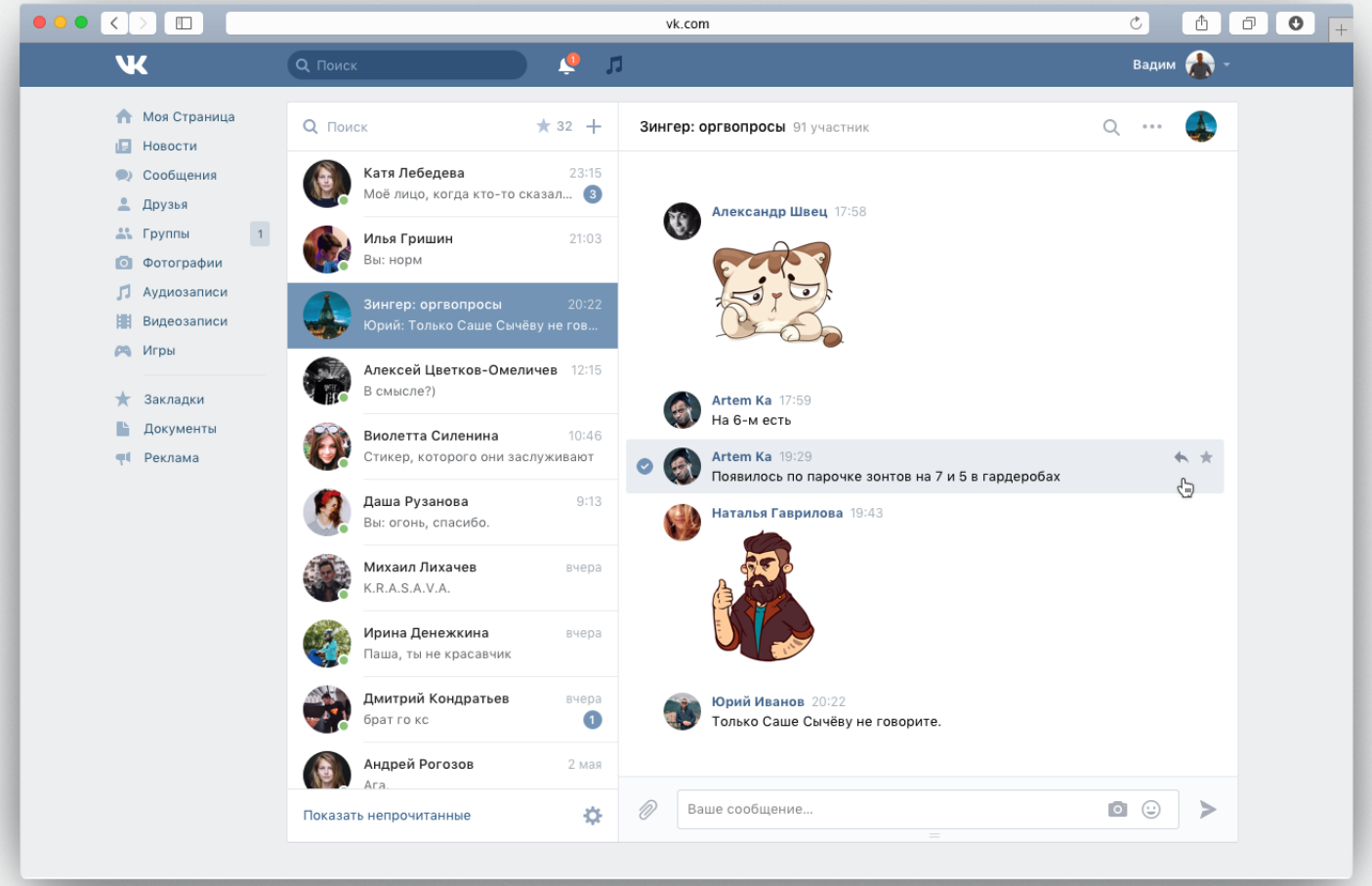
React со скоростью света: не совсем обычный серверный рендеринг

