Intro

На самом деле актуальные фреймворки мы рассматривать не собираемся, но будут рассмотрены незаслуженно обиженные вниманием коллег инструменты frontend разработки, а именно:

- Meteor 1.4
- Vue 2.x
- Polymer 2.0
- Elm 0.18

Agenda

Рассматривать инструменты будем следующим образом:

- Поставка - что за функционал доступен “из коробки”, архитектура проекта
- Конфигурация и сборка проекта
- Возможность использования на сервере
- Где и в каких случаях можно и оправдано его использовать
- Порог входа
- Как пишется SPA, сложности и подводные камни в реализации
- Conclusion

Cats For Everyone

Cart is empty

Amount range:

From 0

To 50000

Age:

From 0

To 20

Only sterile

No Yes

Size:

- [ ] Low
- [ ] Medium
- [ ] Large

Fur density:

Any

Color:

- [ ] Black

Mcfadden

5 years old
Black

$18416

Sullivan

6 years old
Red

$3383

Charlotte

1 years old
Gray

$32933

Kirsten

5 years old
White

$49535

Leslie

11 years old
White

$18071

Frazier

2 years old
Gray

$23921

Langley

7 years old
Red

$27972

Skinner

1 years old
Black

$30843

Recommended by Google and Rob Dodson

POLYMER

Web Components

- Создание собственных HTML тэгов, наследование существующих
- Написание модульного кода
- Vanilla JS/HTML/CSS
- Shadow DOM предоставляет инкапсуляцию CSS and DOM.

```
class HelloWorld extends HTMLElement {...}
window.customElements.define('hello-world', HelloWorld);

<hello-world></hello-world>
```

Polymer

- Polymer – библиотека, предоставляющая набор инструментов для быстрого и удобного создания Web компонент
- Небольшой размер
- Похожа на стандартные DOM элементы.
- Быстра в изучении, предоставляет большие возможности для разработки
- Большая коллекция уже созданных компонентов командой Polymer
- Быстро растущее сообщество разработчиков

Назначение фреймворка

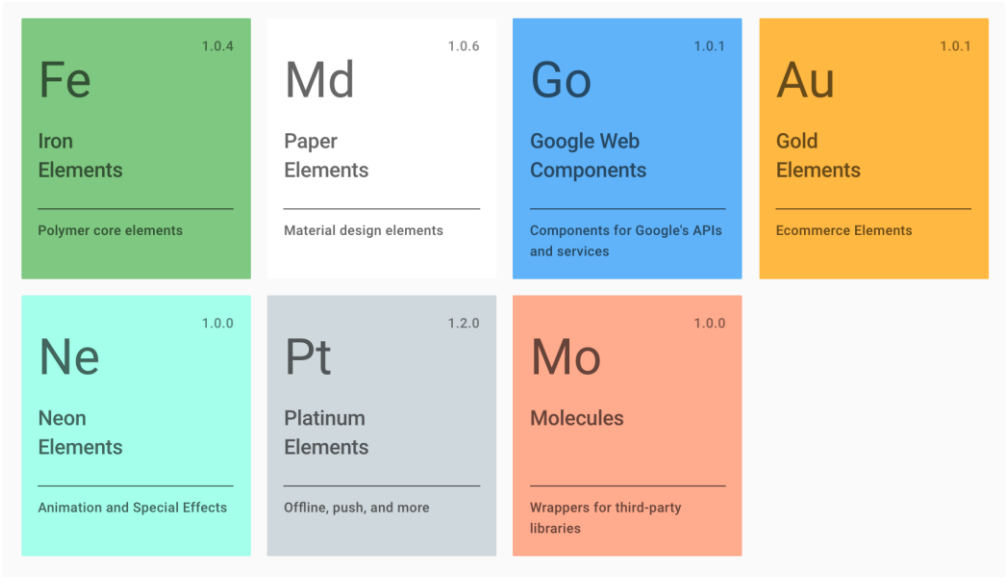
Использование на стороне сервера

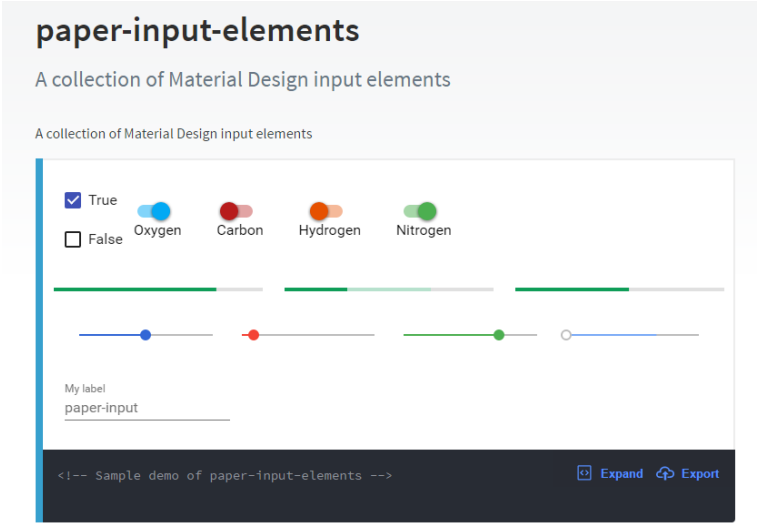
- Polymer – библиотека для использования на стороне клиента. Не используется на серверной стороне.

