Web Standards Days

Как построить DOM

Роман Дворнов
Ostrovok.ru

Киев, Октябрь 2013

Немного о себе

- Работаю в Ostrovok.ru
- Ранее руководил веб-разработкой в ПС Единый кошелек
- Автор и мейтейнер фреймворка basis.js

Зачем нам DOM?

Веб-приложения

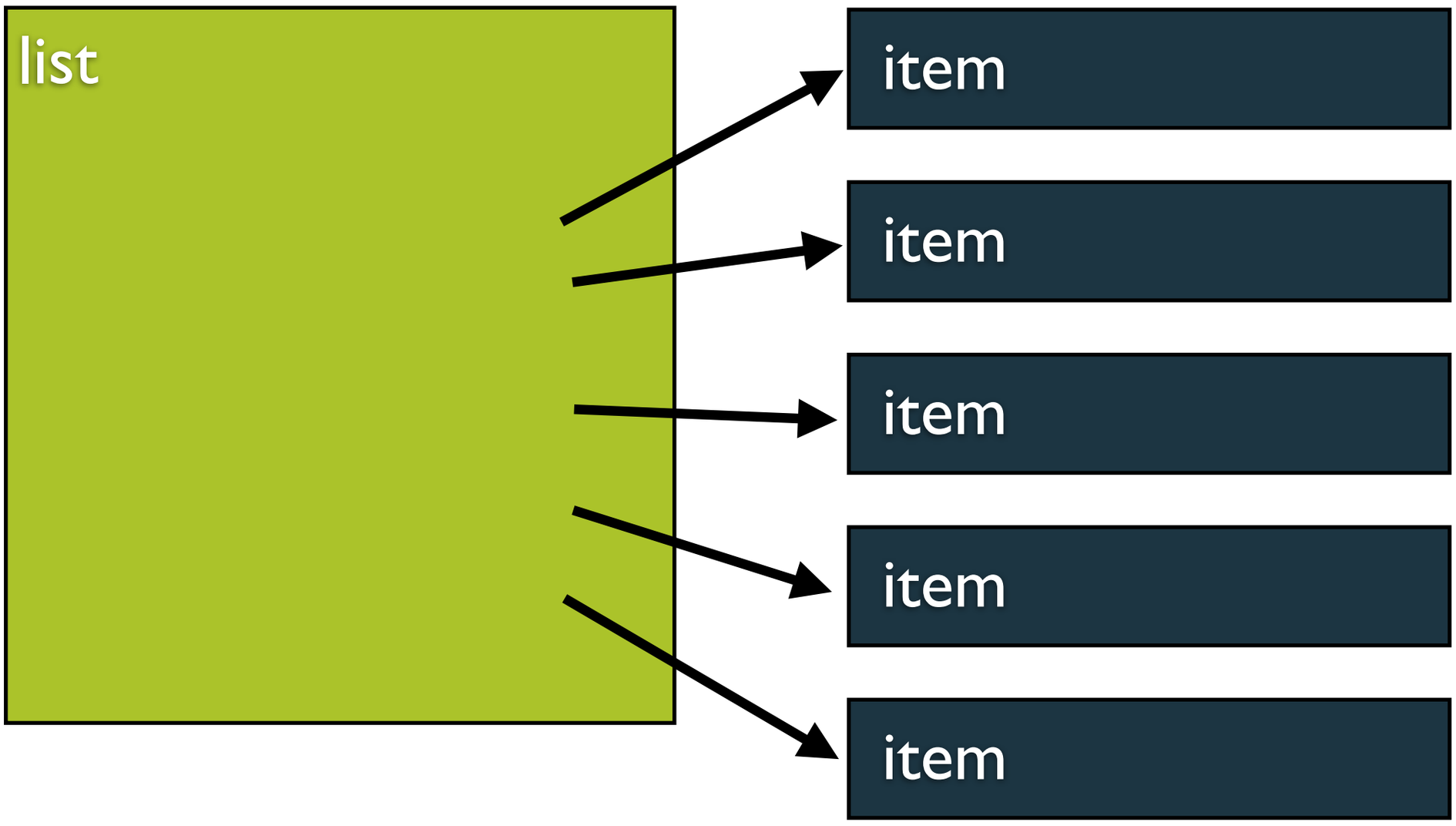
- Компонентный подход, переиспользование
- Разделение ответственности
- Много разных данных
- Ленивое построение
- Точечные обновления
- Динамика, динамика, динамика...

