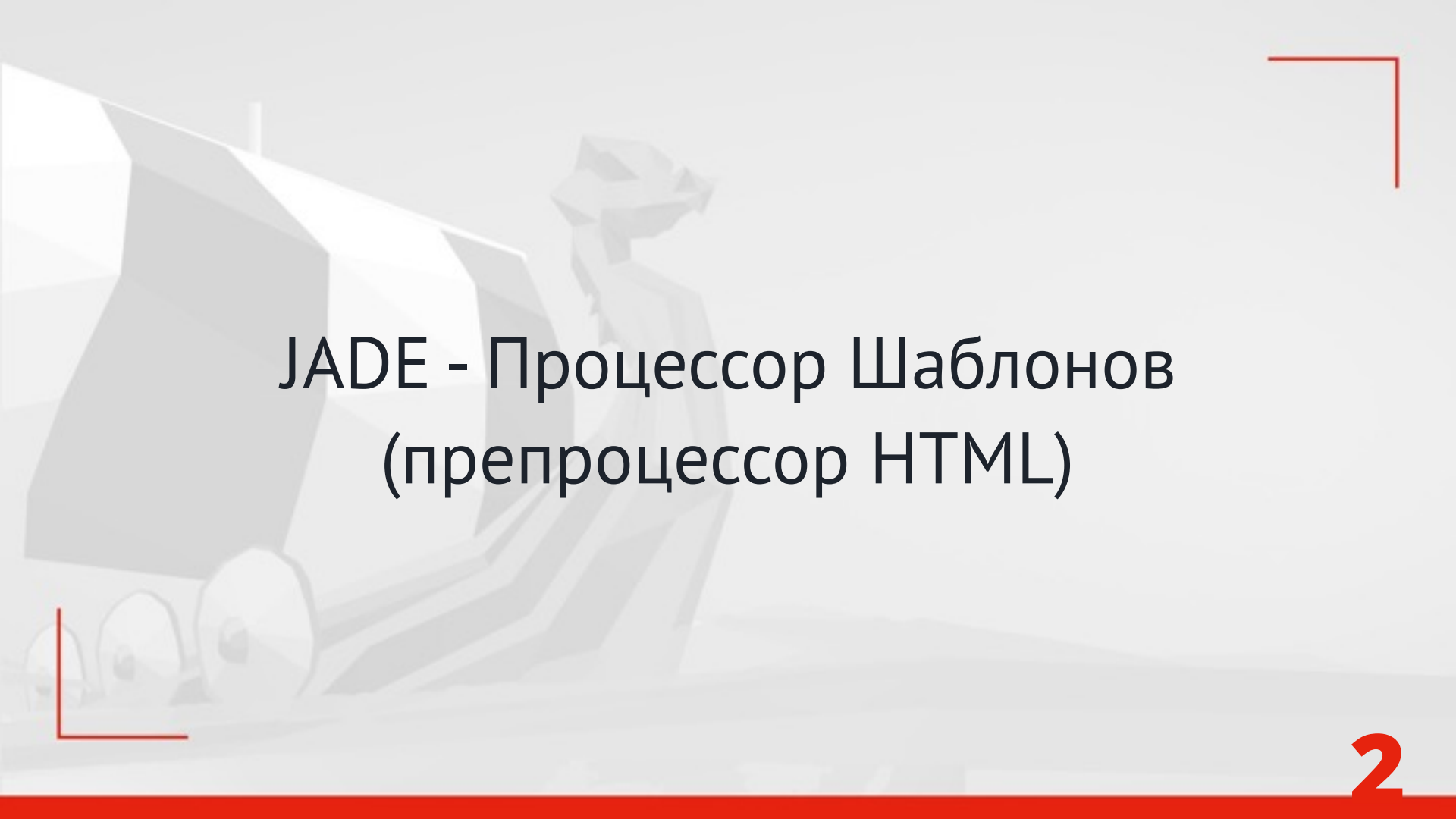




Верстка #2



JADE - Процессор Шаблонов (препроцессор HTML)

Зачем?

