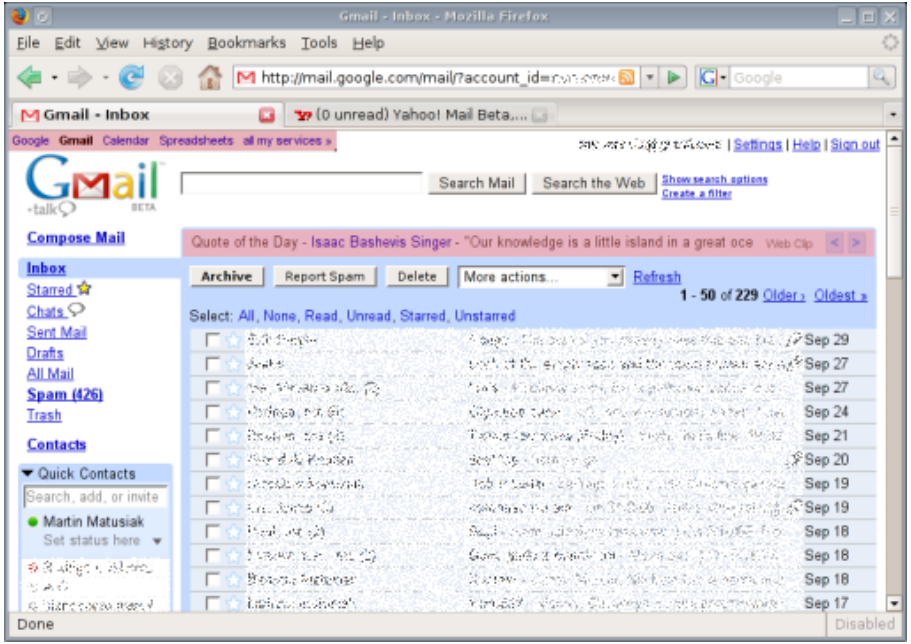
Немного истории

- HTML генерировался на сервере
- На клиенте могло добавляться немного динамики
- Логика сервера и клиента не пересекалась
- С шаблонами никаких проблем (php, smarty)



Немного истории

- Больше динамики на клиенте
- Приходилось дублировать шаблоны для клиента
- Использовались единые шаблоны для сервера и клиента (mustache и подобные)



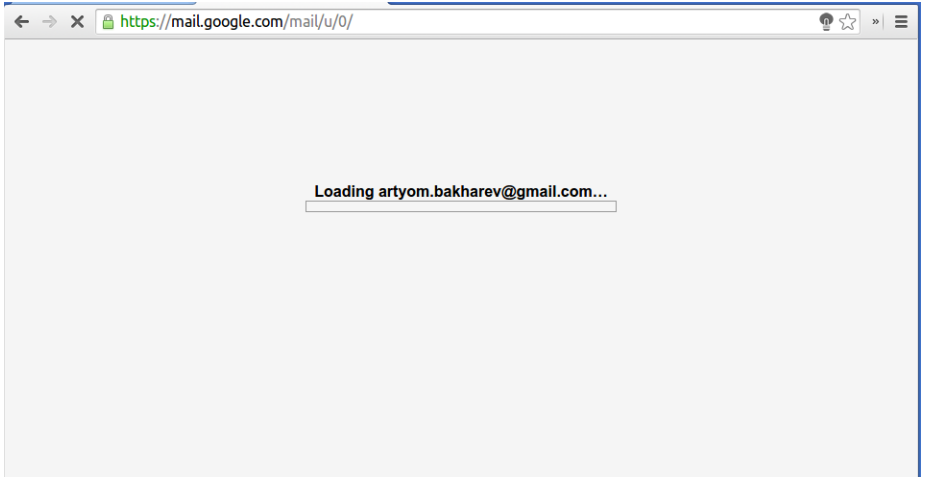
Немного истории

- Клиентская логика начала усложняться, появилась концепция data-binding.
- Не было простых способов использовать data-binding для разметки сгенерированой на сервере.