Где можно использовать

- Благодаря низкому порогу входа, использованию по макисумуму нативного API – приложения на Polymer легко поддерживать и добавлять дополнительный функционал.
- Подходит как для небольших SPA, так и для больших копоративных проектов
- Chrome 53, Opera 40, and Safari 10 поддерживают shadow DOM v1. Edge, Mozilla в планах на разработку
- Нет ограничения на структуру проекта

Большая коллекция готовых Polymer элементов





Зачем использовать Polymer 2?

Работает на последних стандартах v1 Shadow DOM and v1 Custom Elements API’s

Быстрый и удобный переход с Polymer 1

Максимальное использование нативного API.

Меньше шаблонного кода, “магии”

Зачем использовать Polymer 2?

```
class NativeCat extends HTMLElement {
    connectedCallback() {
        this.innerHTML = "<div>" +
            "<span style='background-color: aqua'></span>" +
            "</div>"
    }
    set name(value) {
        this.querySelector('span').textContent = value;
    }
}
customElements.define('native-cat', NativeCat);
```

```
<dom-module id="polymer-cat">
    <template>
        <div>
        <span style='background-color: aqua'>{{ name }}</span>
        </div>
    </template>
</dom-module>
<script>
    class PolymerCat extends Polymer.Element {
        static get is() { return 'polymer-cat'; }

        static get properties() {
            return {
                name: String
            }
        }
    }
    customElements.define(PolymerCat.is, PolymerCat);
</script>
```

Зачем использовать Polymer 2?

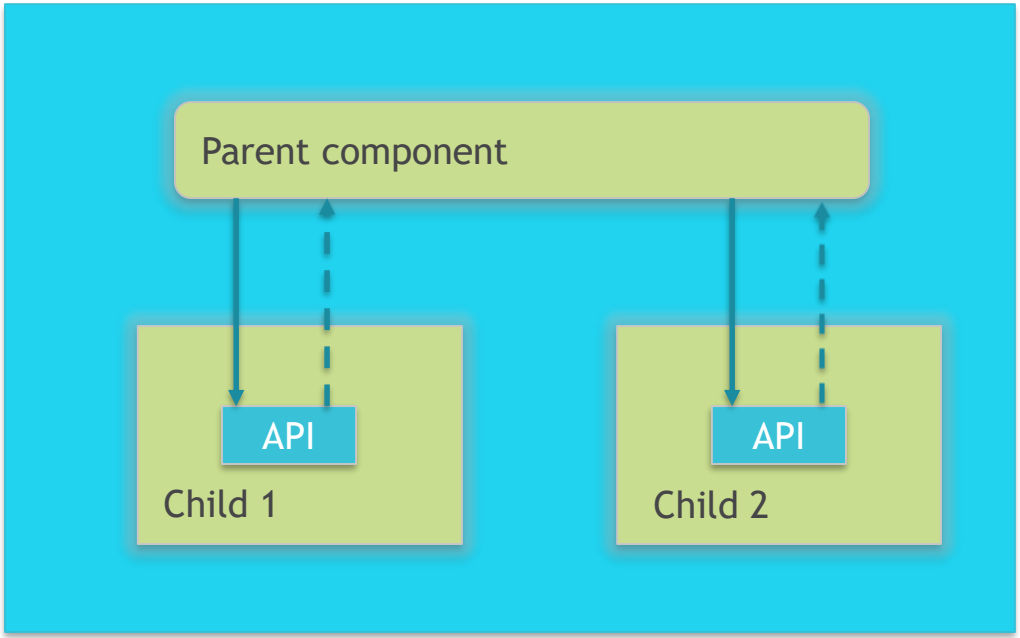
```
▼<search-page name="search" class="iron-selected">
  ▼#shadow-root (open)
    ▶<style>…</style>
     <iron-ajax id="getCatsRequest" url="/cats" handle-as="json" content-
              </iron-ajax>
    ▼<div class="pageContent row">
      ▶<div class="col filter-container">…</div>
      ▼<div class="col search-result-container">
        ▼<thumbnail-view id="thumbnailView">
          ▼#shadow-root (open)
            ▶<style>…</style>
            ▼<div class="viewContainer">
              ▶<thumbnail-info>…</thumbnail-info>
              ▶<thumbnail-info>…</thumbnail-info>
              ▶<thumbnail-info>…</thumbnail-info>
              ▼<thumbnail-info> == $0
                ▼#shadow-root (open)
                  ▶<style>…</style>
                  ▼<div class="card layout horizontal hoverable grey lighten-4">
                    ▶<div class="card-image waves-effect waves-block waves-light">…</div>
                    ▼<div class="card-stacked">
                      ▶<div class="card-content" style="position: relative;">…</div>
                      ▶<div class="card-action">…</div>
                      </div>
                    </div>
                </thumbnail-info>
              ▶<thumbnail-info>…</thumbnail-info>
              ▶<thumbnail-info>…</thumbnail-info>
              ▶<thumbnail-info>…</thumbnail-info>
```

