

Google Closure руководство для начинающих



Мне как всегда не сидится на месте, в этот раз я взялся за [Google Closure Library](#).

В заголовке красуется «для начинающих», но мы то знаем, если вы добрались до Closure, то ваш уровень уже далек от озвученного, хотя основы мы будем проходить по натопанному пути.

Подключение библиотеки

Для подключения Closure нам понадобится скачать библиотеку с [домашней странички проекта](#) на GitHub, и подключить используя простой синтаксис:

```
1 <script type="text/javascript" src="js/closure/goog/base.js"></script>
```

Насчет создания единого CDN хранилища для своей библиотеки Google ответил кратко:

Closure Library differs from a lot of other JavaScript libraries in that it's very broad and large and, in general, every class or namespace is stored in its own file, so just what is needed can be included in an app. As such, it's infeasible to require users to pull all of Closure as they would another library in the AJAX APIs, as the download would be very large and split over a lot of individual requests. So we don't have any immediate plans for AJAX API support.

Хотя, если посмотреть внимательно на тот же Dojo Toolkit, то при похожей архитектуре у них есть возможность организации CDN из коробки.

Подключение пакетов

Closure состоит из отдельных пакетов, которые необходимо подключать по отдельности:

```
1 goog.require('goog.dom');
```

```
2 goog.require( 'goog.events' );  
3 goog.require( 'goog.events.EventType' );
```

Про расчет зависимостей между пакетами — не стоит беспокоиться, они позаботятся о себе самостоятельно (по крайней мере постараются, если им [прописаны зависимости](#)).

Поиск DOM элементов

Тут все просто и очень похоже на ... обычный JavaScript, т.е. универсального вращера который нам предоставляет jQuery у нас нет, а есть пару пакетов:

Ну теперь непосредственно о выборках:

```
1  
2  
3  
4 goog.dom.getElement( 'header' )  
5 goog.dom.$( 'header' )  
6 goog.dom.getElementByClass( 'myclass' )  
7 goog.dom.getElementsByClass( 'myclass' )  
8  
9
```

Очень часто придется использовать метод `getElementsByTagNameAndClass`, т.к. он достаточно универсален:

```
01  
02  
03  
04  
05  
06  
07 goog.dom.getElementsByTagNameAndClass()  
    goog.dom.getElementsByTagNameAndClass( 'p', 'myclass' )
```

```
08 goog.dom.$$('p', 'myclass')
09 goog.dom.$$('div')
10 goog.dom.$$(null, 'myclass')
11 goog.dom.$$('div', 'myclass', goog.dom.$('id'))
12
13
14
15
16
17
```

Если вам и этого показалось мало, тут на помощь приходит [document.QuerySelectorAll](#), ну а точнее его врапер, который основывается на коде Dojo Toolkit (кому нужны эти подробности?):

```
1 goog.dom.query('h1+h2, a:last-child')
2
```

Еще стоит упомянуть поиск элементов среди предков:

```
01
02
03
04
05 goog.dom.getAncestorByTagNameAndClass(goog.dom.$('id'))
06 goog.dom.getAncestorByTagNameAndClass(goog.dom.$('id'), 'div')
07 goog.dom.getAncestorByTagNameAndClass(goog.dom.$('id'), null, 'myclass')
08 goog.dom.getAncestorByTagNameAndClass(goog.dom.$('id'), 'div', 'myclass')
09
10
11
```

Манипуляции с DOM

Кроме как взять готовое, можно создать DOM элемент ручками:

```
1
2  var newHeader = goog.dom.createDom('h2', {'style':'color:red;', 'class':'title'},
3    'Hello World!');
4  var newHeader = goog.dom.$dom('h2', {'style':'color:red;', 'class':'title'},
5    'Hello World!');
6  goog.dom.appendChild(document.body, newHeader);
```

Изменяем классы и стили

Ничего сложного, приведу лишь наглядные примеры, начну с работы с классами:

```
01
02
03
04
05
06
07
08
09
10
11  goog.require('goog.dom.classes');
12  goog.dom.classes.set(goog.dom.$('id'), 'className')
13  goog.dom.classes.get(goog.dom.$('id'))
14  goog.dom.classes.add(goog.dom.$('id'), 'className', 'anotherClass')
15  goog.dom.classes.remove(goog.dom.$('id'), 'className', 'anotherClass')
16  goog.dom.classes.swap(goog.dom.$('id'), 'fromClass', 'toClass')
17  goog.dom.classes.addRemove(goog.dom.$('id'), 'addClass', ['removeClass'])
18  goog.dom.classes.has(goog.dom.$('id'), 'className')
```

```
18 goog.dom.classes.enable(goog.dom.$('id'), 'className', true || false)
19 goog.dom.classes.toggle(goog.dom.$('id'), 'className')
20
21
22
23
24
25
26
27
28
29
```

Работа со стилями тоже не многим сложнее:

```
01
02
03
04 goog.require('goog.style');
05 goog.style.getStyle(goog.dom.$('id'), 'color')
06 goog.style.setStyle(goog.dom.$('id'), 'color', '#ff0000')
07 goog.style.setStyle(goog.dom.$('id'), { 'margin-top': '20px', 'margin-
08 bottom': '20px' })
09
10
```