- Вместо простых шаблонов используются строительные блоки веб-приложения — компоненты.
- Логика инициализации и отрисовки этих компонентов дублировалась на сервере и клиенте.
- Привязывание созданных на клиентне компонентов было нетривиальной задачей

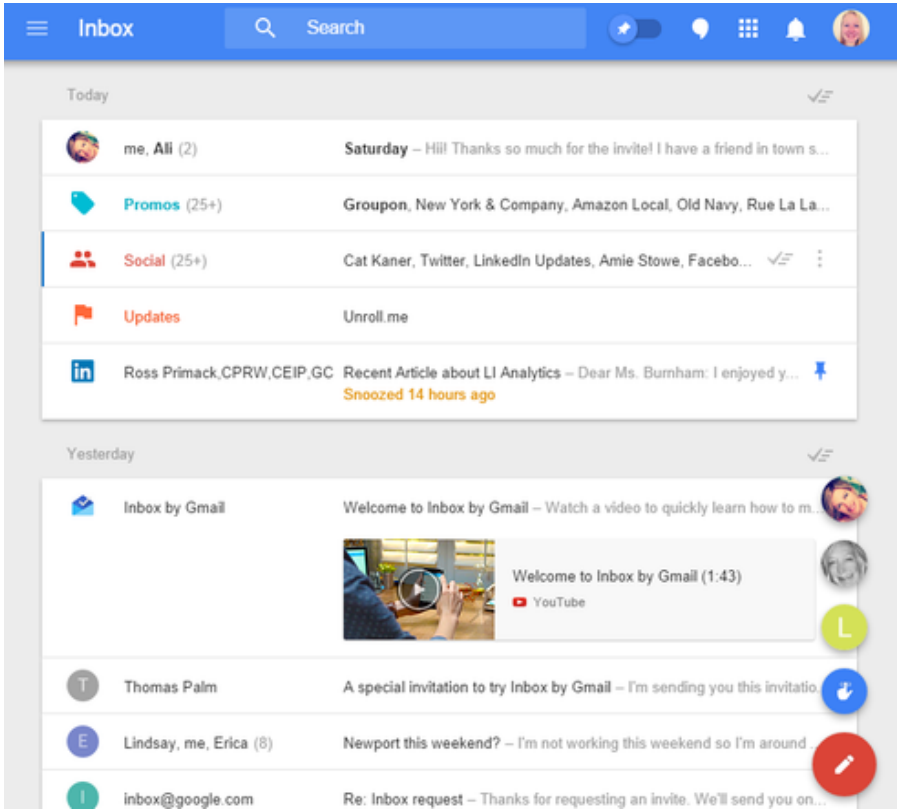
Немного истории

- Многие веб-приложения вообще отказались от генерации разметки на сервере, однако из-за этого страдало SEO и время отображения страницы



Немного истории

- Началось развитие серверного javascript (Node.js)
- Изоморфный код — один код генерации HTML/DOM для сервера и клиента