Тимофей Чаптыков

tim.chaptykov@gmail.com vk.com/tim.chaptykov @chaptykov


vk.com
Поиск
Вадим
Моя Страница
Новости
Сообщения
Друзья
Группы
Фотографии
Аудиозаписи
Видеозаписи
Игры
Закладки
Документы
Реклама
Поиск
32
Катя Лебедева 23:15
Моё лицо, когда кто-то сказал...
Илья Гришин 21:03
Вы: норм
Зингер: оргвопросы 20:22
Юрий: Только Саше Сычёву не гов...
Алексей Цветков-Омеличев 12:15
В смысле?)
Виолетта Силенина 10:46
Стикер, которого они заслуживают
Даша Рузанова 9:13
Вы: огонь, спасибо.
Михаил Лихачев вчера
K.R.A.S.A.V.A.
Ирина Денежкина вчера
Паша, ты не красавчик
Дмитрий Кондратьев вчера
брат го кс
Андрей Рогозов 2 мая
Ага
Показать непрочитанные
Зингер: оргвопросы 91 участник
Александр Швец 17:58
Artem Ka 17:59
На 6-м есть
Artem Ka 19:29
Появилось по парочке зонтов на 7 и 5 в гардеробах
Наталья Гаврилова 19:43
Юрий Иванов 20:22
Только Саше Сычёву не говорите.
Ваше сообщение...

Сейчас быстро запилим

- Виртуальная DOM, декларативный рендеринг
- Архитектура с однонаправленным потоком данных
- Высокий «Developer experience»

Первое, что приходит в голову

1. Всё выбросить и переписать.
2. Использовать для рендеринга на сервере Node.js

Большая база кода

- **Шаблоны:** 150 848 LOC
- **Стили:** 1 250 661 LOC

Нагрузка

- **96 000 000** MAU
- **5 000 000 000** отправленных сообщений в день
- **10 000** раз за 5 минут открывается раздел сообщений

Нельзя делать хуже

Текущий технологический стек

Текущий технологический стек

- **Сервер:** шаблоны на kPHP

KittenPHP

KittenPHP (или kPHP) — транслятор PHP подобного кода в C++.

Пример шаблона на kPHP

```
function sImSimpleLink($href, $content) {
  return <<<HTML
    <a href="{$href}" class="_im_header_link">
      {$content}
    </a>
  HTML;
}
```

Текущий технологический стек

- **Сервер:** шаблоны на kPHP
- **Клиент:**
  - Каждый раздел — отдельное клиентское приложение
  - Переиспользование серверных шаблонов
  - Императивная работа с DOM

Модный технологический стек

Client side application

Universal application

ReactDOMServer.renderToString()


12
1
2
3
4
5
6
7
8
9
10
11

Медленно

Ускорение серверного рендеринга React-приложений

Speed up Server Side Rendering

1. Node.js с `NODE_ENV=production`
2. Минифицированная версия React из `react/dist/react.min`
3. Babel-трансформации `constant-elements` и `inline-elements`
4. ES6 классы и Stateless components вместо `React.createClass()`
5. Streaming
6. Cache

Speed up Server Side Rendering

1. Node.js с `NODE_ENV=production`
2. Минифицированная версия React из `react/dist/react.min`
3. Babel-трансформации `constant-elements` и `inline-elements`
4. ES6 классы и Stateless components вместо `React.createClass()`
5. Streaming
6. Cache

Speed up Server Side Rendering

by Sasha Aickin

1000 ms → 60 ms

А если нужно быстрее?

А что, если...