Поиграемся с размерами:

```
01
02
03
```

```
04 goog.style.getSize(goog.dom.$('id'))
05 goog.style.setSize(goog.dom.$('id'), width, height)
06 goog.style.setSize(goog.dom.$('id'), new goog.math.Size(width, height))
07 goog.style.setHeight(goog.dom.$('id'), height)
08 goog.style.setWidth(goog.dom.$('id'), width)
09
10
11
```

Расположение элементов:

```
1
2
3
4 goog.style.setPosition(goog.dom.$('id'), left, top)
5 goog.style.setPosition(goog.dom.$('id'), new goog.math.Coordinate(x, y))
6 goog.style.getPosition(goog.dom.$('id')) Opacity manipulation:
7
8
```

Прозрачность и видимость элементов:

```
01
02
03
04
05 goog.style.setOpacity(goog.dom.$('id'), 0.5)
06 goog.style.getOpacity(goog.dom.$('id'))
07 goog.style.setStyle(goog.dom.$('id'), "display", "none");
08 goog.style.showElement(goog.dom.$('id'), false);
09 goog.style.setStyle(goog.dom.$('id'), "display", "");
```

```
10 goog.style.showElement(goog.dom.$('id'), true);  
11  
12  
13  
14
```

Работаем с событиями

Если вы работали с событиями в JavaScript'е или в jQuery, то для вас тут будет все знакомо, пожалуй стоит упомянуть, что привычного по jQuery DOM ready в Closure нет:

The short story is that we don't want to wait for DOMContentLoaded (or worse the load event) since it leads to bad user experience. The UI is not responsive until all the DOM has been loaded from the network. So the preferred way is to use inline scripts as soon as possible.

Хотя, есть возможность повесить обработчик, но его не принято использовать:

```
1 goog.events.listen(window, goog.events.EventType.LOAD, function() {  
2  
3 });
```

Это немного сбивает с толку, и вы даже можете встретить использование `<body load="...">`. В остальном, в Closure велосипед не изобрели и работа с событиями похожа на большинство существующих фреймворков, вот что нам необходимо сделать:

Чуть-чуть пояснений:

- **src** — источник события, обычно это некий DOM элемент, но по факту любой объект с методом `dispatchEvent()`
- **type** — имя события, или массив имен
- **listener** — наша функция-обработчик события
- **opt_capture** — необязательны булевый параметр, отвечает за то, в какой момент будет происходить перехват события — всплытия или погружения
- **opt_handler** — необязательный параметр, позволяет перебиндить `this` для

нашей функции (так которая listener)

Еще немного примеров:

```
01
02
03
04
05
06 goog.events.listenOnce(goog.dom.$('id'), 'click', function(event){ } );
07 var key = goog.events.listen(goog.dom.$('id'), 'click', myFunction);
08 goog.events.unlistenByKey(key);
09 goog.events.listen(goog.dom.$('id'), 'click', myFunction, true);
10 goog.events.unlisten(goog.dom.$('id'), 'click', myFunction, true);
11 goog.events.removeAll(goog.dom.$('id'));
12 goog.events.removeAll();
13
14
15
16
17
```

Ух, много текста, но вы не отчаивайтесь, сохраните лучше [шпаргалку](#) в избранном ;)

Разрабатываем пакет

Когда решите, что в вашем проекте требуется нечто большее, чем всплывающие подсказки, то значит вы созрели для написания своих пакетов, и да это тоже не сложно. Стоит начать приблизительно со следующего кода:

```
01
02
03
```



```
04 goog.provide('tutorial.simple');
05 goog.require('goog.dom');
06 goog.require('goog.style');
07
08 tutorial.simple = function() {
09
10 };
11
12
13
```

Теперь стоит немного расширить функционал, надо бы получить какой-то элемент и провести с ним какие-нибудь махинации:

```
01
02
03
04
05
06 tutorial.simple = function(el) {
07     this.el = el;
08 };
09
10
11 tutorial.simple.prototype.el = null;
12
13
14 tutorial.simple.prototype.red = function(color) {
15     goog.style.setStyle(this.el, {"color":color});
16 };
17
```

17

18

Теперь как работать с этим кодом-то:

```
1 <script type="text/javascript" src="js/closure/goog/base.js"></script>
2 <script type="text/javascript" src="js/simple.js"></script>
3 <script type="text/javascript">
4   var simple = tutorial.simple(document.getElementById("header"))
5     simple.red();
6 </script>
```

Если у вас прописаны зависимости, то вторая строчка преобразуется в лаконичную `goog.require('tutorial.simple');`

AJAX

Как же это, писать статью о JavaScript фреймворке и не упомянуть работу с AJAX — конечно же тут есть свой врапер, работа с ним дело не хитрое и каждому по плечу, вот [пример](#):

```
01
02
03
04
05 goog.require('goog.net.XhrIo');
06 goog.net.XhrIo.send('ajax/response.json', function(ev){
07
08     var data = ev.target.getResponseJson();
09
10     var container = goog.dom.$('articles');
11
12     goog.array.forEach(data.articles, function(article){
```

```

13
14     var title = goog.dom.$dom('h2', {'innerHTML':article.title});
15     var content = goog.dom.$dom('p', {'innerHTML':article.content});
16
17
18     goog.dom.appendChild(container, title);
19     goog.dom.appendChild(container, content);
20   })
21 }, 'POST', 'give=data&or=lost');
22
23
24
25

```

Хватит наверное уже вступительной информации, пора переходить к примерам... Хотя вся приведенная информация удобно представлена в виде шпаргалки