Mitchell

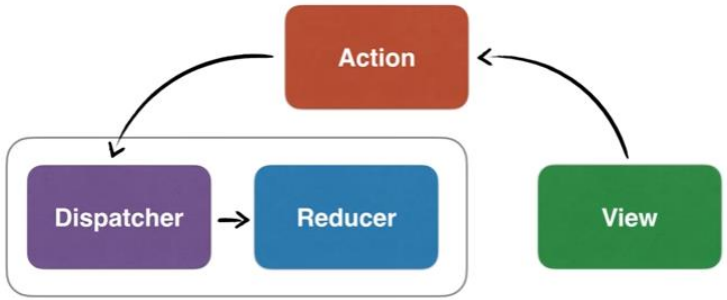
9 year old
gray, black

$36342

Взаимодействие компонент



Взаимодействие компонент



Redux

https://tur-nr.github.io/polymer-redux

Взаимодействие компонент

- Data System основана на Paths, а не на объектах
- Только изменения, регистрируемые в Paths приводят к обновлению bindings и вызова observers

```
// Не вызовет observer, data binding не сработает
this.person.name = 'Sergey';
this.people.push({ name: 'Sergey'});

// Изменения подхватятся Polymer
this.set('person.name', 'Sergey');
// или
// Оповестить Polymer что значение поля изменилось
this.person.name = 'Sergey';
this.notifyPath('person.name');

this.push('people', { name: 'Sergey'});
```

- MutableData mixin – отключает механизм dirty-checking

```
// Произойдет оповещение изменения property
this.property.subproperty = 'new value!';
this.notifyPath('property');
```

Thumbnail компонент

```
class ThumbnailInfo extends Polymer.Element {
    static get is() {
        return 'thumbnail-info';
    }
    static get properties() {
        return {
            item: Object,
            disabled: {
                type: Boolean,
                reflectToAttribute: true
            }
        }
    }
    ...
    addToCart() {
        this.dispatchEvent(new CustomEvent('add-to-cart', { bubbles: true,
            composed: true, detail: { item: this.item } }));
    }
    _getColors(item) {
        return item.furColor.join(', ');
    }
}
customElements.define(ThumbnailInfo.is, ThumbnailInfo);
```

thumbnail.js

```
<dom-module id="thumbnail-info">
    <template>
        <style include="shared-styles iron-flex iron-flex-alignment"></style>
        <style>
            ...
        </style>
        <div class="card layout horizontal hoverable grey lighten-4">
                ...
                <div class="card-stacked">
                    <div class="card-content" style="position: relative;">
                        <a href="#/detail" class="card-title">{{ item.name }}</a>
                        <p>{{ item.age }} year old</p>
                        <p>{{ _getColors(item) }}</p>
                    </div>
                    <div class="card-action">
                        <span class="orange-text">${{ item.price }}</span>
                        <paper-button class="addToCartBtn" disabled$="{{ disabled }}" on-click="addToCart">
                            <iron-icon icon="add"></iron-icon>
                        </paper-button>
                    </div>
                </div>
            </div>
    </template>
    <script src="thumbnail.js"></script>
</dom-module>
```

thumbnail.html

Роутинг

Example:

```
<app-location route="{{ route }}" use-hash-as-path></app-location>
<app-route
           route="{{ route }}"
           pattern="/:page"
           data="{{ routeData }}"
           tail="{{ subroute }}">
</app-route>
<app-route
           route="{{ subroute }}"
           pattern="/:id"
           data="{{ subrouteData }}">
</app-route>
```

```
<app-location route="{{ route }}" use-hash-as-path></app-location>
<app-route
          route="{{ route }}"
          pattern="/:page"
          data="{{ routeData }}">
</app-route>
<iron-pages class="content" selected="{{ routeData.page }}" attr-for-selected="name" fallback-selection="search">
  <search-page name="search" selected-items="{{ selectedItems }}"></search-page>
  <cart-page name="cart" selected-items="{{ selectedItems }}"></cart-page>
  <detail-page name="detail" ></detail-page>
</iron-pages>
```

mainPage.html

Отладка и информативность ошибок

ОТЛАДКА

- Не требует сторонних инструментов

ИНФОРМАТИВНОСТЬ ОШИБОК

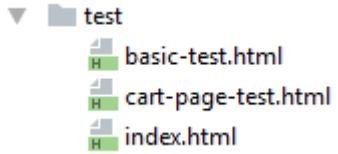
- Легко отловить логические и синтаксические ошибки. Сообщения об ошибках интуитивно понятные
- Тяжело отловить ошибки, связанные с жизненным циклом компоненты и с data binding.