Компонентный подход

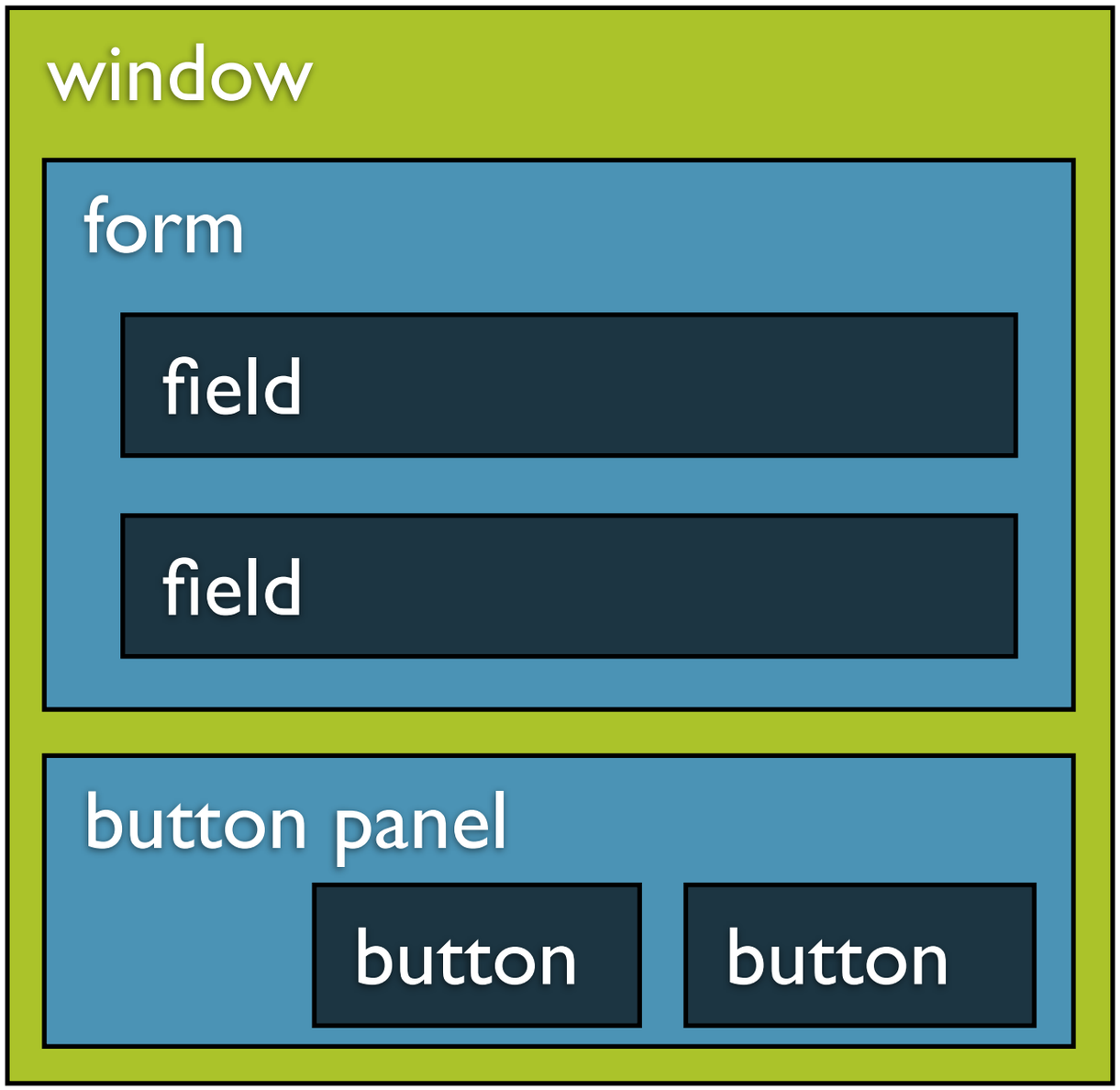
— декомпозиция

Обычный подход

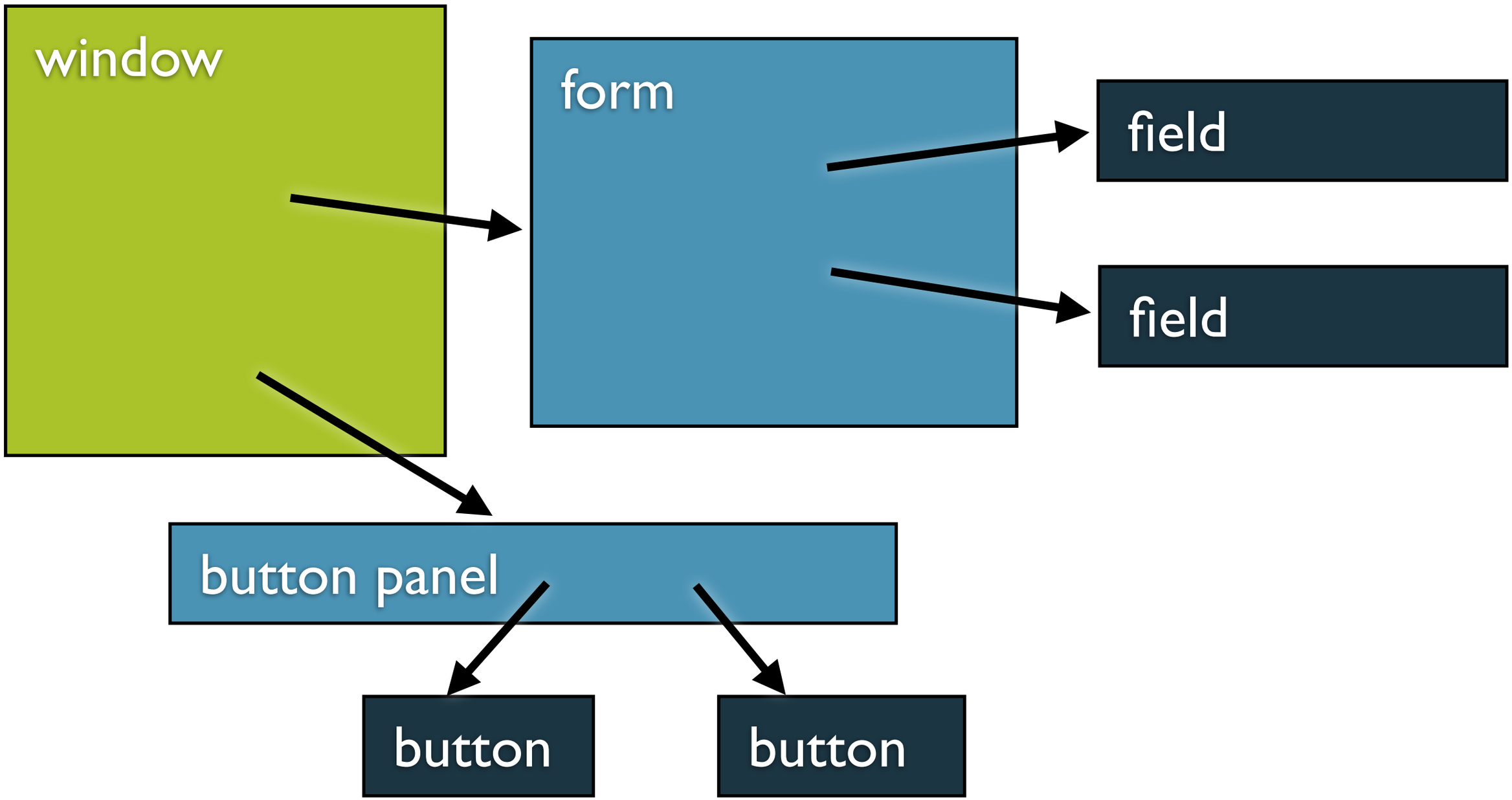
Компонентный подход



Обычный подход



Компонентный подход



Обычный подход

```
function renderList(data){
  var html = '';
  html = html + '<ul class="mylist' +
    (cls ? ' ' + cls : '') + '">';
  for (var i = 0; i < items.length; i++)
      html = html + '<li class="item ' +
          (items[i].selected ? ' selected' : '') + '>' +
          items[i].title +
        '</li>';
  html = html + '</ul>';
  return html;
};
...
element.innerHTML = renderList({ .. });
```

Обычный подход

```
function renderList(data){
  var html =
  html = html + '<ul class="mylist' +
    (cls ? ' ' + cls : '') + '">';
  for (var i = 0; i < items.length; i++)
    html = html + '<li class="item ' +
        (items[i].selected ? 'selected' : '') + '>' +
        items[i].title +
      '</li>';
  html = html + '</ul>';
  return html;
};
...
element.innerHTML = renderList({ .. });
```

DOM спасет ~~мир~~

динамический веб

Первый подход

или жизнь без шаблонов

Hardcore – DOM

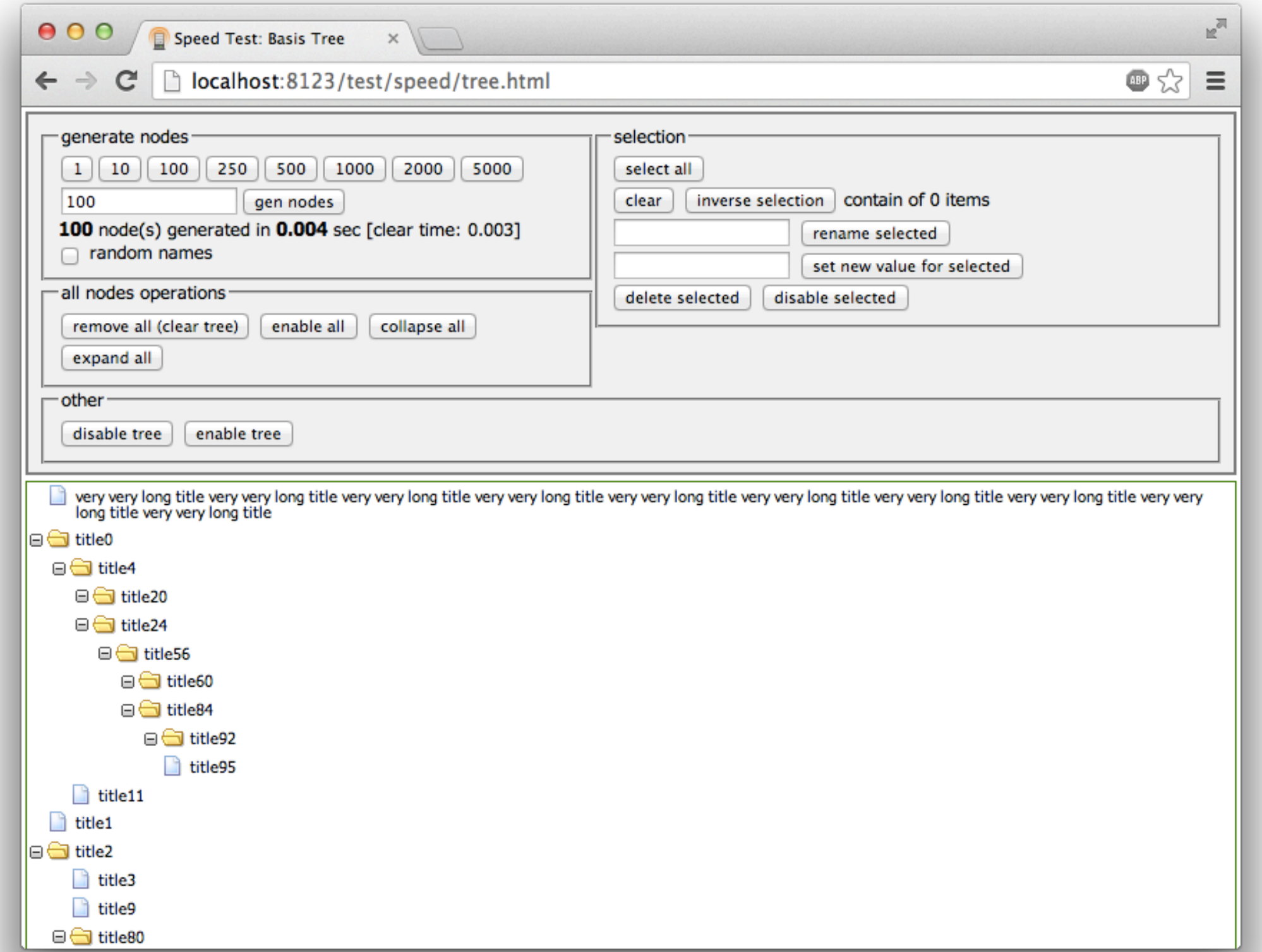
```
var Item = basis.ui.Node.subclass({
  init: function(config){
    this.element = basis.dom.create('li.item',  // создание DOM
      basis.dom.create('span.caption',
        this.title = basis.dom.createText(config.title)
      )
    );
  },
  setTitle: function(title){
    this.title.nodeValue = title; // точечное обновление
  },
  destroy: function(){
    ...
    this.element = null;          // обнуление ссылок
    this.title = null;
  }
});
```

Hardcore – события

```
var Item = basis.ui.Node.subclass({
  init: function(config){
    ...
    // подписка на события
    basis.dom.event.on(this.element, 'click', onclick, this);
  },
  ...
  onclick: function(event){  // подписанные методы получают event
    this.selected ? this.unselect() : this.select();
  },
  ...
  destroy: function(){
    ...
    // отписываемся
    basis.dom.event.off(this.element, 'click', onclick, this);
  }
});
```

Как быстро?