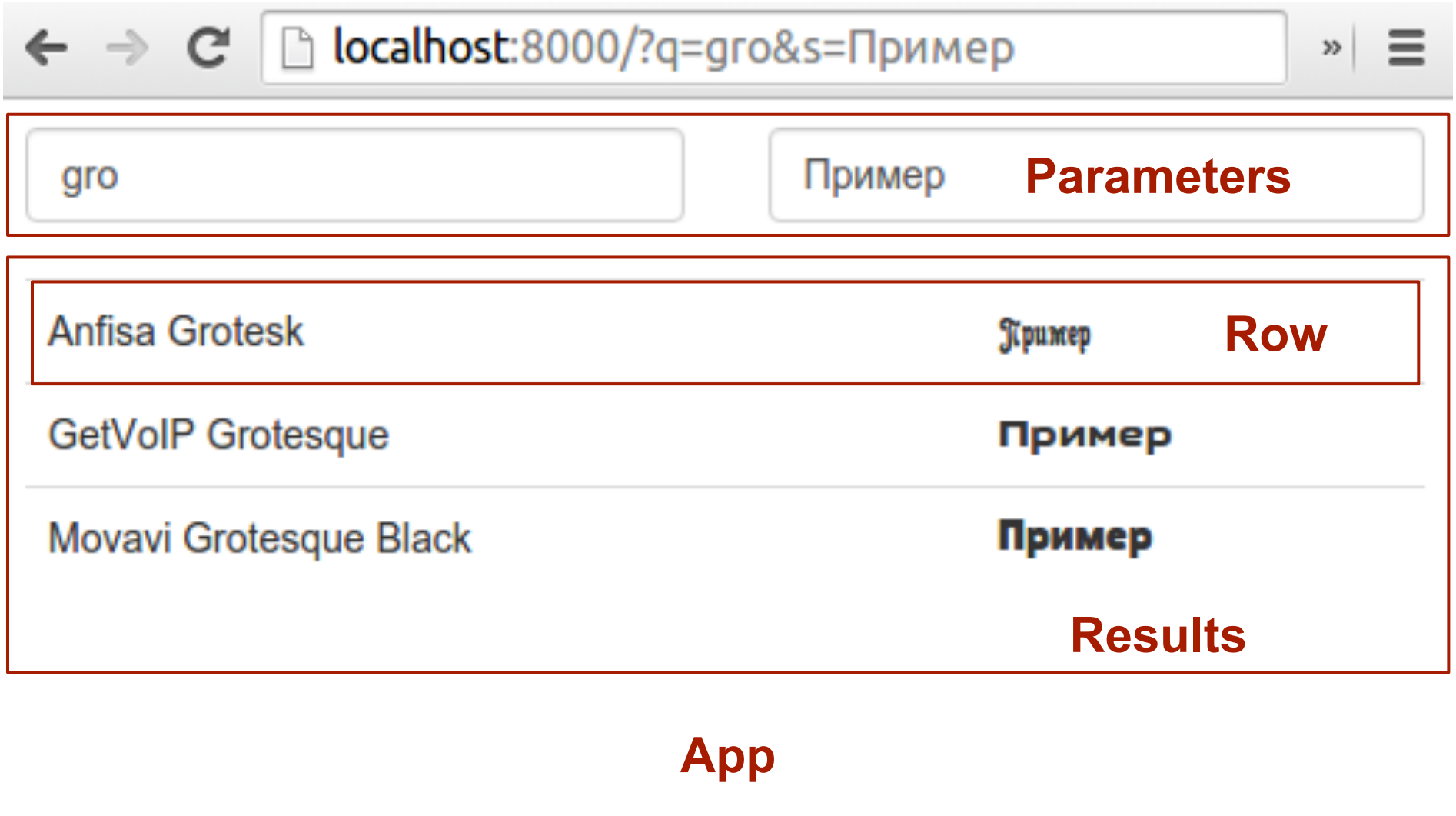
gro

Пример

Anfisa Grotesk — Пример

GetVoIP Grotesque — **Пример**

Movavi Grotesque Black — **Пример**


localhost:8000/?q=gro&s=Пример
gro
Пример
Parameters
Anfisa Grotesk
Пример
Row
GetVoIP Grotesque
Пример
Movavi Grotesque Black
Пример
Results
App

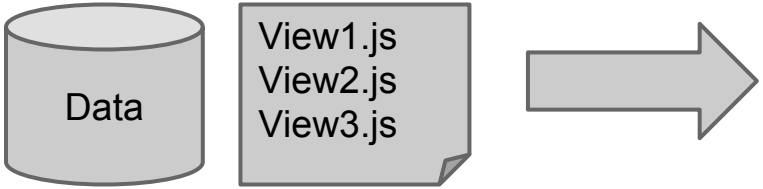
```
//Model
var model = {
 fonts: [
   'Anfisa Grotesk',
   ...
 ],
 query: 'gro',
 str: 'Пример'
};
```

```
<!-- Пример шаблона -->
<tr>
  <td>{font}</td>
  <td style="font-family:
'{font}';">{str}</td>
</tr>
```

```
//App View
function App(){}

//Инициализация данных
App.prototype.init = function(){
 //Выполняем фильтрацию model.fonts
};

//Генерация HTML
App.prototype.render = function(){
 return '<div>...</div>';
};

App.prototype.afterRender = function(){
 //Логика после прикрепления к DOM
 //К примеру, повешать красивый тултип на
строчку
};
```

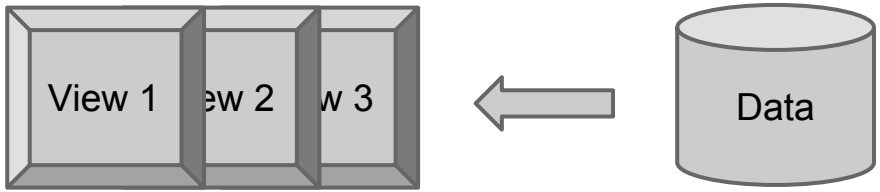
Современная модель изоморфной генерации HTML/DOM