Выдвижная панель

Простой пример, для начала — слайд-панель, она у нас будет двигаться вверх/вниз по клику на ссылке (см. [пример](#)):



Для реализации данного функционала нам понадобится пакет "[AnimatedZippy](#)":

```

1
2

```

```

3 goog.require('goog.dom');
4 goog.require('goog.ui.AnimatedZippy');
5 goog.require('goog.ui.Zippy');
6 goog.require('goog.ui.ZippyEvent');

```

После создания DOM элементов, т.е. внизу HTML страницы пишем код, который должен состоять из одной строки, если бы не мои комментарии:

Для того, чтобы у нас был индикатор со стрелочкой, необходимо описать следующий CSS (за применение данных стилей отвечает сама пакет):

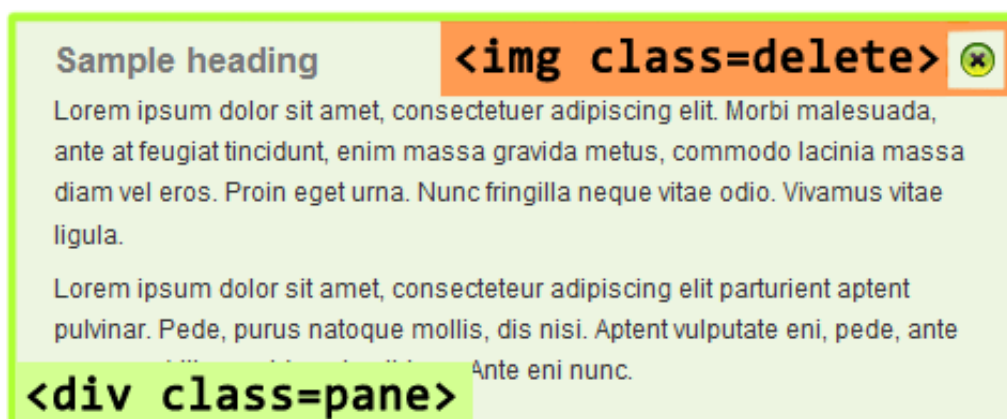
```

1
2 .goog-zippy-expanded {
3     background: url(images/white-arrow.gif) no-repeat right 12px;
4 }
5 .goog-zippy-collapsed {
6     background: url(images/white-arrow.gif) no-repeat right -50px;
7 }
8

```

Магические исчезновения

Еще один незамысловатый пример код — пришел, увидел, [скрыл с глаз долой](#):



Логика элементарна — при клике на картинке с class="delete" находим

родительский элемент и скрываем его:

```
01
02
03 goog.require('goog.dom');
04 goog.require('goog.array');
05 goog.require('goog.events');
06 goog.require('goog.events.EventType');
07 goog.require('goog.fx');
08 goog.require('goog.fx.dom');
09 goog.array.forEach(goog.dom.getElementsByClass('delete'), function(el) {
10
11     goog.events.listen(
12         el,
13         goog.events.EventType.CLICK,
14         function(ev) {
15             var anim = new goog.fx.dom.FadeOutAndHide(el.parentNode, 400);
16             anim.play();
17             ev.preventDefault();
18         });
19 });
20
```

Анимация

Более сложный пример, будет заставлять квадрат двигаться, изменять размер и прозрачность (см. [пример](#)):

Run

Для этого нам понадобятся следующие пакеты:

```
1 goog.require('goog.fx');
2 goog.require('goog.fx.dom');
3 goog.require('goog.fx.AnimationQueue');
4 goog.require('goog.fx.AnimationSerialQueue');
5 goog.require('goog.fx.AnimationParallelQueue');
6
```

Теперь непосредственно код, который будет отвечать за все превращения и перемещения:

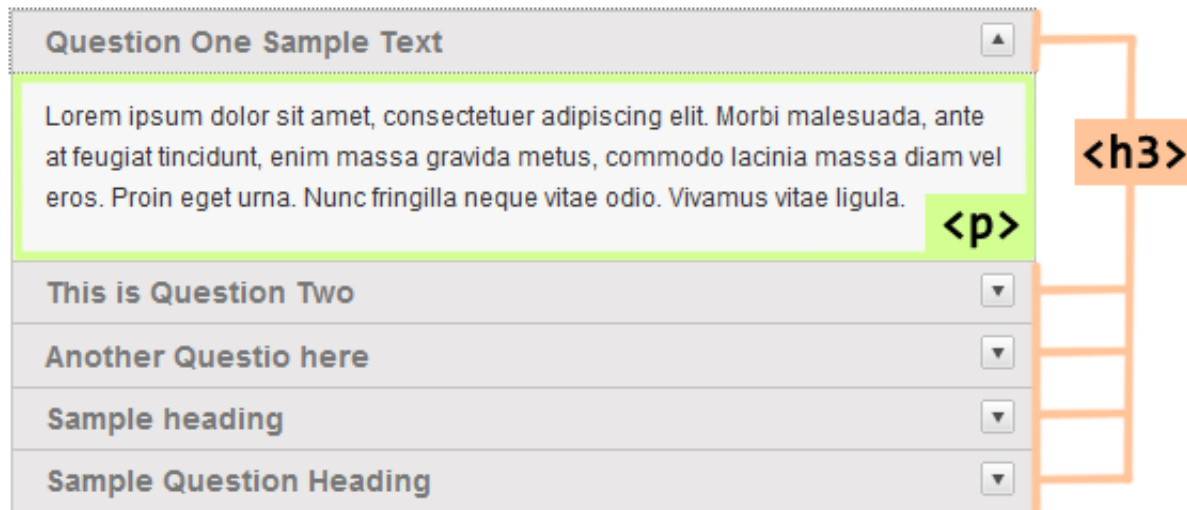
```
01
02
03
04
05 function run() {
06
07     var elem = goog.dom.getElement("box");
08
09     var queue = new goog.fx.AnimationSerialQueue();
10
11     queue.add(new goog.fx.dom.Fade(elem, 1, 0.1, 1200));
12
13     queue.add(new goog.fx.dom.SlideFrom(elem,
```

```
14         [400, 100], 2000,  
15         goog.fx.easing.easeOut  
16     ));  
17  
18     queue.add(animationInner(elem));  
19  
20  
21     queue.add(new goog.fx.dom.SlideFrom(elem,  
22         [100, 100], 500,  
23         goog.fx.easing.easeOut  
24     ));  
25  
26     queue.play();  
27 }  
28 function animationInner(elem){  
29  
30     var innerAnim = new goog.fx.AnimationParallelQueue();  
31  
32     innerAnim.add(new goog.fx.dom.Fade(elem, 0.1, 0.5, 2000));  
33  
34     innerAnim.add(new goog.fx.dom.SlideFrom(elem,  
35         [400, 300], 2000,  
36         goog.fx.easing.easeOut  
37     ));  
38  
39     innerAnim.add(new goog.fx.dom.Resize(elem,  
40         [100, 100], [20, 20], 2000,  
41         goog.fx.easing.easeOut  
42     ));  
  
    return innerAnim;
```

```
43 }  
44 function animationInner2(elem){  
45     var innerAnim = new goog.fx.AnimationParallelQueue();  
46  
47     innerAnim.add(new goog.fx.dom.FadeIn(elem, 1200));  
48  
49     innerAnim.add(new goog.fx.dom.SlideFrom(elem,  
50         [100, 300], 1200,  
51         goog.fx.easing.easeOut  
52     ));  
53  
54     innerAnim.add(new goog.fx.dom.Resize(elem,  
55         [20, 20], [100,100], 1200,  
56         goog.fx.easing.easeOut  
57     ));  
58     return innerAnim;  
59 }  
60  
61  
62  
63
```

Вот такой код, как мы можем заметить, кода поболее чем [при использовании jQuery](#).

Гармошка



Знаю по jQuery, пример создания «гармошки» из элементов пользуется популярностью, в данном руководстве я тоже не смог обойти его стороной, благо это не так трудно, нам необходимо: при клике на заголовке открыть последующий параграф; при повторном клике — скрыть; одновременно открытым должен быть лишь один элемент.

Для реализации, нам потребуются следующие пакеты:

```
1 goog.require('goog.ui.AnimatedZippy');
2 goog.require('goog.ui.Zippy');
3 goog.require('goog.ui.ZippyEvent');
```

Ну и сама реализация с комментариями:

```
01
02
03
04 var panels = goog.dom.$$('h3');
05 var anims  = {};
06 var open   = null;
07 goog.array.forEach(panels, function(pane){
08
09     var animation = new goog.ui.AnimatedZippy(pane,
10     goog.dom.getNextElementSibling(pane));
```

```
11
12
13     goog.events.listen(animation, goog.ui.Zippy.Events.TOGGLE, zippyToggle);
14
15     anims[goog.getUid(animation)] = animation;
16 });
17
18 function zippyToggle(event) {
19
20     var uid = goog.getUid(event.target);
21
22     if (event.expanded) {
23         if (open) {
24             anims[open].setExpanded(false);
25         }
26         open = uid;
27     }
28 }
29
```

Гармошка, вариант 2

Небольшое изменение предыдущего варианта — сделаем открытие/заккрытие одновременным, и [откроем панельку](#) по умолчанию, для этого потребуются не так много изменений в коде:

```
01
02 var def      = 2;
03 goog.array.forEach(panels, function(pane, ind ){
04
05     var animation = new goog.ui.AnimatedZippy(pane,
    goog.dom.getNextElementSibling(pane), (def==ind));
```

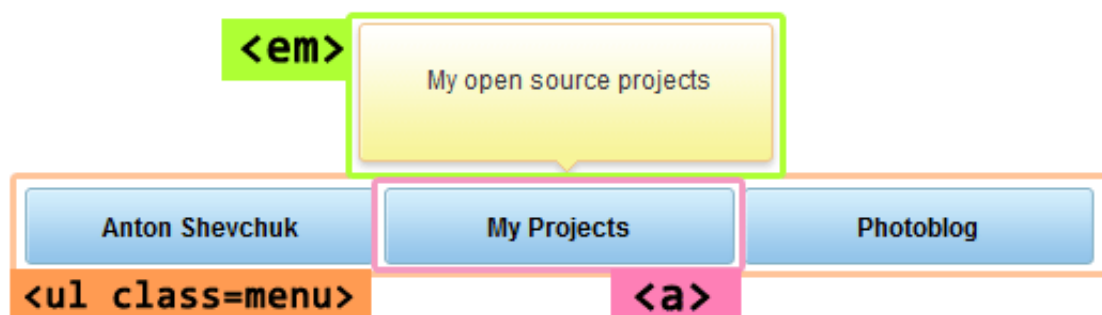
```

06
07     goog.events.listen(animation, goog.fx.Animation.EventType.BEGIN,
08         zippyToggle);
09     anims[goog.getUid(animation)] = animation;
10 }));