Speed tree – *«подопытный кролик»*



Результаты

	IE11	Firefox26	Chrome30	Opera12	iPhone4S
100 узлов, ms	45	15	8	10	50
2000 узлов, ms	1014	309	183	174	981

Доля работы с DOM

Speed tree – 2000 узлов

	IE11	Firefox26	Chrome30	Opera12	iPhone4S
общее время, ms	1014	309	183	174	981
создание DOM, ms	785	246	151	134	807
создание DOM, %	77,4	79,6	82,5	77,0	82,3

Более 75% тратится на создание DOM

Минусы подхода

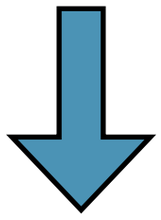
1. Медленое создание DOM фрагмента
2. Сложно менять "описание"
3. Нужно самостоятельно привязывать и удалять обработчики событий
4. Много кода по обновлению DOM
5. Не наглядно
6. Легко получить утечки памяти

Решаем проблему

Медленное создание DOM

Отказ от удобств

```
this.element = basis.dom.create('li.item');
```



```
this.element = document.createElement('li');
this.element.className = 'item';
```

Отказ от удобств

Speed tree – 2000 узлов

	IE11	Firefox26	Chrome30	Opera12	iPhone4S
выигрыш, ms	463	171	110	92	540
выигрыш, %	59,0 %	69,5 %	72,8 %	68,7 %	66,9 %

Привязка событий

- Добавляем обработчики на элемент корневого компонента
- Жертвуем возможностью определять слушаемые события в коде дочерних компонент
- Один обработчик для каждого типа события вместо N

Привязка событий

Speed tree – 2000 узлов

	IE11	Firefox26	Chrome30	Opera12	iPhone4S
выигрыш, ms	115	37	23	13	118
выигрыш, %	14,6 %	15,0 %	15,2 %	9,7 %	14,6 %

cloneNode(true)

```
var Leaf = basis.ui.Node.subclass({
  proto: basis.dom.create('li.item',  // эталон
    basis.dom.create('span.caption',
      basis.dom.createText('no title')
    )
  ),
  init: function(cfg){
    ...
    // получаем клон
    this.element = this.proto.cloneNode(true);

    this.title = this.element.firstChild.firstChild;
    this.title.nodeValue = cfg.title || 'no title';
  }
});
```

Минимизируем операции с DOM

- Любая операция с DOM чего-то стоит
- Даже присвоение того же значения, что уже есть у свойства, часто равносильно присвоению нового значения
- Даже операции чтения, то есть обращение к свойствам firstChild, lastChild, nextSibling и т.д.

Меньше операций с DOM

Speed tree – 2000 узлов

	IE11	Firefox26	Chrome30	Opera12	iPhone4S
выигрыш, ms	49	18	5	15	9
выигрыш, %	6,2 %	7,3 %	3,3 %	11,2 %	1,1 %

После всех оптимизаций

Speed tree – 2000 узлов

	IE11	Firefox26	Chrome30	Opera12	iPhone4S
было	785	246	151	134	807
стало	158	20	13	14	140
выигрыш, ms	627	226	138	120	667
выигрыш, %	79,9 %	91,9 %	91,4 %	89,6 %	82,7 %

Минус один

1. ~~Медленое создание DOM фрагмента~~
2. Сложно менять "описание"
3. Нужно самостоятельно привязывать и удалять обработчики событий
4. Много кода по обновлению DOM
5. Не наглядно
6. Легко получить утечки памяти

Решаем проблему

Сложность описания

И тут появляются шаблоны

basis.html.Template

```
var Leaf = basis.ui.Node.subclass({
  template: new basis.html.Template(
    '<li{element} class="item">{title}</li>'
  ),
  init: function(){
    ...
    // этот вызов создаст DOM фрагмент и вернет карту
    this.tmpl = this.template.createInstance();

    this.tmpl.title.nodeValue = this.title;
  }
});
```

template.createInstance(..)

Создает DOM фрагмент
и возвращает карту ссылок

```
<div{element}>
  <h2>{title}</h2>
  <ul{content}/>
</div>
```

```
{
  element: <div>,
  title: #text,
  content: <ul>
}
```

Парсинг описания

- Без использования innerHTML
- Комбинация RegExp и конечного автомата
- Позволяет использовать собственные конструкции и теги
- Делается только раз

Минус два

1. ~~Медленое создание DOM фрагмента~~
2. ~~Сложно менять "описание"~~
3. Нужно самостоятельно привязывать и удалять обработчики событий
4. Много кода по обновлению DOM
5. Не наглядно
6. Легко получить утечки памяти

Решаем проблему

Привязка событий

В шаблоне

```
<li{element} class="item" event-click="select">
  {title}
</li>
```

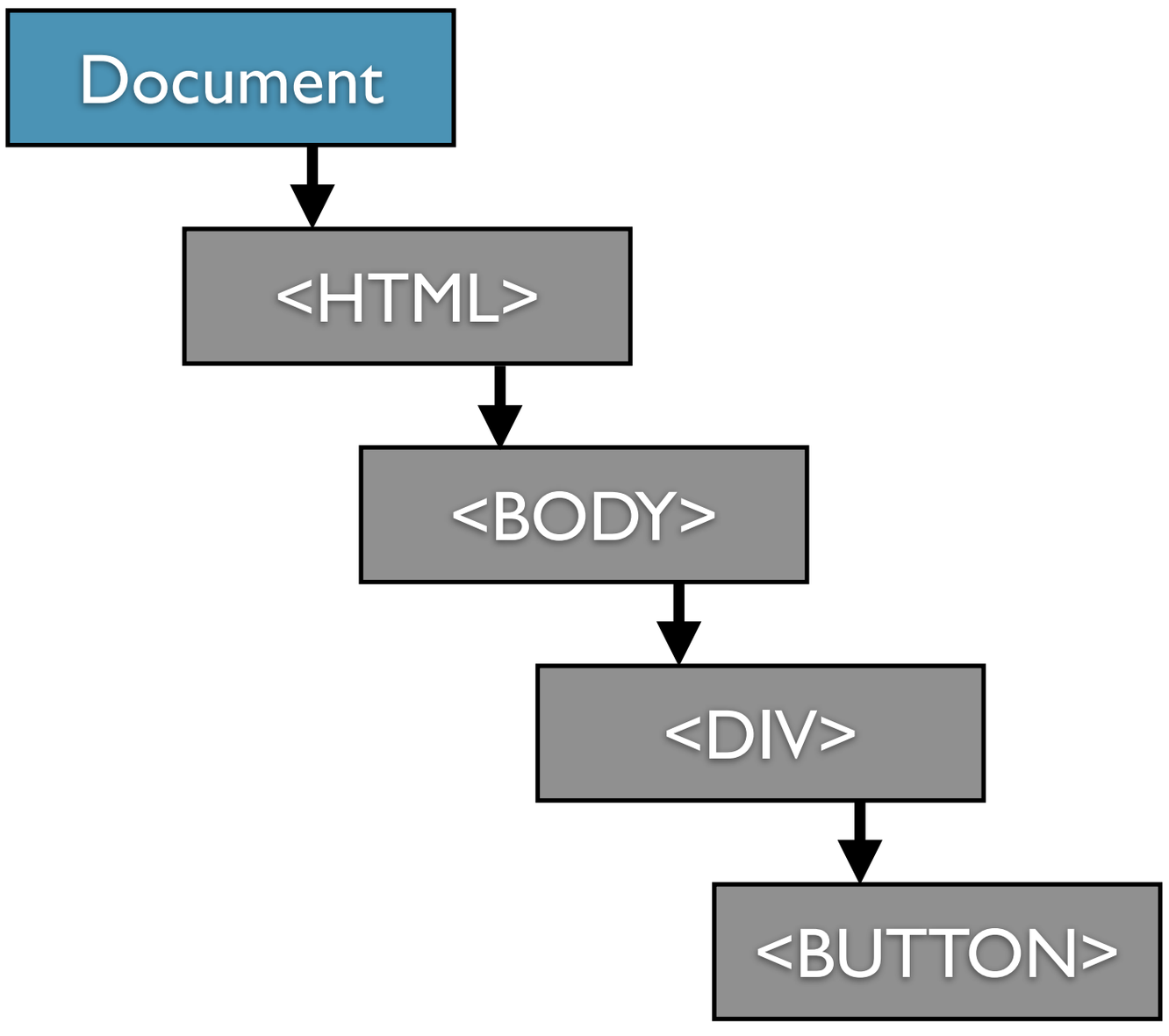
В коде

```
var Leaf = basis.ui.Node.subclass({
  ...
  action: {
    select: function(event){
      this.select();
    },
    ...
  }
});
```