Тестирование

Web Component Tester

1. Mocha
2. Chai
3. Sinon
4. Selenium
5. Accessibility Developer Tools

Тестирование



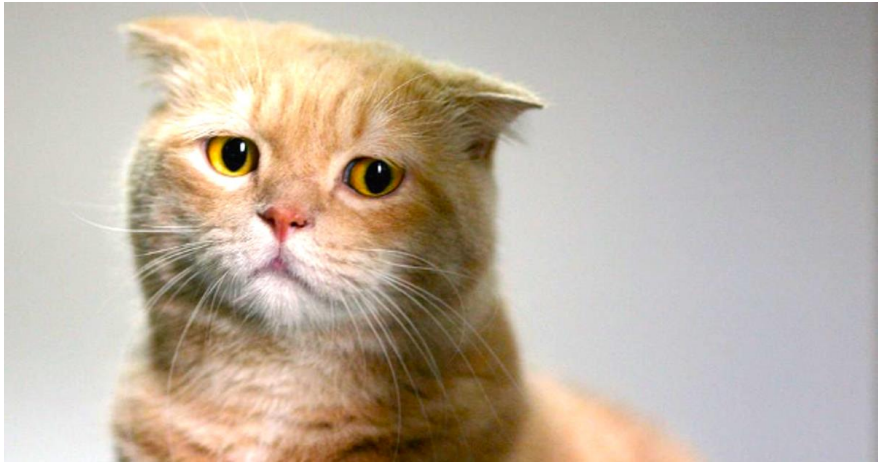

```
<html>
...

<test-fixture id="cart-page-test">
    <template>
        <cart-page></cart-page>
    </template>
</test-fixture>

<script>
   ...
</script>

</html>
```

```
<script>
    suite('<cart-page>', function () {
        var cartPage;
        setup(function () {
            cartPage = fixture('cart-page-test');
            cartPage.items = [
                {
                    "id": "58b66b7ef5f505c72767b4c6",
                    "name": "Shirley",
                },
                {
                    "id": "58b6ab7ef5f505c72767b4c4",
                    "isInCart": true
                }];
        });
        test('check checkout functionality', function () {
            var checkoutButton = cartPage.shadowRoot.querySelector('#checkout-button');
            checkoutButton.click();
            assert.equal(cartPage.items.filter((item) => item.isInCart).length, 0);
        });
    });
</script>
```

cart-page-test.html

Возникшие проблемы

1. На момент написания презентации – библиотека не в релизе
2. Пустой “stack overflow”
3. Нестандартные подходы к data binding, routing



Дополнительные источники (Polymer)

1. https://www.polymer-project.org/2.0/docs/about_20
2. https://developers.google.com/web/fundamentals/getting-started/primers/customelements
3. https://www.webcomponents.org/collection/Polymer/elements
4. Rob Dodson’s “Polycasts”: https://www.youtube.com/watch?v=PahsgJn0sgU
5. Thinking in Polymer (The Polymer Summit 2015): https://www.youtube.com/watch?v=ZDjiUmx51y8
6. Polymer 2016 summit: https://www.youtube.com/watch?v=iJ9hS54BRag

Fastest way to build

METEOR

Meteor

- **Изоморфный код**
- **Постоянный обмен данными** (Distributed Data Protocol)
- **Обширная экосистема**: богатая документация и обучающие материалы, большая база плагинов, комьюнити, в т.ч. собственное, Stackoverflow
- **Full-stack реактивность**, позволяет UI работать с реактивными состояниями легко и с минимальными затратами на разработку

Базовые возможности Meteor

ПОСТАВКА

- Node.js, MongoDB, Cordova, ES6, Blaze (Handlebars)
- Свой собственный менеджер пакетов
- Преднастроенный сборщик проекта

АРХИТЕКТУРА

- Абстракция для работы с данными через коллекции (одна модель для сервера и клиента)
- DDP (EJSON) - работает с использованием WebSockets
- Строгая структура проекта и порядок загрузки скриптов
- Можно использовать с Angular или React

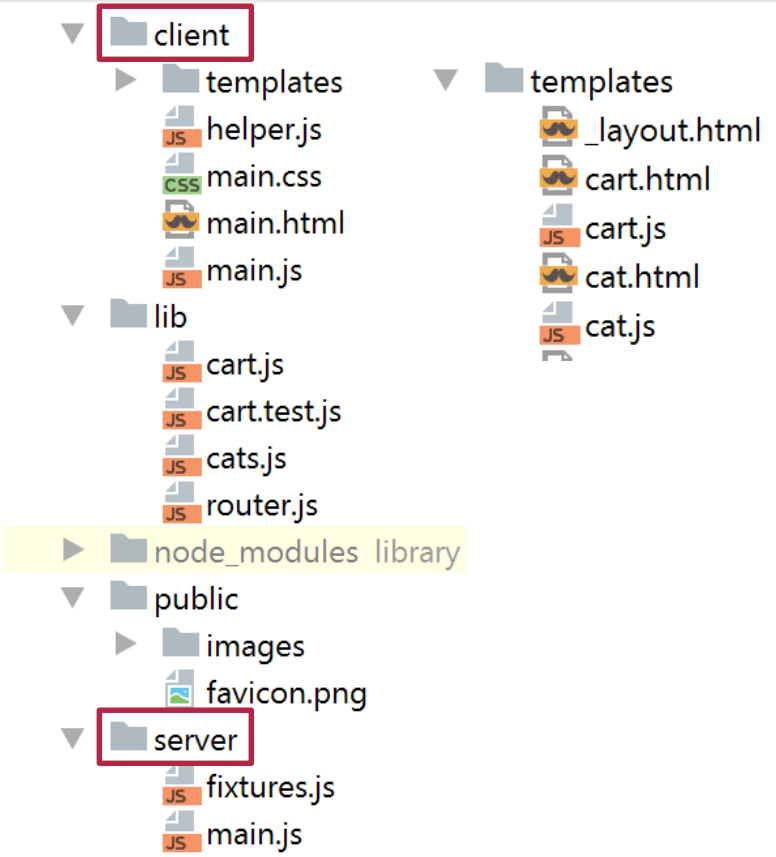
Пример организации приложения

1. Разделение клиентского и серверного кода в структуре папок и условиями вида

```
if(Meteor.isServer) {
  //code for server execution
}
if(Meteor.isClient) {
  //code for client execution
}
```

2. Шаблоны Blaze

```
<template name="Layout">
  {{> Nav}}
  {{> yield}}
</template>
```