Оставить на сервере строковую шаблонизацию?

```jsx
<ul className={cls}>
  {elements.map(item => (
    <li>
      <Item item={item} />
    </li>
  ))}
</ul>
```

```handlebars
<ul class="{{cls}}">
  {{#each elements}}
    <li>
      {{> Item item}}
    </li>
  {{/each}}
</ul>
```

```jsx
<ul className={cls}>
  {elements.map(item => (
    <li>
      <Item item={item} />
    </li>
  ))}
</ul>
```

```hbs
<ul class="{{cls}}">
  {{#each elements}}
    <li>
      {{> Item item}}
    </li>
  {{/each}}
</ul>
```

```jsx
<ul className={cls}>
  {elements.map(item => (
    <li>
      <Item item={item} />
    </li>
  ))}
</ul>
```

```hbs
<ul class="{{cls}}">
  {{#each elements}}
    <li>
      {{> Item item}}
    </li>
  {{/each}}
</ul>
```

```jsx
<ul className={cls}>
  {elements.map(item => (
    <li>
      <Item item={item} />
    </li>
  ))}
</ul>
```

```hbs
<ul class="{{cls}}">
  {{#each elements}}
    <li>
      {{> Item item}}
    </li>
  {{/each}}
</ul>
```

Почти то же самое, но...

Нельзя вот так

```
<div{{#if cls}} class="{{cls}}">{{else}}>{{/if}}
  Some block content
</div>
```

А вот так уже можно

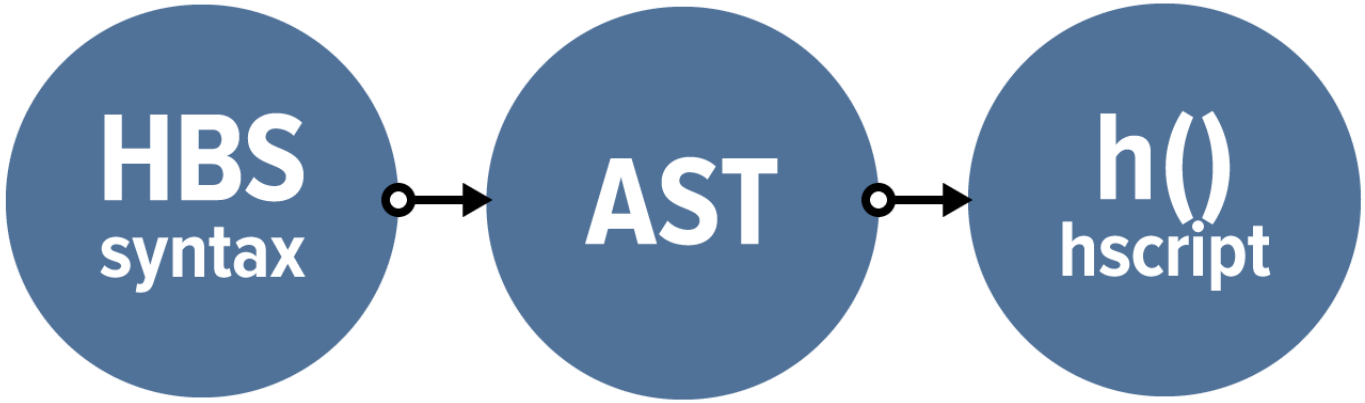
```
<div {{#if cls}}class="{{cls}}"{{/if}}>
  Some block content
</div>
```

Нельзя произвольный JS

```
<div>
  Sum: {arr.reduce((s, a) => (s + a), 0)}
</div>
```

Нужно готовить данные для всего приложения перед рендером

Реализация


HBS
syntax
AST
h()
hscript

До:

```
<ul class="{{cls}}">
  {{#each elements}}
    <li>
      {{> Item item}}
    </li>
  {{/each}}
</ul>
```

После:

```
h('ul', { className: cls },
  elements.map(ctx => (
    h('li', null,
      h(Item, ctx.item)
    )
  ))
)
```

Что парсит Handlebars

```
<ul class="{{cls}}">
  {{#each elements}}
    <li>
      {{item}}
    </li>
  {{/each}}
</ul>
```

Нам нужно парсить всё

```
{
  tag: "h1",
  attributes: [],
  children: [
    { type: "text",       value: "Hello, " },
    { type: "handlebars", value: "name"    }
  ]
}
```

Написание парсера


ИНТЕРЕСНО

PEG.js

Генерирует парсеры по описанной грамматике.