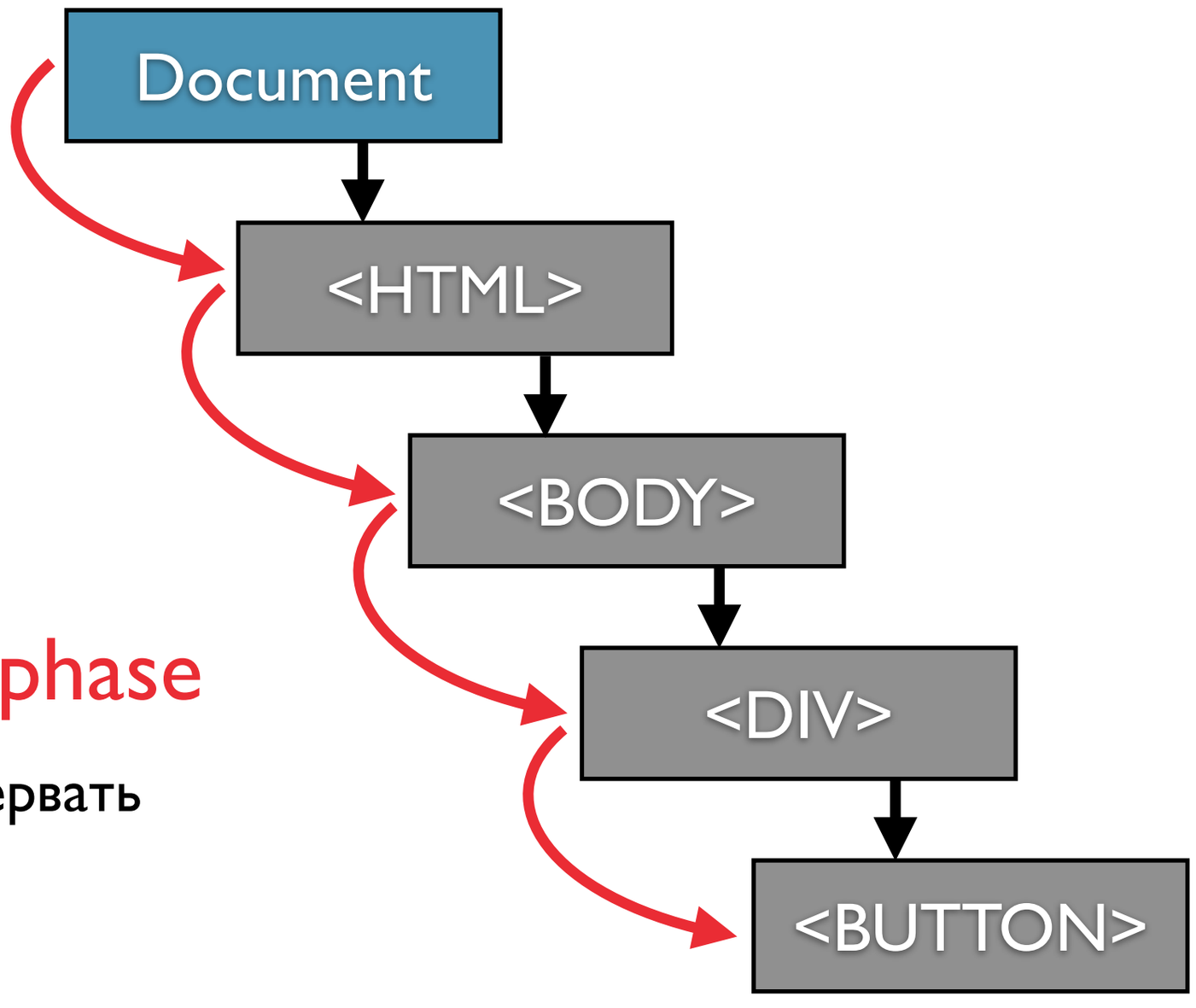
action – дополняется при наследовании

Как это работает

Событийная модель



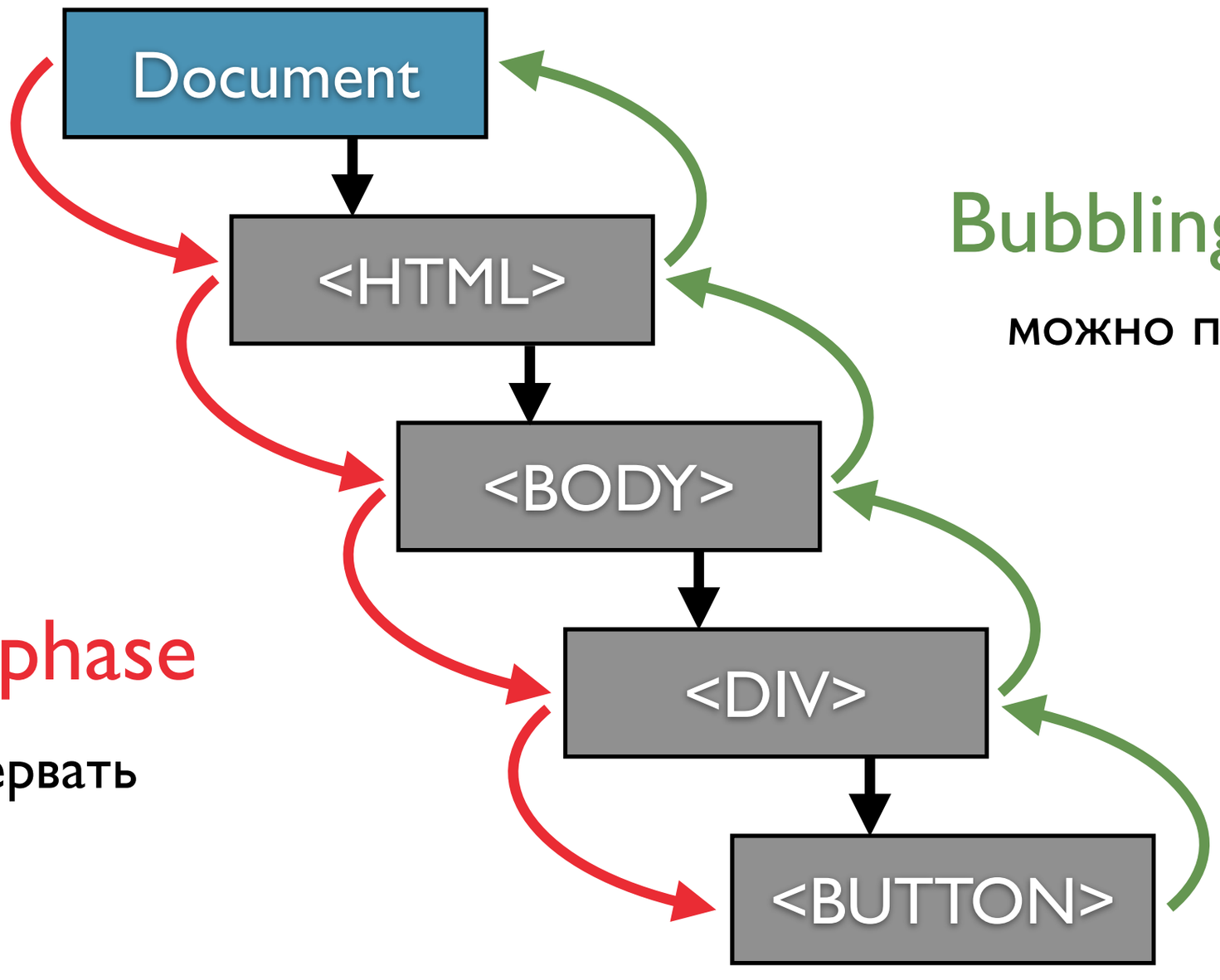
Событийная модель



Capture phase

нельзя прервать

Событийная модель



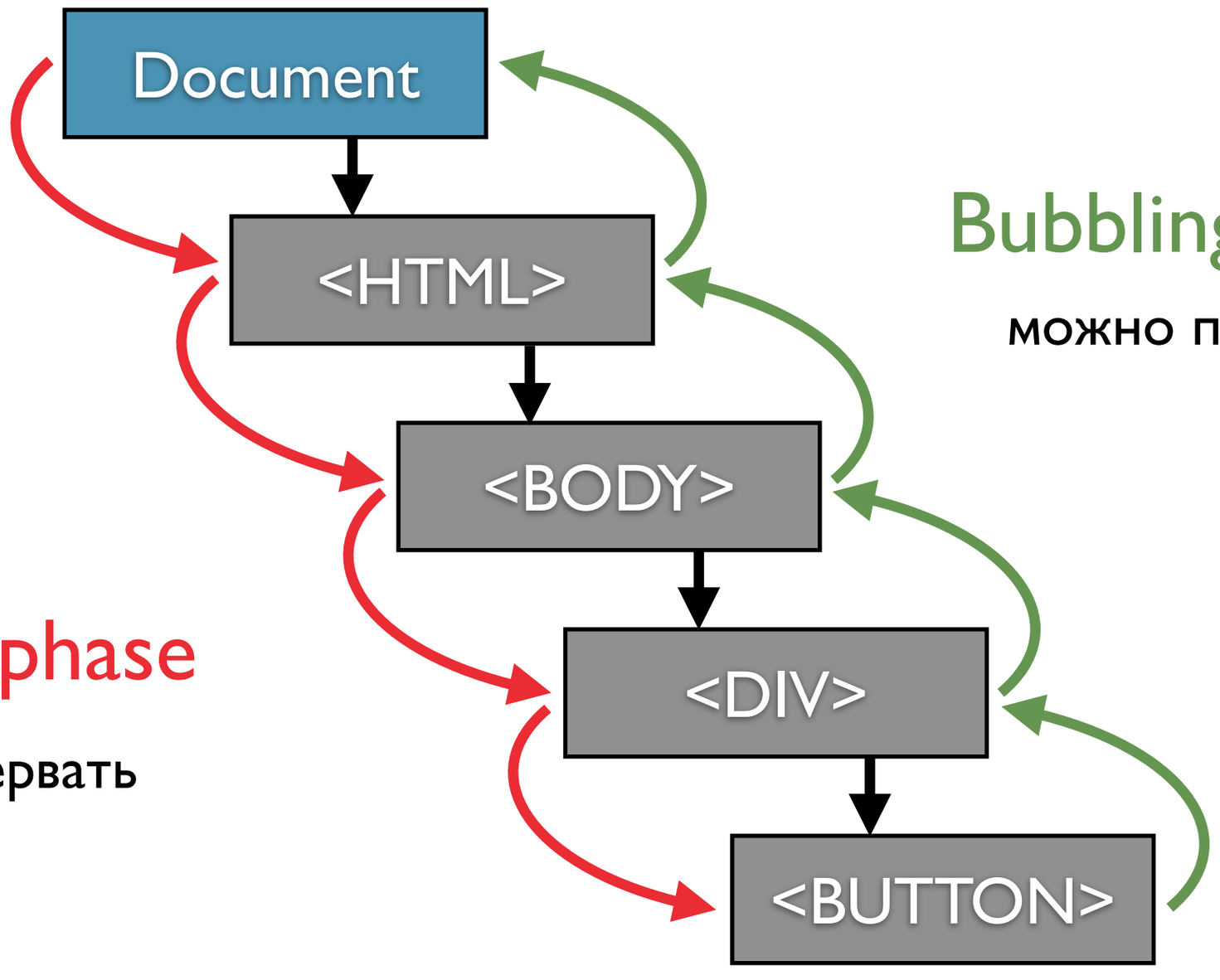
Bubbling phase

можно прервать

Capture phase

нельзя прервать

Добавляем обработчик



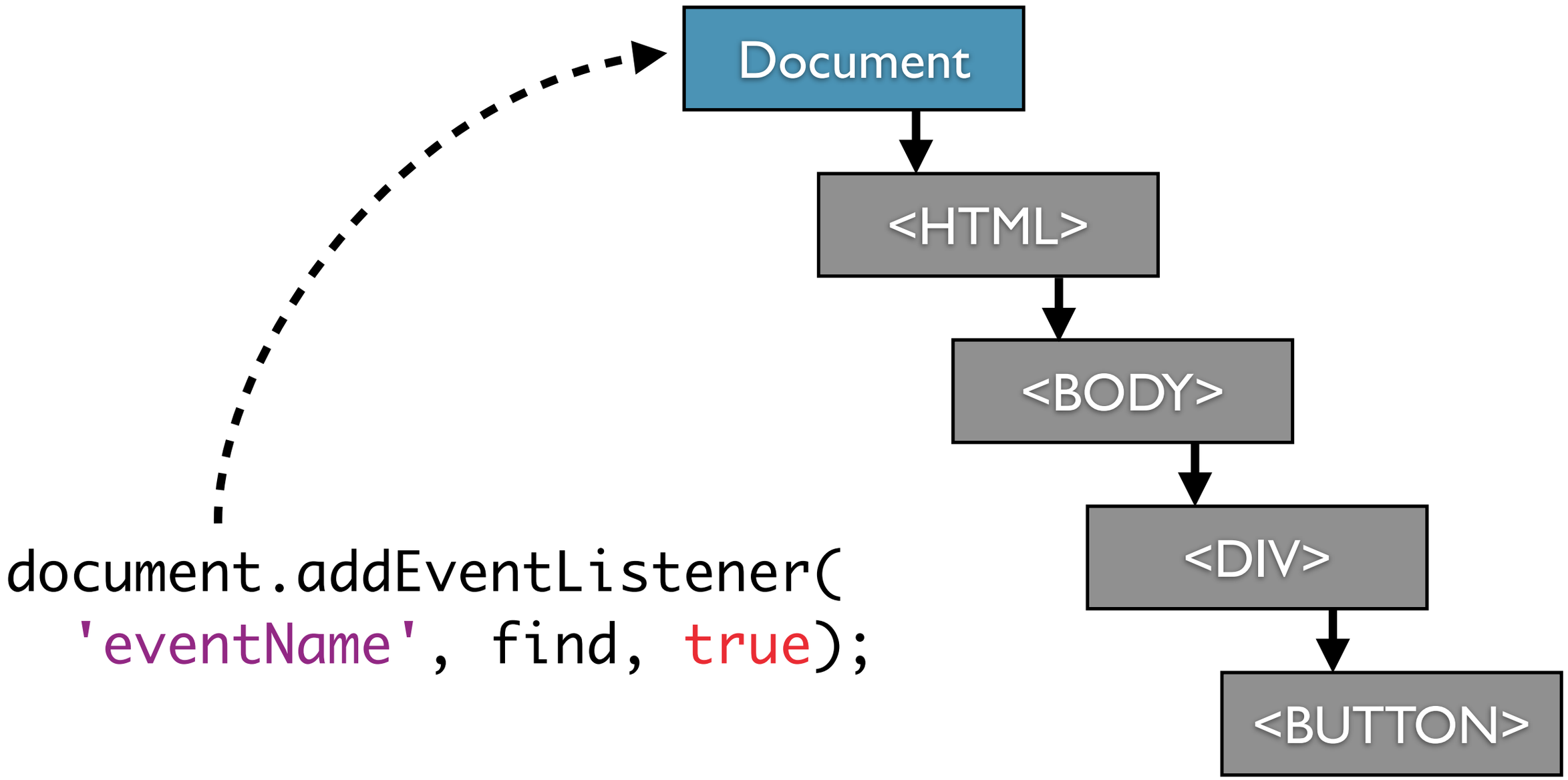
Bubbling phase

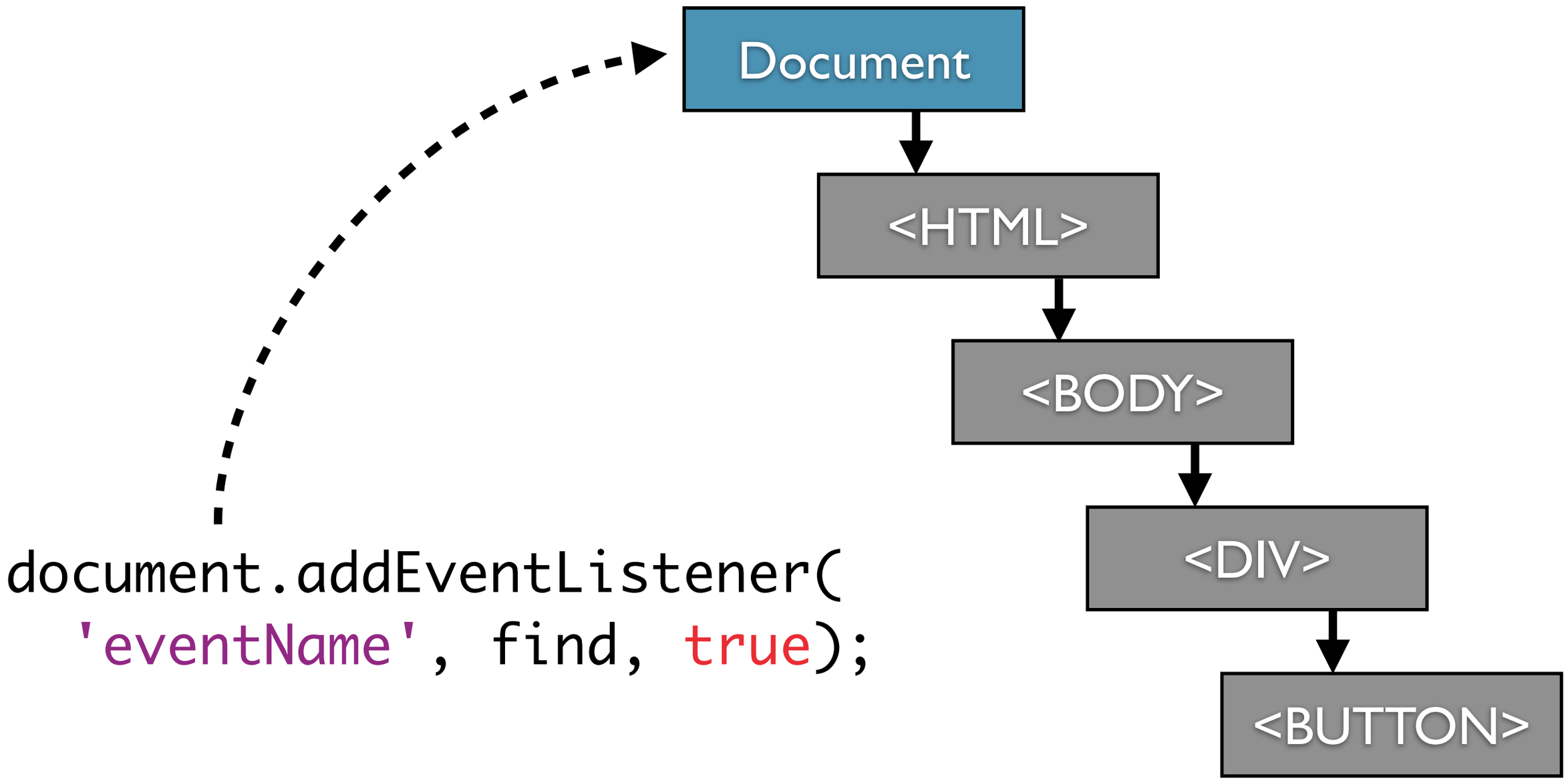
можно прервать

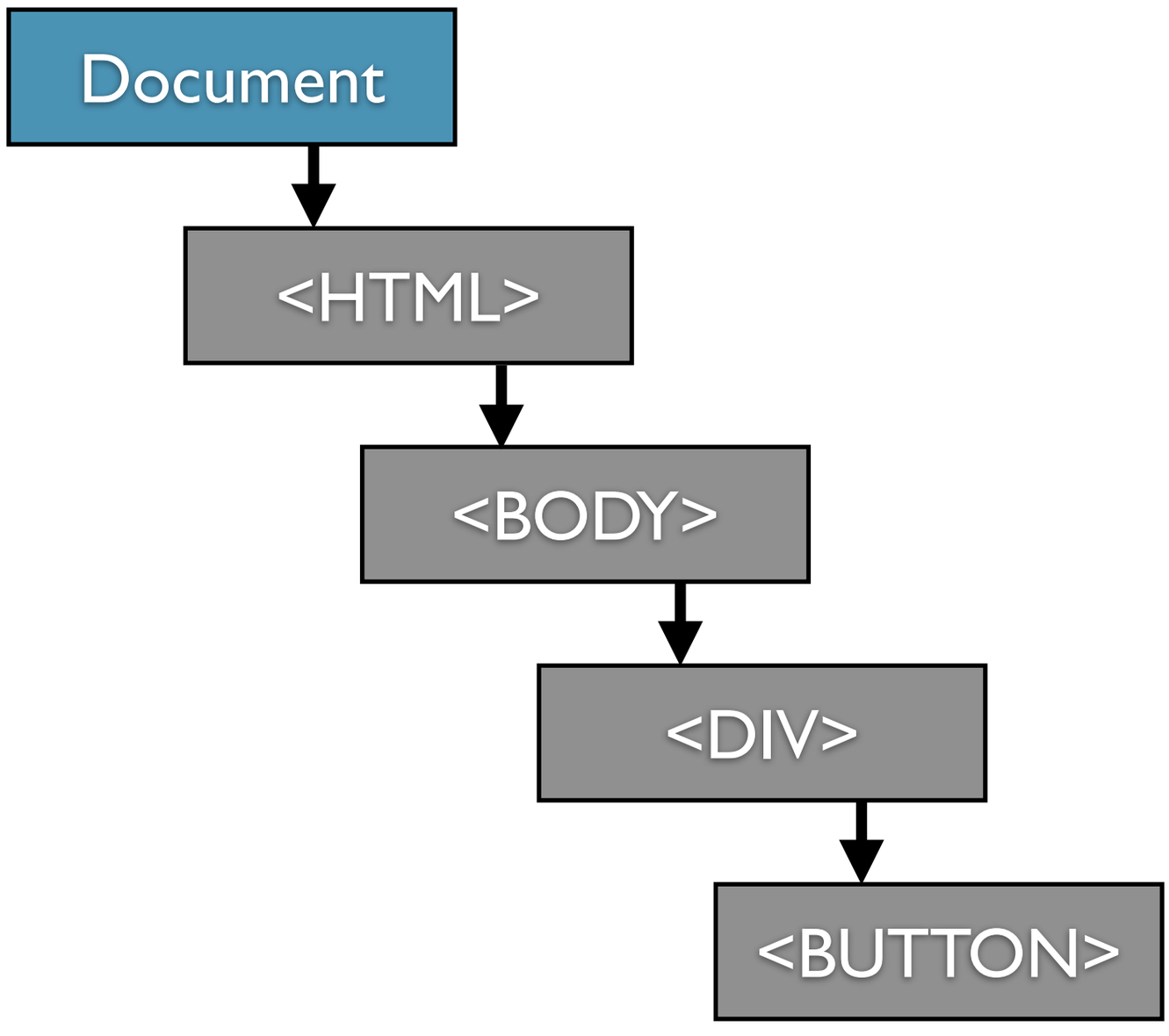
Capture phase

нельзя прервать

Добавляем обработчик

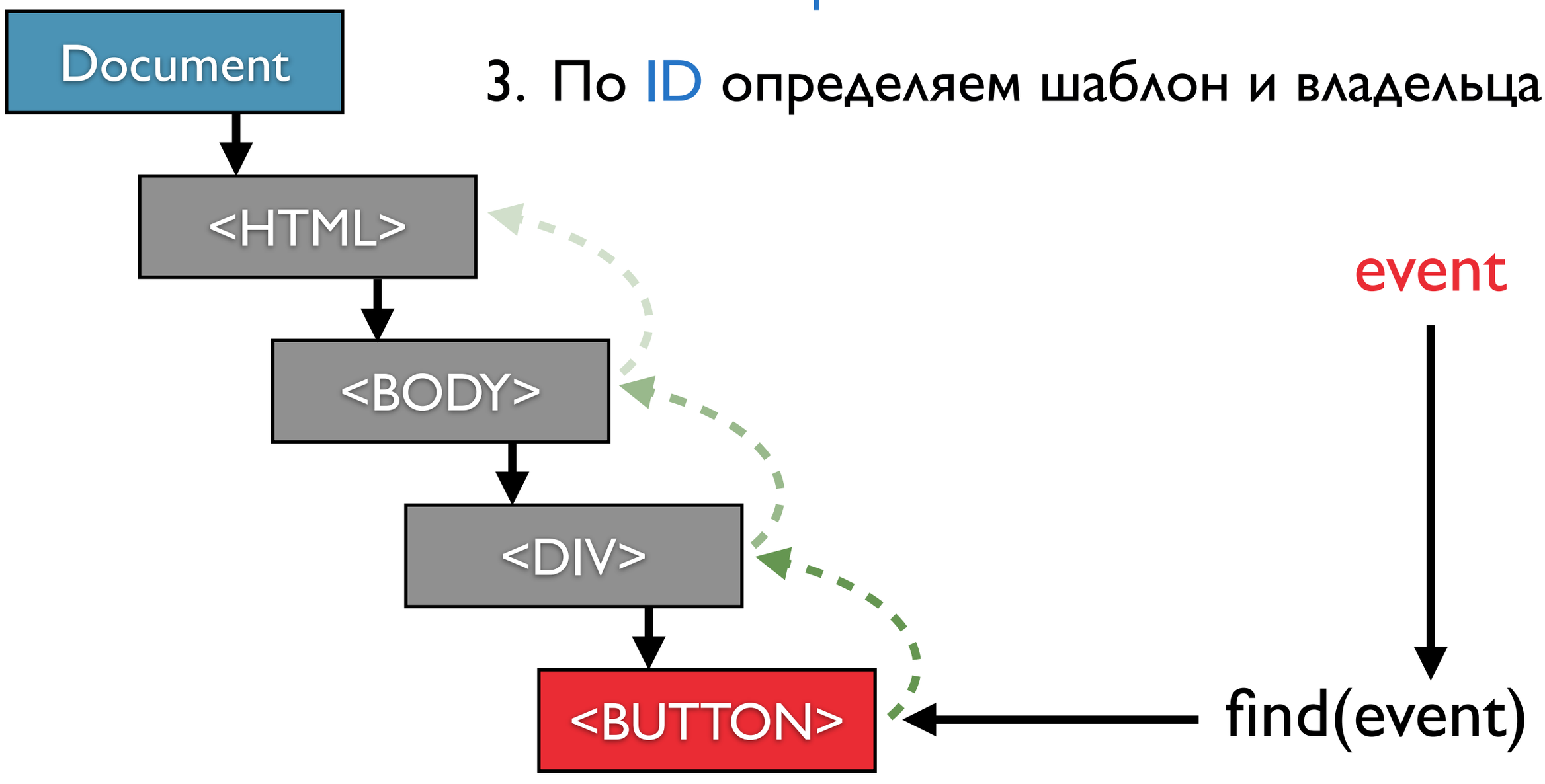



Document
<HTML>
<BODY>
<DIV>
<BUTTON>
document.addEventListener(
'eventName', find, true);



find(event)

1. Ищем ближайший от target элемент