Пример компонента

helpers.js

```
Template.registerHelper('FormatMoney', (number) => {
  return '$' + number.toLocaleString();
});
```

cat.html

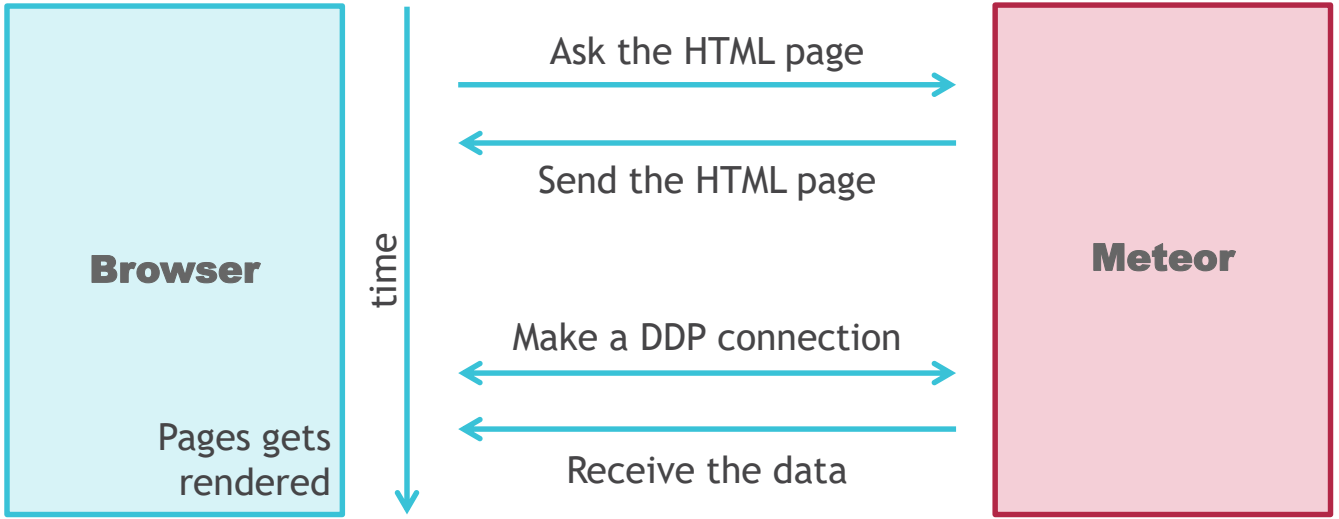
```
<template name="Cat">
 ...
 <p class="orange-text">{{FormatMoney price}}</p>
 {{#if isInCart}}
     <button id="add">Add to cart</button>
 {{else}}
     <a href="/cart">Go to cart</a>
 {{/if}}
 ...
</template>
```

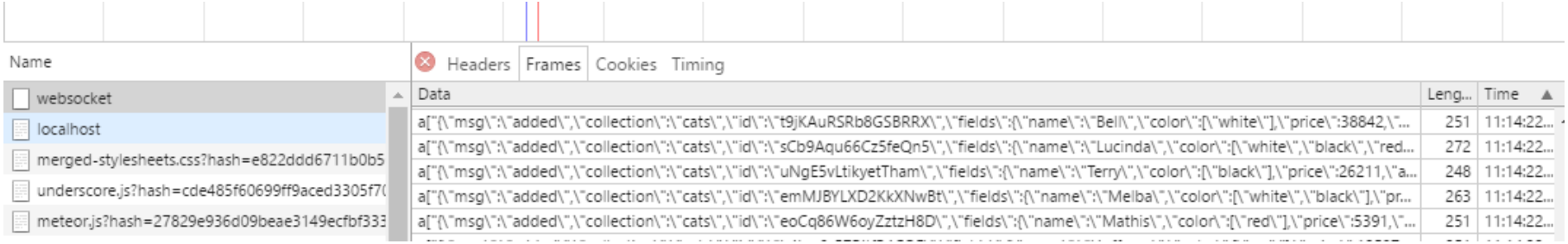
from template data

cat.js

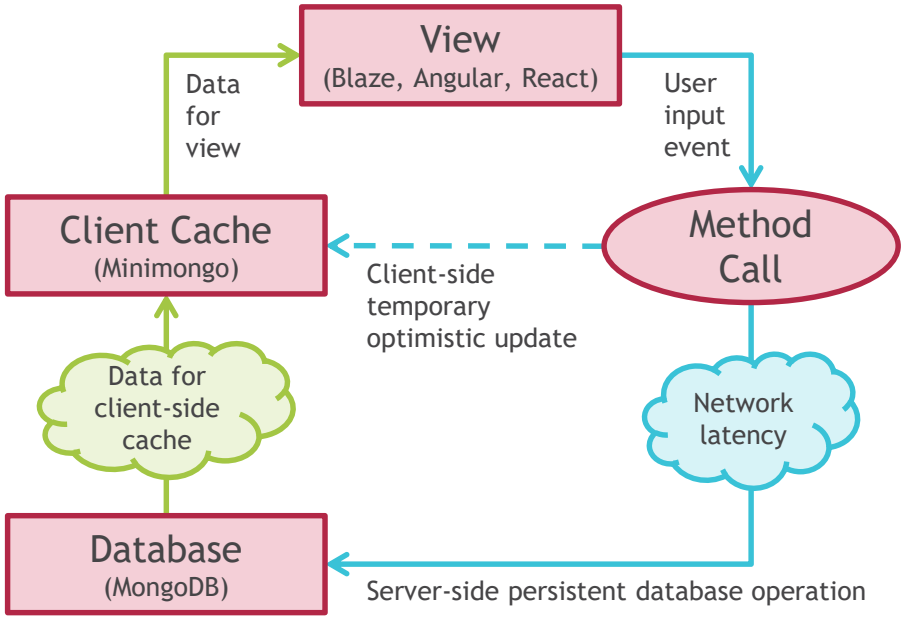
```
import { Template } from 'meteor/templating';
import { Cart } from '../../lib/cart.js';

Template.Cat.helpers({
  isInCart() {
    const instance = Template.instance();
    return !Cart.findOne({id: instance.data._id});
  },
});

Template.Cat.events({
  'click #add'(event, instance) {
    event.preventDefault();
    Cart.insert({id: instance.data._id});
  },
});
```