handlebars-ex

Парсит handlebars-шаблон в AST с учетом HTML-тегов, атрибутов и пр.

blueHTML

- Парсит handlebars-шаблоны и умеет конвертировать полученное AST в hscript для библиотеки Virtual DOM
- Имеет адаптер для React

BlueHTML предназначен
для переиспользования
legacy-шаблонов на Handlebars

Необязательно React

- Preact
- Virtual DOM
- Inferno

Webpack-лоадер

```
01. class Foo extends React.Component {
02.   render() {
03.     return require('./foo.html')(this.props);
04.   }
05. }
```

Переиспользование DOM

Если использовать текущую схему, приложение на клиенте будет полностью перерисовываться

Переиспользование DOM в React

Компонент

```
01. class Foo extends React.Component {
02.   render()
03.     const name = 'World';
04.     return <h1>Hello, {name}</h1>;
05.   }
06. }
```

renderToString

```
<h1
  data-reactroot=""
  data-reactid="1"
  data-react-checksum="-380863587">
  <!-- react-text: 2 -->Hello, <!-- /react-text -->
  <!-- react-text: 3 -->World<!-- /react-text -->
</h1>
```

reactroot

```
<h1
  data-reactroot=""
  data-reactid="1"
  data-react-checksum="-380863587">
  <!-- react-text: 2 -->Hello, <!-- /react-text -->
  <!-- react-text: 3 -->World<!-- /react-text -->
</h1>
```

react-checksum

```
01. <h1
02.   data-reactroot=""
03.   data-reactid="1"
04.   data-react-checksum="-380863587">
05.   <!-- react-text: 2 -->Hello, <!-- /react-text -->
06.   <!-- react-text: 3 -->World<!-- /react-text -->
07. </h1>
```

react-checksum

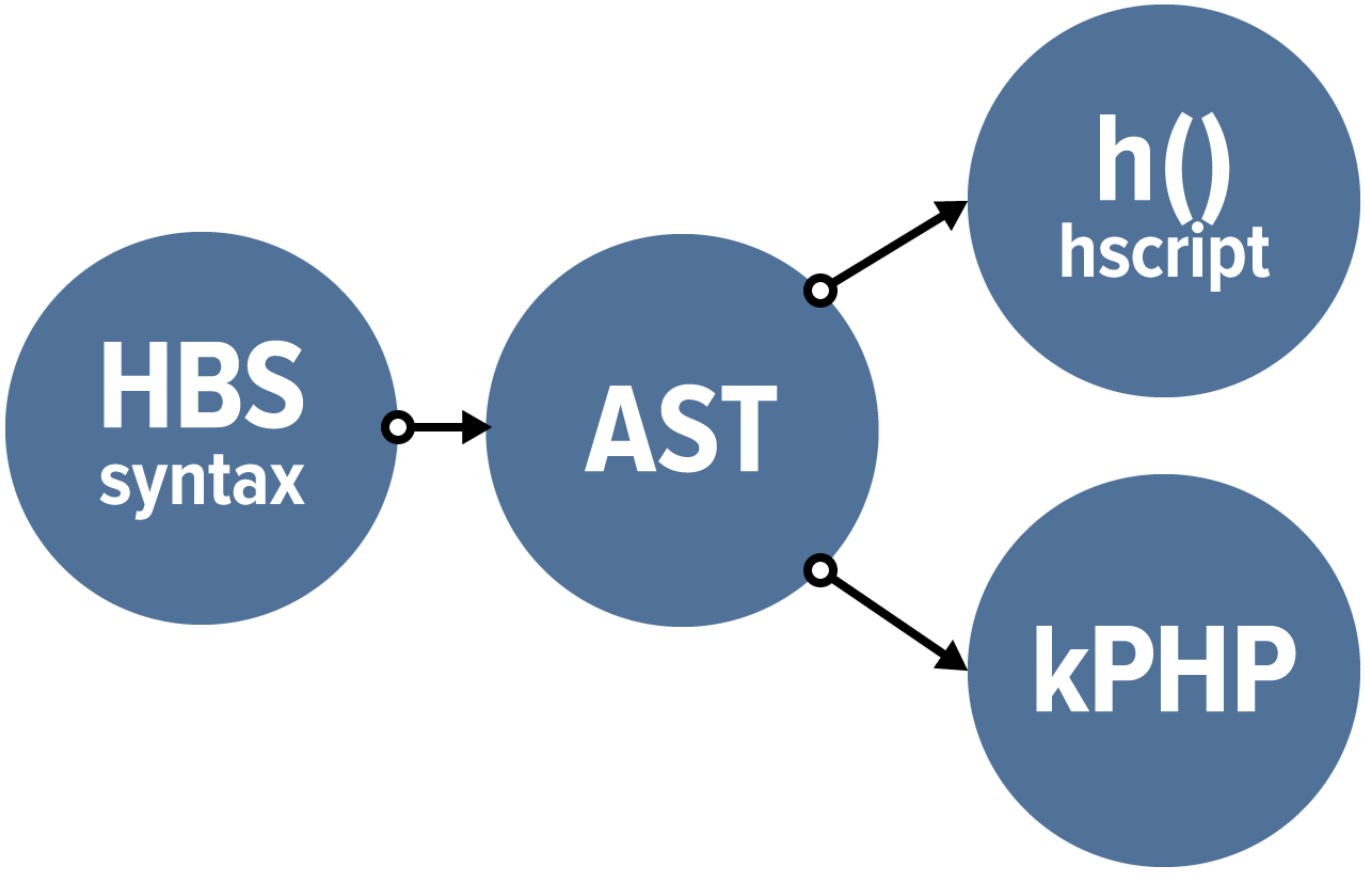
```
// ./src/renderers/dom/stack/server/ReactMarkupChecksum.js
var checksum = adler32(markup);
```