   с атрибутом event-{eventName}
2. От него ближайший элемент
   с basisTemplateId
3. По ID определяем шаблон и владельца



IE8 и старые браузеры

Описание шаблона

```
<div event-click="action">
  {title}
</div>
```

Шаблонизатор добавляет атрибут в эталонный DOM фрагмент,
но только для браузеров без addEventListener

```
<div event-click="action"
     onclick="basisTemplateHandler('click', event)">
  {title}
</div>

// basisTemplateHandler вызывает find
```

Привязка событий

- Один глобальный обработчик на каждое уникальное событие
- Привязки и отвязки обработчиков как таковых нет
- Можно добавить одно и тоже действие на несколько узлов
- Изменение шаблона ничего не ломает
- Нет места для утечек памяти

Минус три

1. ~~Медленое создание DOM фрагмента~~
2. ~~Сложно менять "описание"~~
3. ~~Нужно самостоятельно привязывать и удалять обработчики событий~~
4. Много кода по обновлению DOM
5. Не наглядно
6. Легко получить утечки памяти

Решаем проблему

Обновление DOM

В шаблоне

```
<li class="item {selected} {disabled}">
  <span title="title is {title}">
    {title} ({count})
  </span>
</li>
```

В коде

```
var Leaf = basis.ui.Node.subclass({
  title: 'no title',
  binding: {
    title: function(leaf){
      return leaf.title;
    }
  },
  setTitle: function(newTitle){
    this.title = newTitle;
    this.updateBind('title'); // когда надо обновить
  }
});
```

binding – дополняется при наследовании

Если есть события

```
var Leaf = basis.ui.Node.subclass({
  binding: {
    disabled: {
      events: ['disable', 'enable'],