DDP


Browser
Pages gets rendered
time
Ask the HTML page
Send the HTML page
Make a DDP connection
Receive the data
Meteor


Name
websocket
localhost
merged-stylesheets.css?hash=e822ddd6711b0b5
underscore.js?hash=cde485f60699ff9aced3305f7(
meteor.js?hash=27829e936d09beae3149ecfbf333
Headers
Frames
Cookies
Timing
Data
Leng...
Time
a["{\"msg\":\"added\",\"collection\":\"cats\",\"id\":\"t9jKAuRSRb8GSBRRX\",\"fields\":{\"name\":\"Bell\",\"color\":[\"white\"],\"price\":38842,\"... 251 11:14:22...
a["{\"msg\":\"added\",\"collection\":\"cats\",\"id\":\"sCb9Aqu66Cz5feQn5\",\"fields\":{\"name\":\"Lucinda\",\"color\":[\"white\",\"black\",\"red... 272 11:14:22...
a["{\"msg\":\"added\",\"collection\":\"cats\",\"id\":\"uNgE5vLtikyetTham\",\"fields\":{\"name\":\"Terry\",\"color\":[\"black\"],\"price\":26211,\"a... 248 11:14:22...
a["{\"msg\":\"added\",\"collection\":\"cats\",\"id\":\"emMJBYLXD2KkXNwBt\",\"fields\":{\"name\":\"Melba\",\"color\":[\"white\",\"black\"],\"pr... 263 11:14:22...
a["{\"msg\":\"added\",\"collection\":\"cats\",\"id\":\"eoCq86W6oyZztzH8D\",\"fields\":{\"name\":\"Mathis\",\"color\":[\"red\"],\"price\":5391,\"... 251 11:14:22...

Data flow



Создание коллекции

```
Cart = new Mongo.Collection('cart');
```

Публикация и подписка

```
Meteor.publish('cats', function() {
  return Cats.find();
});

Meteor.subscribe('cats');
```

Удалённый вызов процедур (RPC)

```
Meteor.methods({
  'cart.clear'() {
    return Cart.remove({});
  },
});

Meteor.call('cart.clear');
```

Реактивность на клиенте

filters.js

```
import { Template } from 'meteor/templating';
import { ReactiveDict } from 'meteor/reactive-dict';

export const state = new ReactiveDict();

Template.Filters.onCreated(function() {
  state.clear();
});

Template.Filters.events({
   //....
  'input #agefrom'(event, instance) {
    state.set('ageFrom', event.target.value);
  }
  //....
});
```

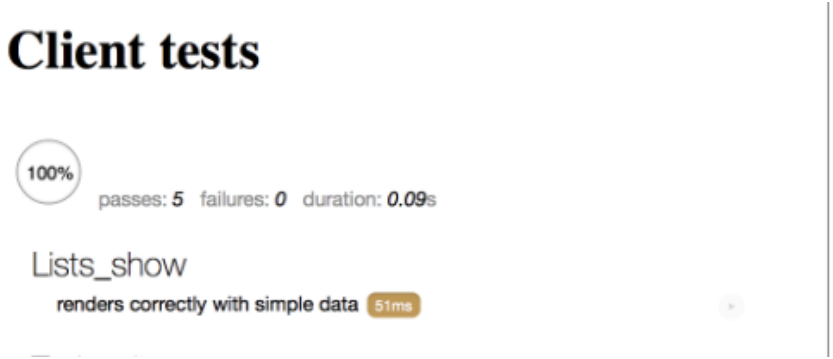
home.js

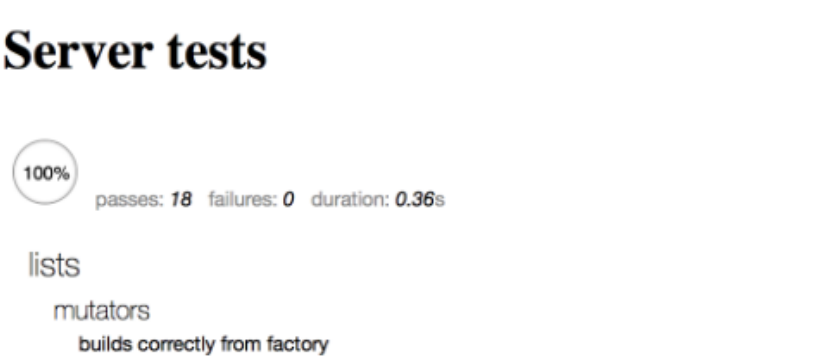
```
import { Template } from 'meteor/templating';
import { Cats } from '../../lib/cats.js';

import { state } from './filters.js';

Template.Home.helpers({
  cats() {
    if (!state) return [];
    let filters = state.all();
    let params = {};
    //......
    if (filters.colors && filters.colors.length > 0) {
      params.color = { $all: filters.colors };
    }
    //......
    return Cats.find(params).fetch();
  },
});
```