```

Анимация для события hover

Данный [пример](#) реализует анимацию для события hover — в меру симпатично и функционально:



```

<ul class="menu">
  <li>
    <a href="http://anton.shevchuk.name">Anton Shevchuk</a>
    <em>Personal Blog</em>
  </li>
  <li>
    <a href="http://hohli.com">My Projects</a>
    <em>My open source projects</em>
  </li>
  <li>
    <a href="http://photo.hohli.com">Photoblog</a>
    <em>Full size photos under CC</em>
  </li>
</ul>

```

Для реализации потребуется:

```

01
02
03
04 goog.require('goog.dom');
05 goog.require('goog.dom.query');
06 goog.require('goog.array');

```

```
06 goog.require('goog.events');
07 goog.require('goog.events.EventType');
08 goog.require('goog.fx');
09 goog.require('goog.fx.dom');
10 goog.require('goog.fx.AnimationQueue');
11 goog.require('goog.fx.AnimationSerialQueue');
12 goog.require('goog.fx.AnimationParallelQueue');
13
14
```

Код реализации:

```
01
02
03
04 goog.array.forEach(goog.dom.query('.menu a'), function(el){
05
06     goog.dom.getNextElementSibling(el).style.display = 'none';
07
08     goog.events.listen(el, goog.events.EventType.MOUSEOVER, function(ev){
09
10         var em = goog.dom.getNextElementSibling(this);
11
12         var anim = new goog.fx.AnimationParallelQueue();
13         anim.add(new goog.fx.dom.FadeInAndShow(em, 1500));
14         anim.add(new goog.fx.dom.Slide(em,
15             [-15, -85], [-15, -75], 1500,
16             goog.fx.easing.easeOut
17             ));
18         anim.play();
19     });
20 }
```

```
19     }, false, el);
20
21     goog.events.listen(el, goog.events.EventType.MOUSEOUT, function(ev){
22
23         var em = goog.dom.getNextElementSibling(this);
24
25         var anim = new goog.fx.AnimationParallelQueue();
26
27         anim.add(new goog.fx.dom.FadeOutAndHide(em, 500));
28
29         anim.add(new goog.fx.dom.Slide(em,
30             [-15, -75], [-15, -85], 500,
31             goog.fx.easing.easeOut
32             ));
33
34         anim.play();
35     }, false, el);
36 })
```

Анимация для события hover. Вариант 2

Наличие лишнего тега `` лишь для подсказки меня не радует, куда как лаконичней использовать атрибут `title` тега `<a>`:

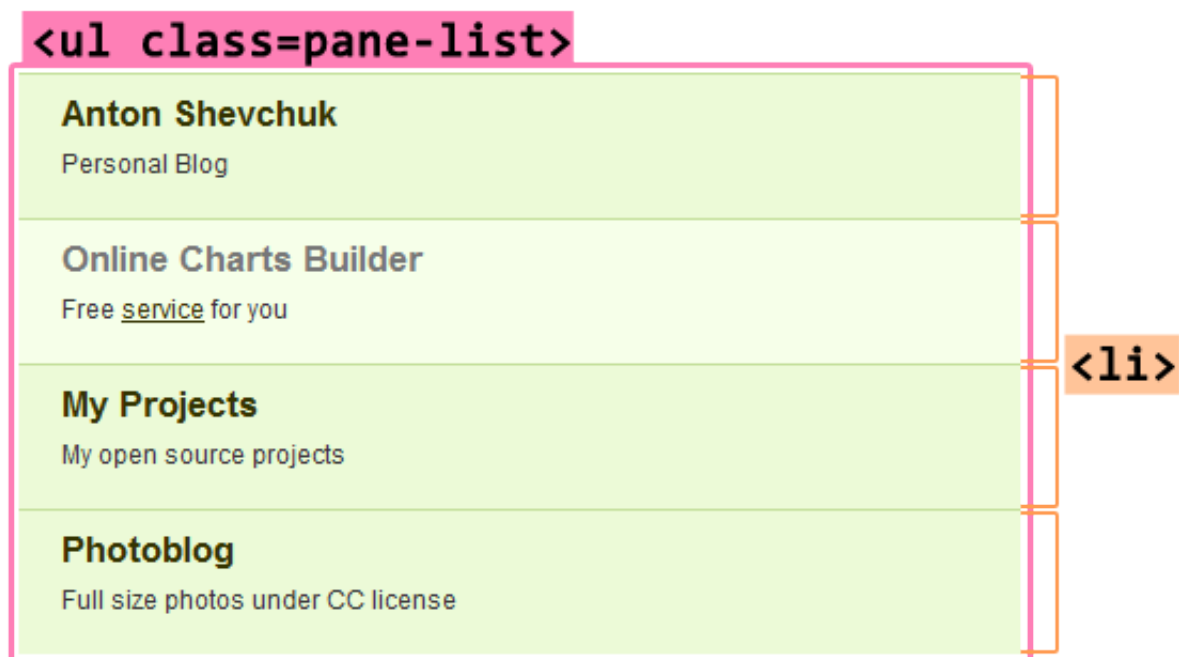
Теперь посредством Javascript'a создадим элемент ``, и добавим в существующий код:

```
1
2 goog.array.forEach(goog.dom.query('.menu a'), function(el){
3     goog.dom.append(el.parentNode, goog.dom.$dom('em', {'style':'display:none;',
4     'innerHTML':el.title}));
5 });
```

[Результат](#) не отличим визуально от предыдущего.