      getter: function(leaf){
        return leaf.isDisabled();
      }
    }
  }
});
```

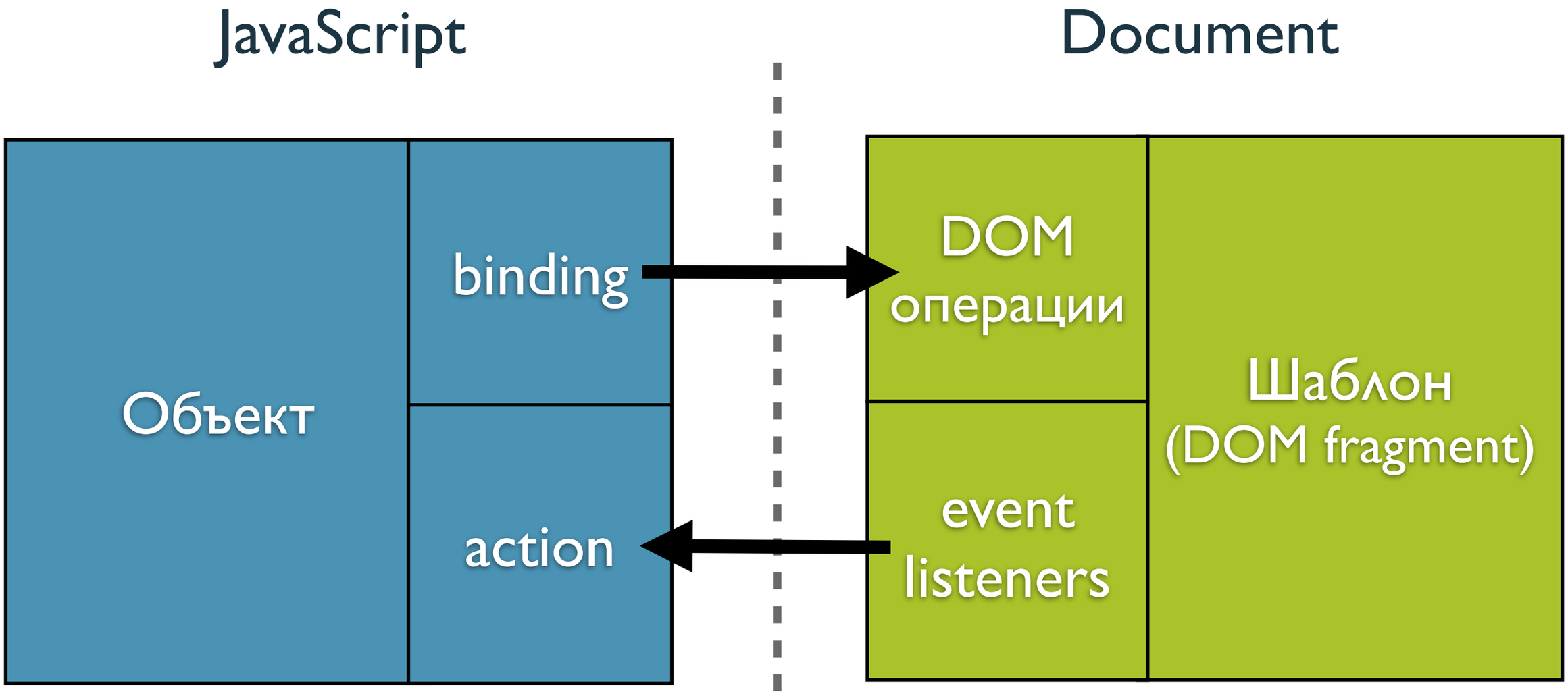
Проблемы решены

1. ~~Медленое создание DOM фрагмента~~
2. ~~Сложно менять "описание"~~
3. ~~Нужно самостоятельно привязывать и удалять обработчики событий~~
4. ~~Много кода по обновлению DOM~~
5. ~~Не наглядно~~
6. ~~Легко получить утечки памяти~~

Компоненты независимы от описания шаблона

большая часть операций с DOM автоматизирована и осуществляется шаблоном

Разделение логики и представления



DOM vs. Template

Speed tree 2000 узлов

	IE11	Firefox26	Chrome30	Opera12	iPhone4S
DOM	785	246	151	134	807
DOM + оптимизации	158	20	13	14	140
basis.js templates	230	65	55	58	337

DOM vs. innerHTML

Причем тут innerHTML?