- Вынесение общих блоков в отдельные файлы
- Использование циклов и условий
- Использование массивов и объектов
- Создание и наследование шаблонов страниц
- Определение собственных функций (примесей)
- Ремонтопригодность
- Исчезает возможность не закрыть тег

Jade - в разы ускоряет процесс разработки и делает написание разметки удобным.





Прощай, закрывающий тег!

Что бы указать тег в **jade**, необходимо просто написать его имя.
Если тег не указан то считается что это div.

jade

```
// создаст <span class="text"></span>
```

```
span.text
```

```
// создаст <div class="block"></div>
```

```
.block
```



Содержимое

Передать содержимое блока можно несколькими способами

jade

```
// обычно делают так
.block Текст внутри блока

// а можно еще так
.block
  | Текст внутри блока

// или так
.block.
  Текст внутри блока
```



Вложенность блоков

Что бы указать джейд вложенность блоков нужно перенести блок на новую строку и “отбить” табуляцией.

jade

```
.block
  .inner-block Текст внутреннего блока

.sibling-block
  | Текст соседнего блока
```

html

```
<div class="block">
  <div class="inner-block">
    Текст внутреннего блока
  </div>
</div>

<div class="sibling-block">
  Текст соседнего блока
</div>
```



Атрибуты

Атрибуты передаются в скобках перед определением класса

jade

```
img(src="", alt="второй атрибут").img-class
```

```
// или так, если много атрибутов
```

```
input(  
  type="text"  
  placeholder="Плейсхолдер"  
  required  
).input-class
```

html

```
<div class="block">  
  <div class="inner-block">  
    Текст внутреннего блока  
  </div>  
</div>  
  
<div class="sibling-block">  
  Текст соседнего блока  
</div>
```



Компилируем

Для начала нужно Jade установить

```
npm install jade --global
```

Компилировать можно стандартными средствами (**CLI**)

```
jade dir_name
```

Компилирует все файлы в папке

```
jade --watch --pretty dir_name
```

Следит за изменениями файлов в папке

```
jade --watch --pretty jade_dir --out html_dir
```

Следит за изменениями и перекладывает в нужную папку



Gulp!

```
npm install gulp-jade --save-dev
```

jade

```
var gulp = require('gulp');
var jade = require('gulp-jade');

gulp.task('jade', function() {
  var YOUR_LOCALS = {}; // можно подключить JSON с данными

  gulp.src('./jadePath/*.jade')
    .pipe(jade({
      locals: YOUR_LOCALS,
      pretty: '\t', // отступы в 1 таб
    }))
    .pipe(gulp.dest('./htmlPath/'));
});
```

Структурирование!

- `include`
- `extends`
- `mixin`
- Массивы
- Объекты
- Циклы

Jade - позволяет создавать поддерживаемую структуру проекта, и вносить изменения быстрее.





INCLUDE

Jade позволяет подключать сторонние файлы при помощи директивы **include** (расширение можно не указывать).

Например: макетов много - хедер один.



Вынесение общих блоков

jade

```
// header.jade
head
  title My Site
  script(src='/javascripts/jquery.js')
  script(src='/javascripts/app.js')

// footer.jade
#footer
  p Copyright (c) foobar

// - index.jade
doctype html
html
  include ./includes/head.jade
  body
    h1 My Site
    p Welcome to my super lame site.

  include ./includes/footer.jade
```

ШАПКА САЙТА
(header.jade)

НАВИГАЦИЯ
(nav.jade)

Контент страницы

САЙДБАР
(sidebar.jade)