Кликабельные блоки

Теперь решим следующую задачу, у нас есть блок текста, внутри которого есть ссылка, при клике на ссылку должен осуществляться переход по ссылке:



Подключаем пакеты:

```
1 goog.require('goog.dom');
2 goog.require('goog.array');
3 goog.require('goog.events');
4 goog.require('goog.events.EventType');
```

Создаем обработчик события для каждого ``:

```
1 goog.array.forEach(goog.dom.$$('li'), function(el){
2     goog.events.listen(el, goog.events.EventType.CLICK, function(ev){
3
4         var link = goog.dom.$$('a', null, this)[0];
5     });
6 });
```

```
6         window.location = link.href;
7         ev.preventDefault();
8     }, false, el);
9 });
```

Смотрим на пример в живую.

Складывающиеся панельки

Данный [пример](#) чем-то схож с «гармошкой», но добавлено немного полезны фич:



Задача требует от нас следующих действий:

- скрываем все элементы `<div class=message_body>` после первого
- скрываем все элементы `` после пятого
- клик по `<p class=message_head>` — вызывает изменение отображение (показываем/прячем) для следующего элемента `<div class=message_body>`
- клик по `` — скрывает все `<div class=message_body>`
- клик по `` — скрывает элемент, и отображает ``, так же скрывает все элементы `` после пятого
- клик по `` — скрывает элемент, и отображает ``, так же отображает все `` после пятого

Нам потребуются:

```
01
02
03
04 goog.require('goog.dom');
05 goog.require('goog.dom.query');
06 goog.require('goog.array');
07 goog.require('goog.events');
08 goog.require('goog.events.EventType');
09 goog.require('goog.style');
10 goog.require('goog.ui.AnimatedZippy');
11 goog.require('goog.ui.Zippy');
12 goog.require('goog.ui.ZippyEvent');
13
14
15
```

Ну и листинг кода с комментариями:

```
01
02
03
04
05
06
07
08
09 function showMore(flag) {
10
11     goog.array.forEach(goog.dom.query('#messages li:nth-child(n+6)'),
12     function(el){
```



```
13         el.style.display = (flag?'':'none');
14     });
15
16     goog.style.showElement(goog.dom.$('show_recent_only'), flag);
17     goog.style.showElement(goog.dom.$('show_all_message'), !flag);
18 }
19 showMore(false);
20 goog.style.showElement(goog.dom.$('show_recent_only'), false);
21 var zippy = [];
22 goog.array.forEach(goog.dom.$$('h4',null,goog.dom.$('messages')), function(pane,
23 index){
24     zippy.push(new goog.ui.AnimatedZippy(pane,
25     goog.dom.getNextElementSibling(pane), (index==0)));
26 });
27 goog.events.listen(goog.dom.$('collapse_all_message'), 'click', function(ev){
28
29     goog.array.forEach(zippy, function(anim){
30
31         anim.setExpanded(false);
32     });
33     ev.preventDefault();
34 });
35 goog.events.listen(goog.dom.$('show_all_message'), 'click', function(ev){
36     showMore(true);
37     ev.preventDefault();
38 });
39 goog.events.listen(goog.dom.$('show_recent_only'), 'click', function(ev){
40     showMore(false);
41     ev.preventDefault();
42 });
43 }
```

42

43

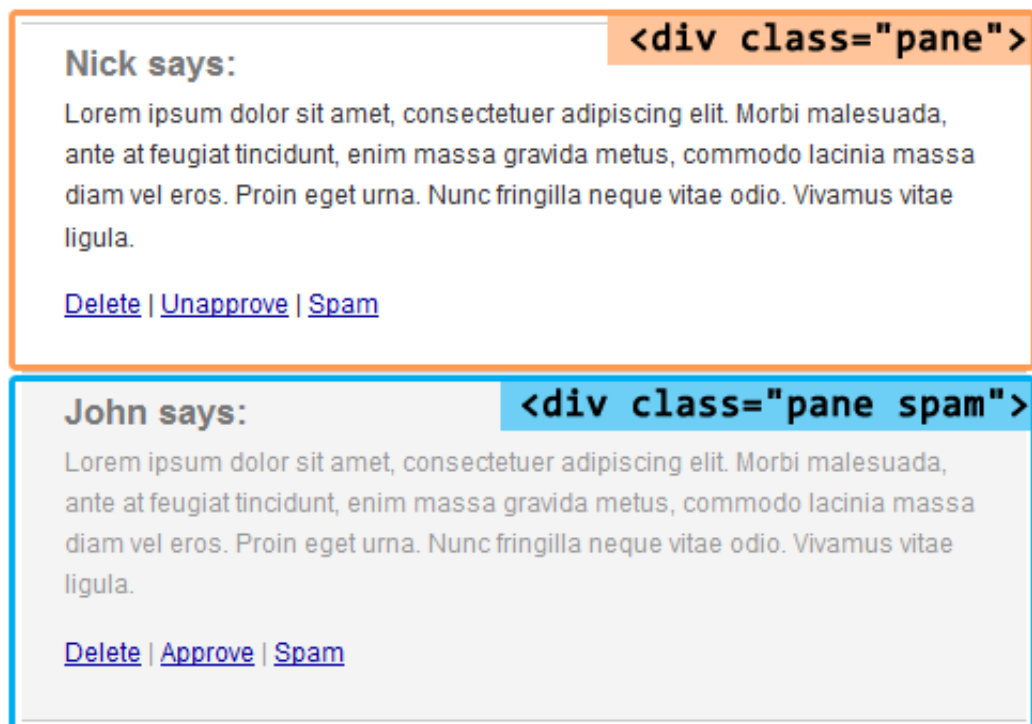
44

45

46

Управление комментариями (для WordPress)