- Является простым и наиболее популярным трансформатором HTML → DOM
- Его использует метод jQuery.html

innerHTML baseline

```
this.element = document.createElement('li');
this.element.className = 'tree-node';
this.element.innerHTML =
'<div class="title">' +
  '<span class="caption">' +
    this.data.title +
  '</span>' +
'</div>';
```

DOM vs. innerHTML

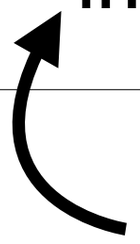
Speed tree – 2000 узлов

	IE11	Firefox24	Chrome30	Opera12	iPhone4S
DOM + оптимизации	158	20	13	14	140
basis.js templates	230	65	55	58	337
innerHTML	217	42	44	37	272

DOM vs. innerHTML

Speed tree – 2000 узлов

	IE11	Firefox24	Chrome30	Opera12	iPhone4S
DOM + оптимизации	158	20	13	14	140
basis.js templates	230	65	55	58	337
innerHTML	217	42	44	37	272



Но здесь нет пост-процессинга

DOM vs. innerHTML

Speed tree – 2000 узлов

	IE11	Firefox24	Chrome30	Opera12	iPhone4S
DOM + оптимизации	158	20	13	14	140
basis.js templates	230	65	55	58	337
innerHTML	217	42	44	37	272

А тут есть

Пост-процессинг

- Добавление классов и другие изменения DOM фрагмента
- Навешивание обработчиков событий
- Получение ссылок на элементы

Пример из TodoMVC на backbone.js

```
app.TodoView = Backbone.View.extend({
  ...
  render: function () {
    this.$el.html(this.template(this.model.toJSON()));
    this.$el.toggleClass('completed',
      this.model.get('completed'));
    this.toggleVisible();
    this.$input = this.$('.edit');
    return this;
  },
  ...
});
```

tinyurl.com/nkpcyd6

Делаем так же

```
this.$el = $('<li>').attr({ class: 'tree-node' });
this.element = this.$el[0];
this.$el.html('...html...');

this.childNodesElement = $el.find('.content')[0];

this.$el.toggleClass('selected', this.selected);
this.$el.toggleClass('disabled', this.disabled);

this.$el.on('click', '.caption', this.select.bind(this));
```

DOM vs. innerHTML

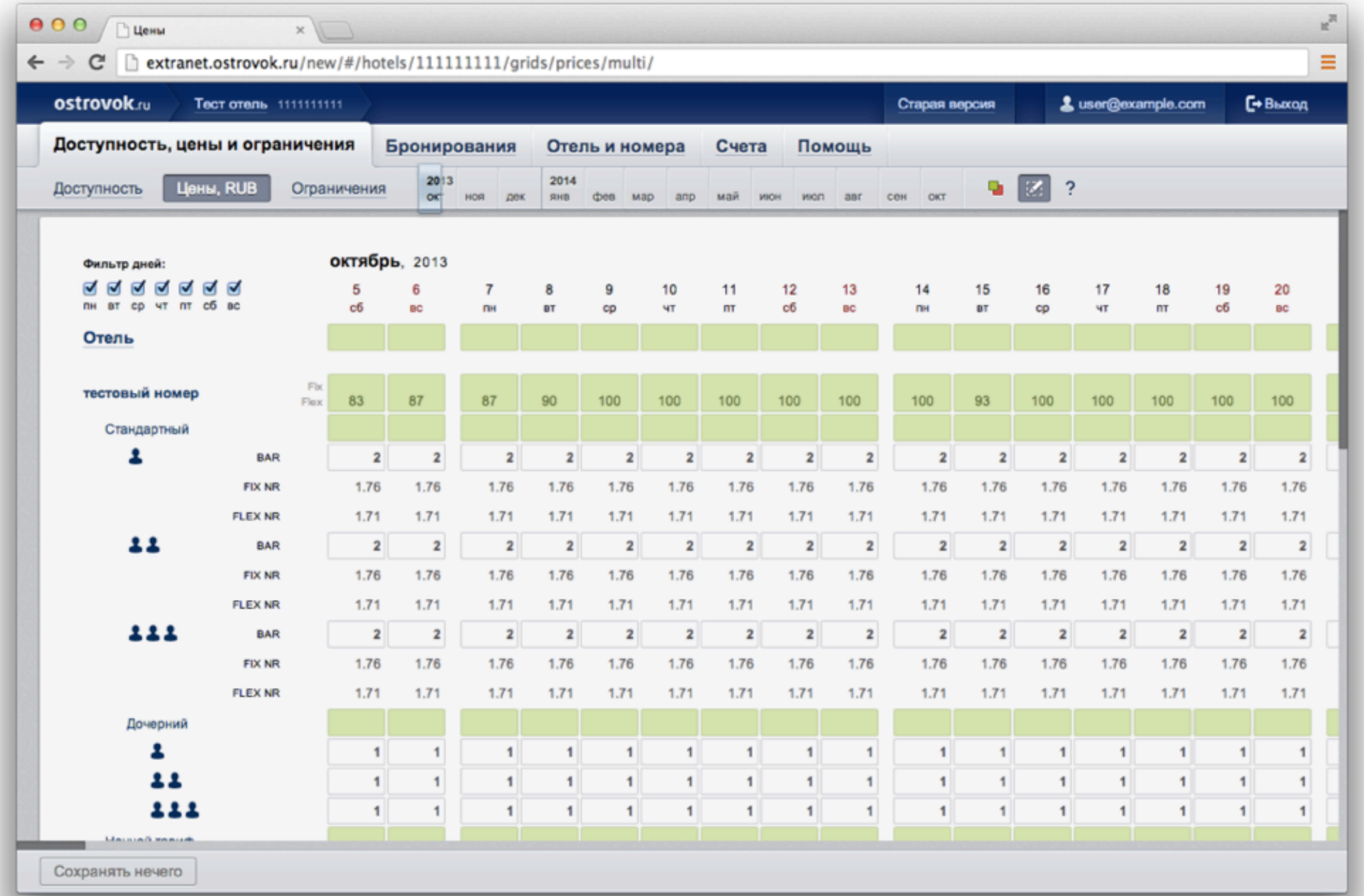
Speed tree – 2000 узлов

	IE11	Firefox24	Chrome30	Opera12	iPhone4S
DOM + оптимизации	158	20	13	14	140
basis.js templates	230	65	55	58	337
innerHTML	217	42	44	37	272
backbone style	499	296	306	208	1138

— *У вас никогда не будет 2000 узлов?*

«640КБ должно быть достаточно для каждого»

Билл Гейтс, 1981-й год


Цены
extranet.ostrovok.ru/new/#/hotels/111111111/grids/prices/multi/
ostrovok.ru
Тест отель 1111111111
Старая версия
user@example.com
Выход
Доступность, цены и ограничения
Бронирования
Отель и номера
Счета
Помощь
Доступность
Цены, RUB
Ограничения
2013 окт ноя дек
2014 янв фев мар апр май июн июл авг сен окт
Фильтр дней:
пн вт ср чт пт сб вс
октябрь, 2013
5 сб 6 вс 7 пн 8 вт 9 ср 10 чт 11 пт 12 сб 13 вс 14 пн 15 вт 16 ср 17 чт 18 пт 19 сб 20 вс
Отель
тестовый номер
Fix Flex
83 87 87 90 100 100 100 100 100 100 93 100 100 100 100 100
Стандартный
BAR 2
FIX NR 1.76
FLEX NR 1.71
Дочерний
1
Сохранять нечего

Цифры не включают reflow и repaint

Не включают время создания компонент

В бою

- Более сложные шаблоны, композиция
- Дополнительная логика
- Обработка данных
- Создание нескольких view в один момент
- И многое другое...

В бою то же время будет для гораздо меньшего количества узлов

Выводы

innerHTML достаточно быстрый, но медленный пост-процессинг

Построение DOM применяя API и оптимизации быстрее innerHTML

Шаблоны basis.js – удобный способ создавать и обслуживать DOM фрагменты

И эти шаблоны умеют много чего еще

Подключение стилей, live update, темы, локализация, интроспекция, инструменты, ...

Заключение

Изучайте и используйте DOM

Сравнивайте

Экспериментируйте

Ищите решения

И получится что-то крутое :)

Но это еще не все...

basis-templates.js

basis.js template & l10n modules
as standalone library

basis-templates.js

63.2 KB min
23.7 KB min + gzip

basisjs.com/templates

github.com/basisjs/templates

basis-templates.js

библиотека ориентирована на встраивание в системы и фреймворки, в виде модулей или плагинов

bbt.js

backbone basis-templates.js plugin

backbone.js – 510 ms

backbone.js + bbt.js – 202 ms

TodoMVC

	100 todo	1000 todo
AngularJS	125 ms	1491 ms
Backbone.js	53 ms	510 ms
Knockout	39 ms	489 ms
vanilla	23 ms	1882 ms
jQuery	20 ms	184 ms
Backbone.js + bbt.js	18 ms	202 ms
basis.js	8 ms	95 ms

Переделаны только шаблон и код TodoView для использования basis-templates.js

В планах

- Уменьшить объем
- Улучшить API
- Обеспечить поддержку live update, тем и других возможностей шаблонов basis.js
- Tasks для grunt (live update и сборка)
- Больше примеров и документации

basis-templates.js

Пробуйте, используйте, принимайте участие!

github.com/basisjs/templates

basis-templates.js

Сделаем веб быстрее!

github.com/basisjs/templates

Web Standards Days

Вопросы?

Роман Дворнов
@rdvornov
rdvornov@gmail.com

basis.js
basisjs.com
github.com/basisjs

Минск, Октябрь 2013