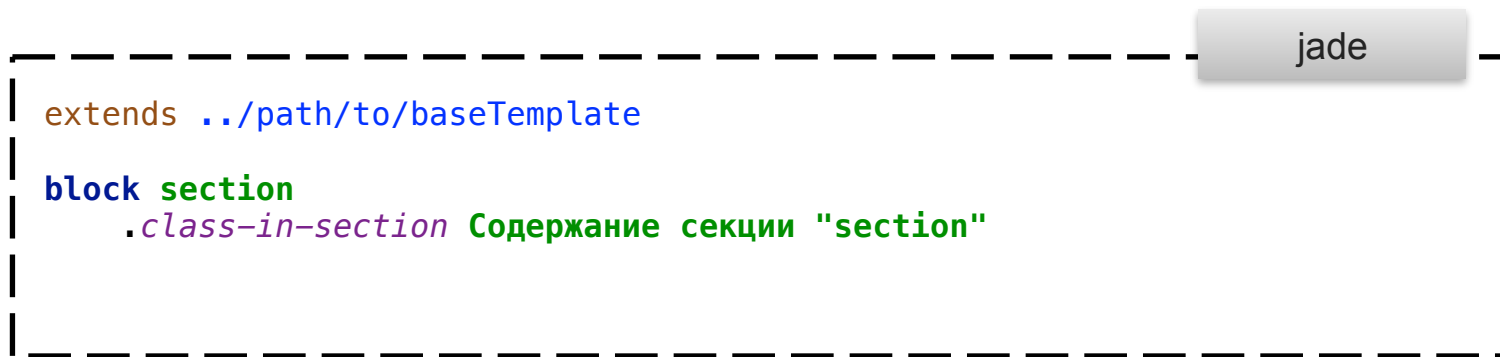
ПОДВАЛ
(footer.jade)

12



Наследование

Jade позволяет создавать и наследовать шаблоны директивой **extend**, а так же размечать изменяемые в нем секции директивой **block**



Extends & Block



jade

```
//- layout.jade
doctype html
html
  head
    block title
      title Default title

  body
    block content

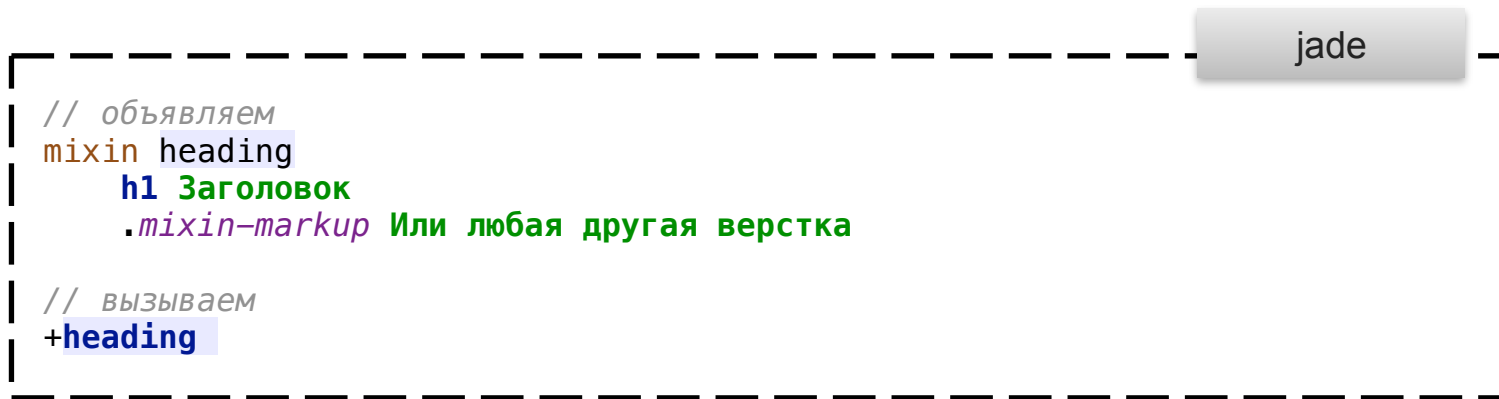
//- index.jade
extends ./layout.jade

block title
  title Article Title
block content
  h1 My Article
```



Примеси

Примеси в **Jade** - это функция которая может принимать параметры и возвращает разметку сгенерированную внутри этой функции.