Еще один пример — [управление комментариями](#) в стиле wordpress:



Для этого потребуется:

```
1 goog.require('goog.dom');
2 goog.require('goog.dom.query');
3 goog.require('goog.dom.classes');
4 goog.require('goog.array');
5 goog.require('goog.fx');
6 goog.require('goog.fx.dom');
7 goog.require('goog.events');
8 goog.require('goog.events.EventType');
```

Ну и листинг кода:

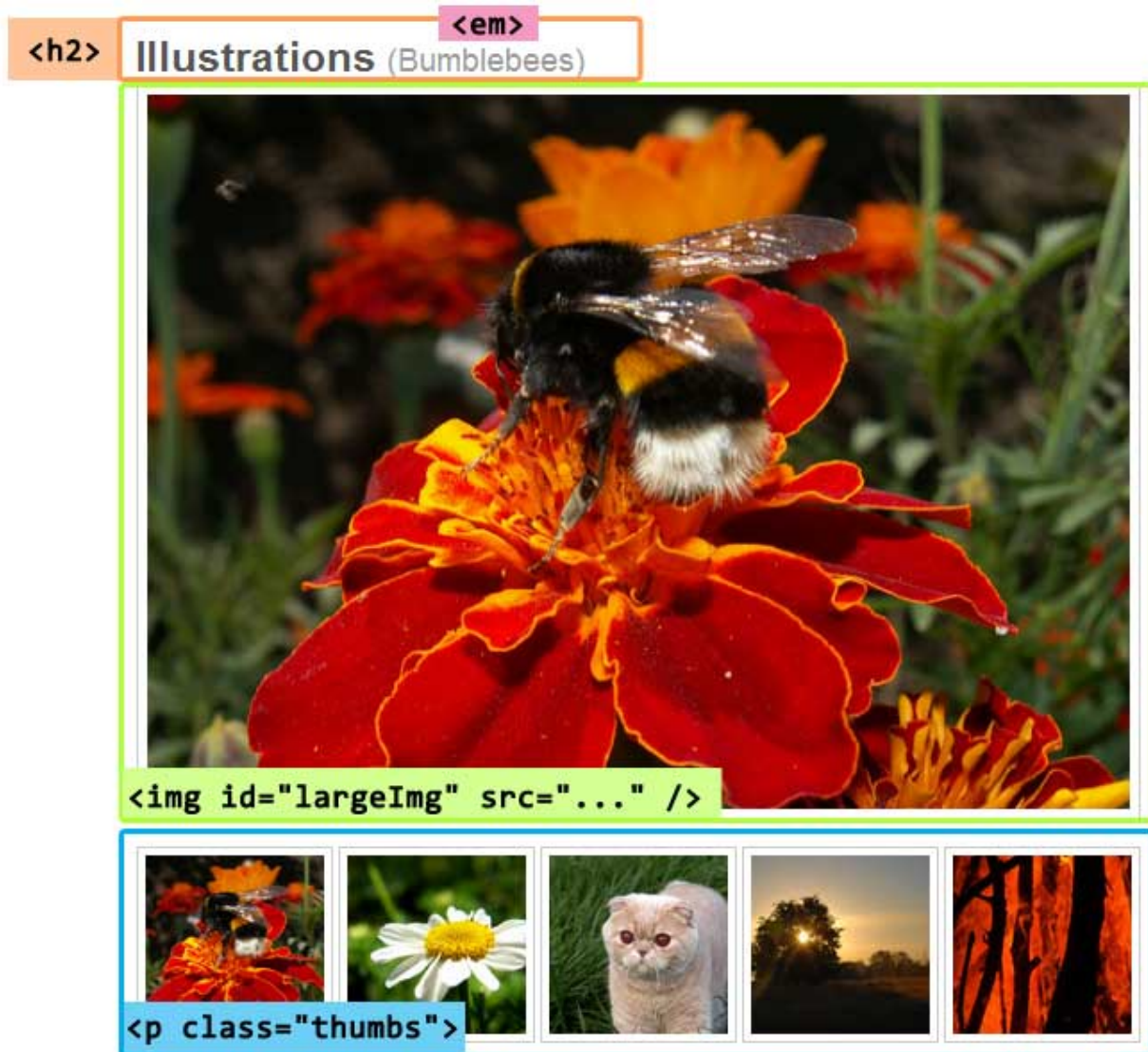
```
01
02
03
04
05
06
07
08
09
10
11
12
13 goog.array.forEach(goog.dom.query('.pane:nth-child(2n)'), function(el) {
14     goog.dom.classes.add(el, 'alt');
15 });
16 function unapprove(el) {
17
18     goog.events.listen(el, goog.events.EventType.CLICK, function(event){
19
20         var pane = goog.dom.getAncestorByTagNameAndClass(event.target, 'div',
21 'pane');
22
23         goog.fx.dom.bgColorFadeIn(pane, [255, 240, 120], 2000);
24
25         goog.dom.classes.add(pane, 'spam');
26         event.preventDefault();
27
28         var button = goog.dom.createDom('a', {'href':'#','class':'btn-
29 approve','innerHTML':'Approve'});
```

```
28         approve(button);
29
30
31         goog.dom.replaceNode(button, event.target);
32     });
33 }
34 function approve(el) {
35
36     goog.events.listen(el, goog.events.EventType.CLICK, function(event){
37
38         var pane = goog.dom.getAncestorByTagNameAndClass(event.target, 'div',
39 'pane');
40
41         goog.fx.dom.bgColorFadeIn(pane, [200, 240, 180], 2000);
42
43         goog.dom.classes.remove(pane, 'spam');
44         event.preventDefault();
45
46         var button = goog.dom.createDom('a', {'href':'#','class':'btn-
47 unapprove','innerHTML':'Unapprove'});
48
49         unapprove(button);
50
51         goog.dom.replaceNode(button, event.target);
52     });
53 }
54 goog.array.forEach(goog.dom.query('.pane .btn-unapprove'), unapprove);
55 goog.array.forEach(goog.dom.query('.pane .btn-approve'), approve);
56 goog.array.forEach(goog.dom.query('.pane .btn-spam'), function(el) {
57     goog.events.listen(el, goog.events.EventType.CLICK, function(event){
58
59         var pane = goog.dom.getAncestorByTagNameAndClass(event.target, 'div',
60 'pane');
```

```
57
58     goog.fx.dom.bgColorFadeIn(pane, [250, 200, 200], 2000);
59
60     new goog.fx.dom.FadeOutAndHide(pane, 500).play();
61     event.preventDefault();
62 });
63 });
64 goog.array.forEach(goog.dom.query('.pane .btn-delete'), function(el) {
65     goog.events.listen(el, goog.events.EventType.CLICK, function(event){
66         alert("This comment will be deleted!");
67
68         var pane = goog.dom.getAncestorByTagNameAndClass(event.target, 'div',
69 'pane');
70
71         goog.fx.dom.bgColorFadeIn(pane, [250, 200, 200], 2000);
72
73         new goog.fx.dom.FadeOutAndHide(pane, 500).play();
74         event.preventDefault();
75     });
76 });
77
78
79
80
81
82
83
84
85
```