Генерация DOM на клиенте

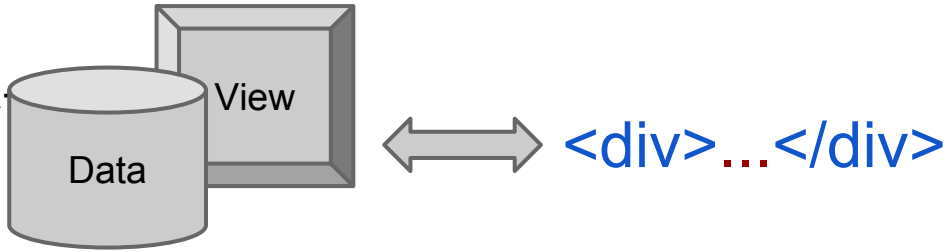


1. Начальные данные (модель) и код представлений (компонентов/виджетов) передаются в браузер

2. На клиенте создаётся иерархия представлений, которые инициализируются моделью



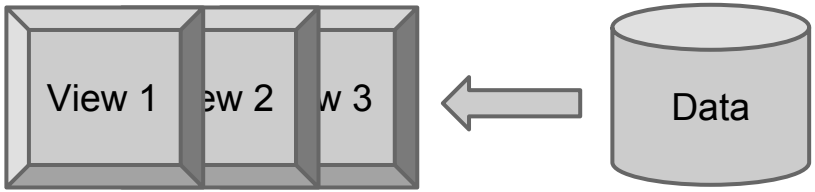
3. Представления генерируют DOM, происходит связывание данных (data-binding)



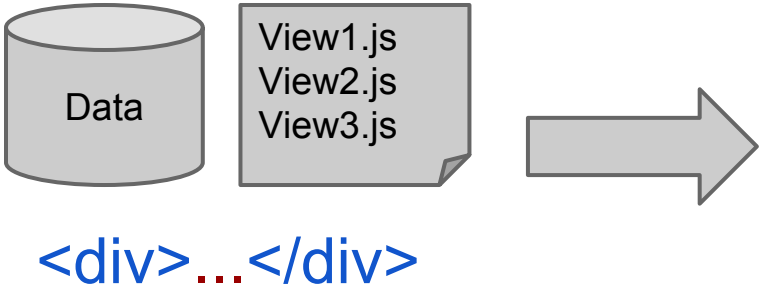
4. Выполняется завершающая стадия инициализации представлений

afterRender()

Генерация HTML на сервере



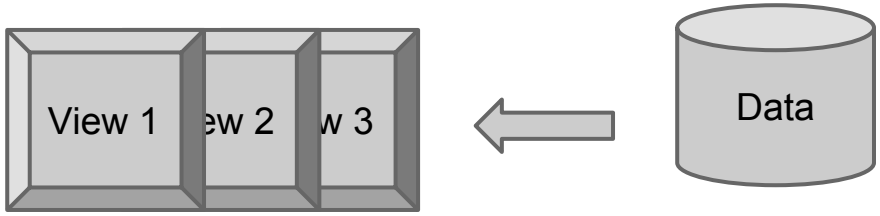
1. На сервере создаётся иерархия представлений, которые инициализируются моделью



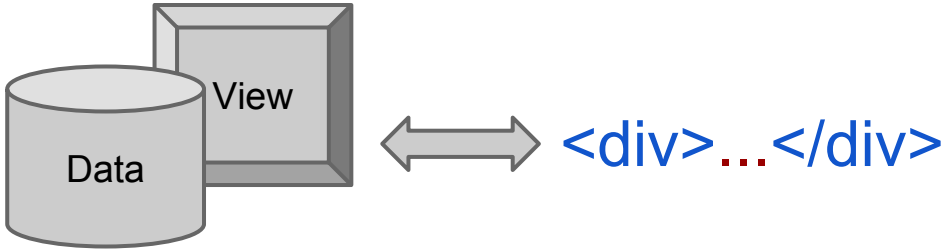
2. Представления генерируют HTML, который отдаётся браузеру. Также браузеру передаются начальные данные и код представлений.

Генерация HTML на сервере

3. На клиенте создаётся такая же иерархия представлений, которые инициализируются моделью.



4. Представления прикрепляются к построенному DOM, происходит связывание данных (data-binding)



5. Выполняется завершающая стадия инициализации представлений

afterRender()

Инструменты для изоморфной генерации HTML/DOM

- DerbyJS
- ReactJS / Redux (Flux)
- RiotJS
- 2gis Slot