Mixin



jade

```
mixin heading(articleText)  
  .article  
    .article-inner= articleText  
    .article-sibling  
  
+heading('Текст внутри')
```

html

```
<div class="article">  
  <div class="article-inner">  
    Текст внутри  
  </div>  
</div>  
<div class="article-sibling"></div>
```




Массивы

Конструкции **jade** понимает при помощи дэша (-)

jade

```
// массивы
- var array = ['one', 'two', 3];

// объекты
- var object = {'key' : 'one', 'prop' : 'two', 'option' : 3};

// объекты в несколько строк!
-
  var object = {
    'key' : 'one',
    'prop' : 'two',
    'option' : 3
  };

```



Массивы и объекты можно перебирать!

Директива **each** перебирает массивы и объекты

jade

```
// перебираем массив
each item, index in array
  .item-content= item
  .item-index индекс элемента в массиве = #{index}
  //0, 1, 2, 3...

// перебираем объект
each value, key in object
  .item-key= key
  .item-value= value
```



Обычный цикл

Циклы задаются директивой **for**

jade

```
ul.list
  - for (var i = 0; i < 4; i++)
    li.list-item= i
```

html

```
<ul class="list">
  <li class="list-item">0</li>
  <li class="list-item">1</li>
  <li class="list-item">2</li>
  <li class="list-item">3</li>
</ul>
```

Условия

Условия задаются директивой if

jade

```
ul.list
  - for (var i = 0; i < 4; i++)
    if i != 2
      li.list-item= i
    else
      li.list-item
        | условие отработало
```

html

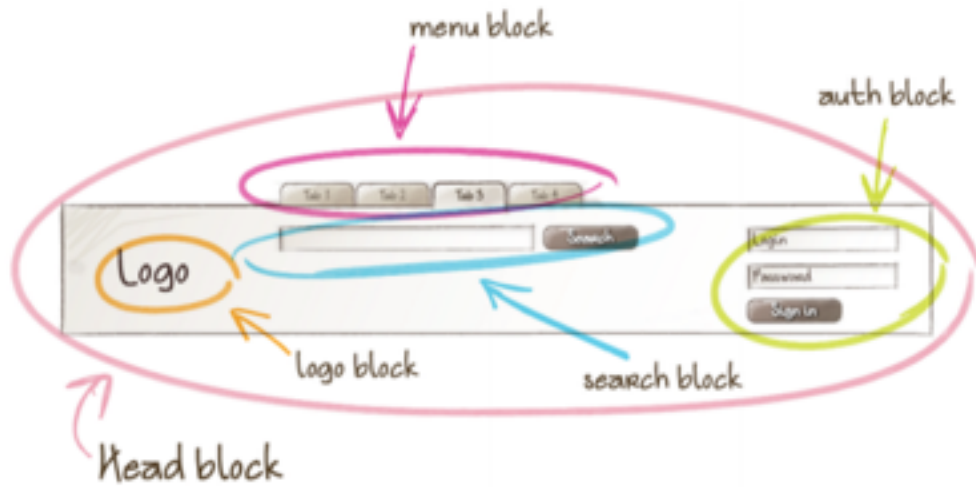
```
<ul class="list">
  <li class="list-item">0</li>
  <li class="list-item">1</li>
  <li class="list-item">
    условие отработало
  </li>
  <li class="list-item">3</li>
</ul>
```