Тестирование

1. Mocha, Chai
2. Плагины для мок-данных, random и прочее
3. Kadira, Meteor DevTools (Chrome plugin) для профилирования DDP соединения

```
describe('Cart', () => {
  describe('methods', () => {
    beforeEach(() => {
      Cart.remove({});
      Cart.insert({ id: "123456" });
    });

    it('can clear the cart', () => {
      Meteor.server.method_handlers['cart.clear']();
      assert.equal(Cart.find().count(), 0);
    });
  });
});
```





Применение и подводные камни

1. Удобство деплоя приложения под разные платформы
2. Простой код, отличная документация и примеры использования
3. Быстрый старт

1. Высокий порог входа из-за большого количества технологий
2. Мало Meteor-разработчиков
3. Неполная поддержка Windows

Простой, быстрый, крутой

VUE

Vue

- **Доступный.** Низкий порог вхождения, интуитивно понятный синтаксис
- **Разносторонний.** Простое минималистичное ядро и постепенная интеграция в приложения любого масштаба.
- **Производительный.** Всего 18 Кб (min+gzip), быстрый виртуальный DOM, оптимизация без сложностей
- **Быстро развивающийся.** Экосистема стремительно развивается. Уже существуют:
  - Weex для Android и iOs (Alibaba Group)
  - Nuxt JS для SSR
  - И многое другое...

Инициализация приложения

- Инициализируем приложение, монтируя главный компонент (App.vue) в корневой index.html

```
<!DOCTYPE html>
<html lang="en">
  <head> ... </head>
  <body>
    <div id="app"></div>
    <script src="/dist/build.js"></script>
  </body>
</html>
```

index.html

```
import Vue from 'vue'
import App from './pages/App.vue'
import store from './store'
import router from './router'

new Vue({
  el: '#app',
  store,
  router,
  render: h => h(App)
});
```

main.js

Роутинг

- Роутинг простой и интуитивно понятный

```
import Cats from './pages/Cats.vue'
import Details from './pages/Details.vue'
import CartPage from './pages/Cart.vue'
import Vue from 'vue'
import VueRouter from 'vue-router'

Vue.use(VueRouter)
var router = new VueRouter({
  routes: [
    { path: '/', component: Cats },
    { path: '', redirect: '/' },
    { path: '/details/:id', component: Details,
      props: true },
    {path: '/cart', component: CartPage}
  ]
})

export default router
```

router.js

Глобальное состояние

- Состояние приложения хранится в глобальном Store, для создания которого используем Vuex

```
import Vue from 'vue'
import Vuex from 'vuex'
import * as actions from './actions'
import * as getters from './getters'
import cart from './modules/cart'
import cats from './modules/cats'

Vue.use(Vuex)

export default new Vuex.Store({
  actions,
  getters,
  modules: {
    cart,
    cats
  }
})
```

store/index.js

```
import shop from '../../api/shop'
import * as types from '../mutation-types'
import helper from '../helperFunctions'

// initial state
const state = {
  all: [],
  cats: []
};

// actions
const actions = {
  getAllCats ({ commit }) {
    shop.getCats().then(response => {
      let cats = response.body
      commit(types.RECEIVE_CATS, { cats })
    })
  }
};

// getters
const getters = {
  allCats: state => state.cats
};

// mutations
const mutations = {
  [types.RECEIVE_CATS] (state, { cats }) {
    state.all = cats;
    state.cats = cats
  },

  [types.ADD_TO_CART] (state, { id }) {
    state.cats = helper
      .changeState(state.cats, id, false)
    }

export default { state, getters, actions,
mutations }
```

store/modules/cats.js

Структура компонента

- Структура состоит из template, script и style
- Может включать вложенные компоненты
- Экспортируется объект конфигурации, который содержит методы и параметры объекта.
- Параметры могут быть статическими, вычисляемыми и переданными через props.
- Входные параметры обеспечивают однонаправленный поток данных от родительского компонента к потомкам.
- Вызывает хуки жизненного цикла.
- Компилируется с помощью vue-loader
- Возможно быстрое создание проекта с помощью vue-cli