reactid

```
<h1
  data-reactroot=""
  data-reactid="1"
  data-react-checksum="-380863587">
  <!-- react-text: 2 -->Hello, <!-- /react-text -->
  <!-- react-text: 3 -->World<!-- /react-text -->
</h1>
```

reactid

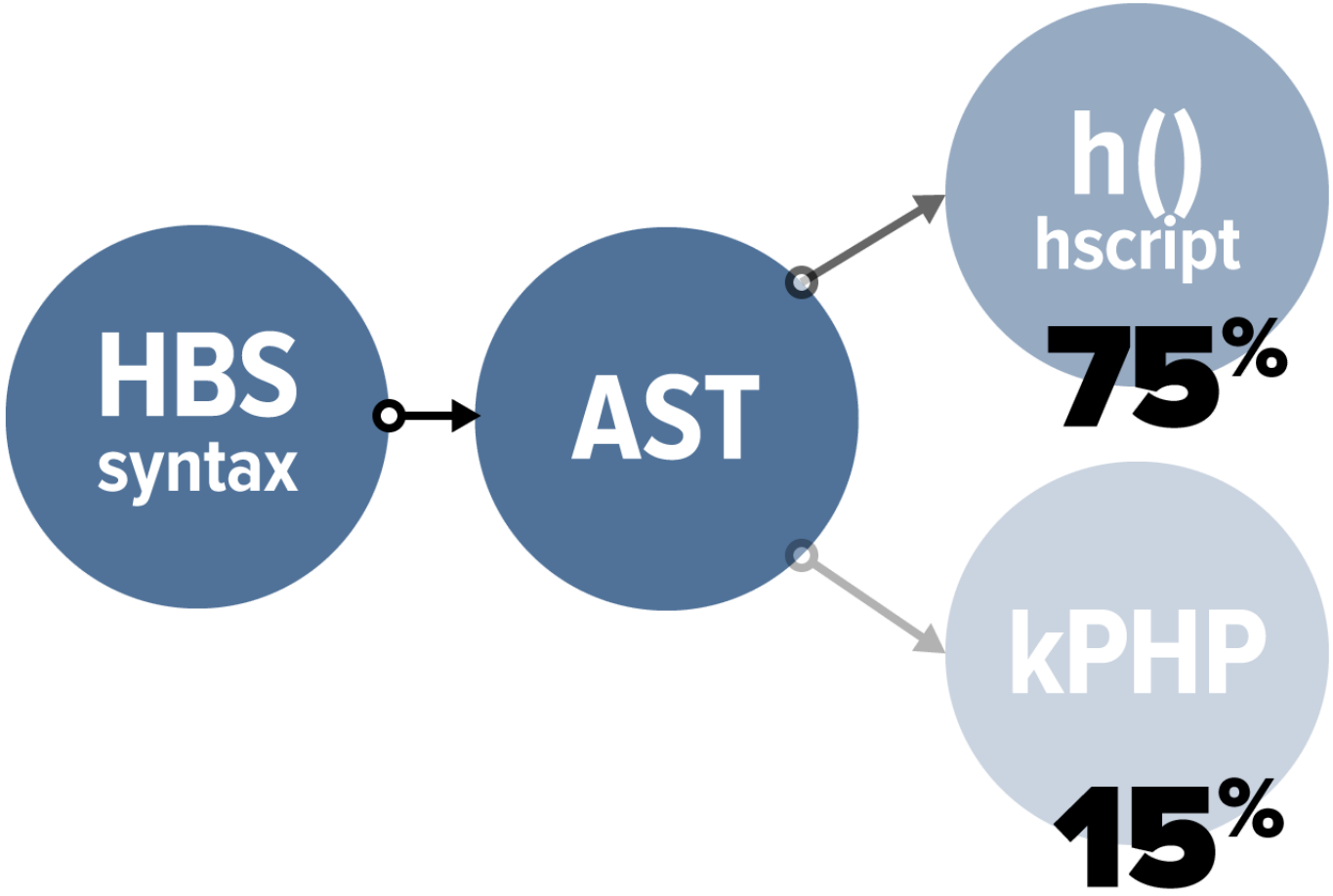
```
React.createElement('h1', {}, ['Hello, ', name]);
//                      1          2         3
```


HBS
syntax
AST
h()
hscript
kPHP

Итоги

Список из 10 000 элементов

React.renderToString()	1096 ms
React.renderToString() optimized	800 ms
Handlebars.php	880 ms
PHP precompiled	126 ms


HBS
syntax
AST
h()
hscript
75%
kPHP
15%

Проблемы

- Сгенерированные компоненты не всегда можно полноценно использовать на клиенте

Не всё поддерживается

- Partials
- With
- Lookup helper
- Log helper

BlueHTML TODO list

Все атрибуты приводятся к строке

```
01. obj['onClick'] = '';
02. obj['onClick'] += escape(ctx.onClick);
```

Проблемы

- Генерация Virtual DOM не всегда позволяет полноценно использовать компоненты на клиенте
- Зависимость от версии React

Обновление React заставит заново попадать в каждый пробел

Планы

Планы

- Расширить поддерживаемый синтаксис исходных шаблонов
- Реализовать генерацию kPHP-шаблонов
- Улучшить генерацию клиентских компонентов
- Заменить React на аналог?

Зачем это может пригодиться вам

- Возможность переиспользовать legacy-шаблоны на клиенте с Virtual DOM или React
- Универсальные шаблоны, которые можно рендерить на сервере строковым шаблонизатором

Зачем это может пригодиться вам

- Возможность переиспользовать legacy-шаблоны на клиенте с Virtual DOM или React
- Универсальные шаблоны, которые можно рендерить на сервере строковым шаблонизатором **— только для отчаянных**

Зачем это нам

- Декларативный рендеринг, возможность использовать библиотеки с Virtual DOM на клиенте, используя серверные шаблоны.
- Сохраняем возможность использовать строковую шаблонизацию в разделах, которые не требуют сложного фронтенда
-


ВУАЛЯ

vk.cc/6lnawe

Тимофей Чаптыков

tim.chaptykov@gmail.com

vk.com/tim.chaptykov

@chaptykov