Блок Элемент Модификатор

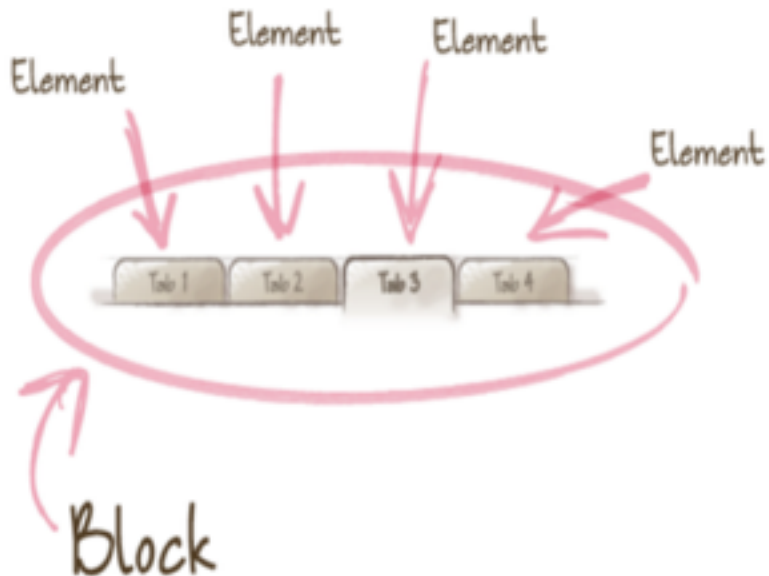


Блок



- Блок - это логический элемент интерфейса, внутри него находятся элементы (так же могут находится другие более мелкие блоки).
- Блок именуется односложно (если название из нескольких слов, то применяется "-" (**dash**) для разделения)

Элемент



- Элемент - то из чего состоит единица интерфейса (блок). Без блока не имеет смысла.
- Имя элемента складывается из имени блока и имени элемента. Таким образом:

block__element



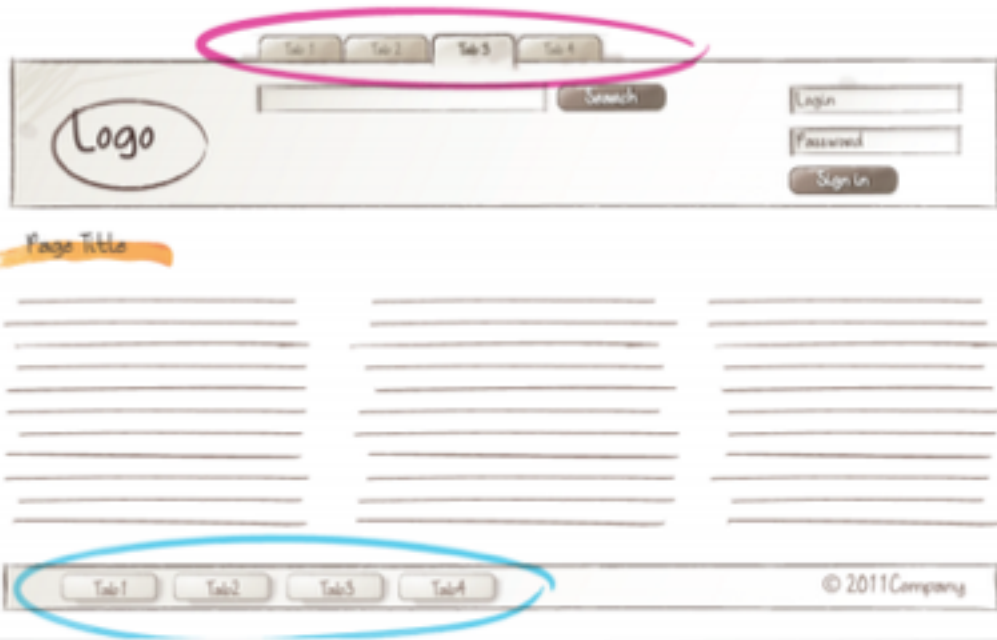
Модульность



- БЭМ помогает обеспечить модульность, и масштабируемость проекта
- Из-за особенной структуры имени классов, сводится к минимуму из пересечение, при разрастании проекта.
- Но без модульного подхода - не будет модульности!



Модификатор



- Модификатор - это класс, при “навешивании” которого изменяется отображение блока (или элемента)
- Имя модификатора формируется из имени блока, имени элемента (если применен к элементу) и имени модификатора, например:

block__element-name_modifier



Правила

- Верстка должна быть модульной (любой блок можно вырвать из одного места страницы и вставить в другое, так что бы он не “развалился” и не поломал ничего вокруг)
- На странице нет “голых” тегов! Все теги имеют свой класс!
- Нет селекторов по тегам (за исключением текстового изменяемого контента)
- Не использовать относительных селекторов (*, >, +)
- Зависимость классов допускается только от модификаторов блока!
- Название классов формируется по правилам синтаксиса BEM!

Время для ваших вопросов