Галерея изображений

Простейший пример реализации галереи, без перезагрузки страницы. (см. [пример](#)):



Листинг с комментариями:

```


01
02
03
04 goog.require( 'goog.dom' );
05 goog.require( 'goog.dom.query' );
    goog.require( 'goog.array' );


```


```
06 goog.require('goog.style');
07 goog.require('goog.events');
08 goog.array.forEach(goog.dom.query('.thumbs a'), function(el) {
09
10     goog.events.listen(el, 'click', function(ev){
11
12         ev.preventDefault();
13
14         goog.dom.setProperties(goog.dom.$("largeImg"), { src: this.href, alt:
15             this.title });
16
17         goog.dom.query('h2 em')[0].innerHTML = "("+this.title+")";
18     }, false, el);
19 });
20
```

Стилизируем ссылки

Данный [пример](#) хорошо иллюстрирует работу с CSS селекторами, в данном случае — необходимо изменить класс ссылок в тексте в зависимости от того куда она ведет:


 [PNG file](#) (icons preview)

 [ZIP file](#) (icons)

 [PDF file](#) (presentation)

[#anchor link](#) (#anchor)

[Anton Shevchuk](http://anton.shevchuk.name) (http://anton.shevchuk.name)

 [My Projects](http://hohli.com) (http://hohli.com)

Для реализации данной задачи нам потребуются:

```
1 goog.require('goog.dom');
2 goog.require('goog.dom.query');
3 goog.require('goog.array');
```

```
4 goog.require('goog.style');
```

И следующий код:

Использование UI

В Closure так же есть множество пакетов для быстрой реализации пользовательских интерфейсов, собственно приведу небольшой список из всего [обилия](#):

- [CSS 3 Buttons](#)
- [Checkbox](#)
- [Combobox](#)
- [Dimension picker](#)
- [Menu Buttons](#)
- [Select](#)
- [Tab Bar](#)
- [WYSIWYG Editor](#)
- [Progress bar](#)
- [Slider](#)
- [Datepicker](#)

Все UI пакеты имеют два метода `render()` и `decorate()` — первый создаёт все необходимые DOM элементы в указанном контейнере, второй же использует заранее созданную структуру и пытается с ней работать. Хотя на этом останавлиюсь, если получится — то о UI я расскажу в рамках следующей статьи.

P.S.

Обычно, после подобных статей, у меня спрашивают — «А лучше ли этот фреймворк, чем jQuery?». И я отвечаю — это разного уровня фреймворки, и предназначены для решения различных задач, если вы успешно справляетесь с использованием jQuery и плагинов, то не стоит сильно заморачиваться, ведь всё и так работает. Если же говорить о Dojo Toolkit — это как двоюродный брат, много общих черт, но они таки разные. И да, те кто говорит, что по Dojo мало документации, то вы наверное не читали документацию по Closure ;)