pages/cats.vue

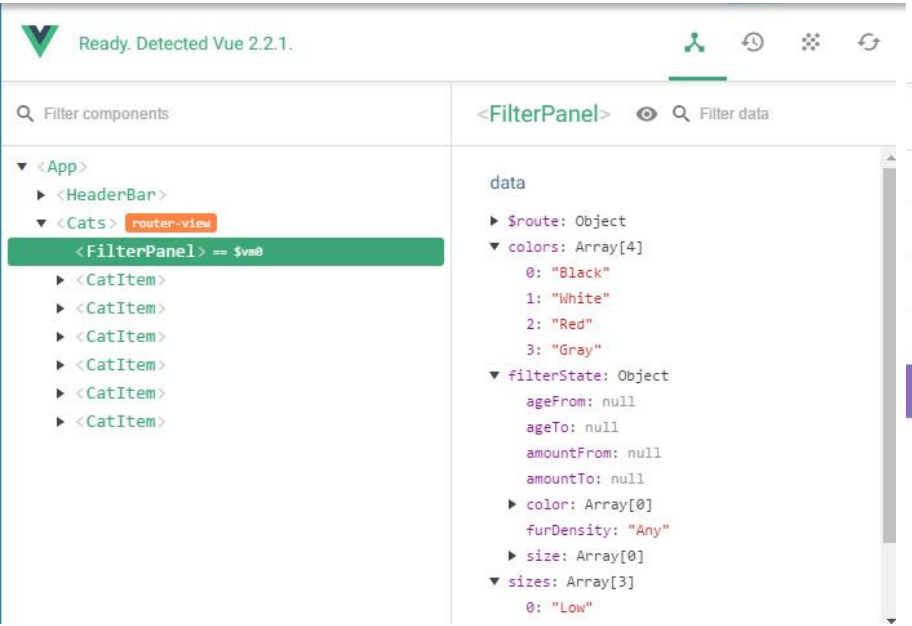
```
<template>
  <section>
    <filter-panel :class="filterPanelClass"></filter-panel>
    <div class="col s9">
      <div class="col s6" v-for="cat in cats">
        <cat-item :id="cat.id"></cat-item>
      </div>
    </div>
  </section>
</template>

<script>
  import filterPanel from '../components/filterPanel.vue'
  import catItem from '../components/catItem.vue'

  export default {
    data: {
      filterPanelClass: 'active'
    },
    computed: {
      cats: this.$store.getters.allCats
    },
    components: {
      filterPanel,
      catItem
    }
  }
</script>

<style lang="scss" src="cats.scss" scoped></style>
```

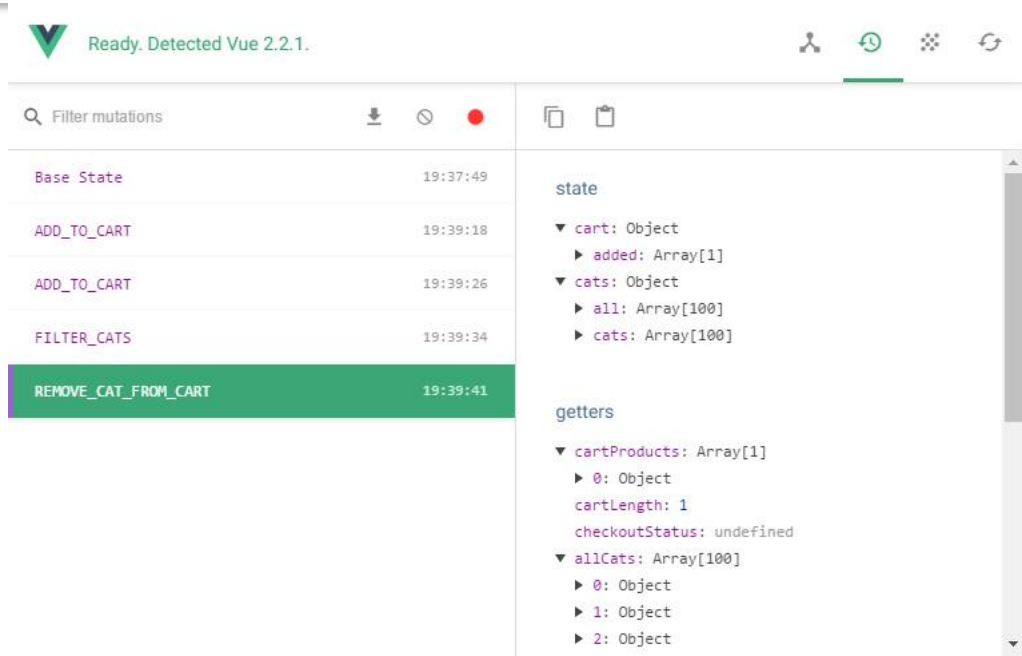
Создание приложения. Тестирование

- Тестирование любыми инструментами

```
import cart from '../src/store/modules/cart'
import * as types from '../src/store/mutation-types'

const addToCart = cart.mutations[types.ADD_TO_CART];

describe('mutations', () => {
  it('ADDTOCART', () => {
    const state = { added: [] };
    addToCart(state, { id: 123 });
    expect(state.added.length).toEqual(1);
    expect(state.added[0].id).toEqual(123);
  })
});
```

Создание приложения. Отладка





- Специальный инструмент отладки
- Возможность импорта-экспорта состояния

A delightful language for reliable webapps with pain and types

Elm

- **Статическая типизация**
- **Нет Runtime Exceptions.** В Elm есть только ошибки времени компиляции. В Elm нет exception system
- **Неизменяемость.** Все значения в Elm являются неизменяемыми (immutable)
- **Взаимодействие с JavaScript.** Код написаный на Elm может взаимодействовать с кодом на чистом js.
- **Абстракции над HTML/CSS**

Поставка

ПОСТАВКА

- Свой собственный менеджер пакетов
- Готовые решения для routing, websockets, geolocation
- Есть плагины для grunt/gulp
- Хорошая поддержка IDE (Brackets, IntelliJ *, Atom, Sublime etc)

СТРУКТУРА

- Elm не предъявляет строгих требований к структуре проекта

Функционал

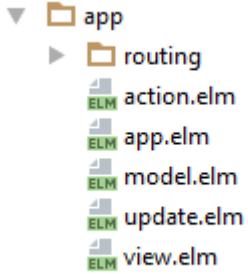
ИСПОЛЬЗОВАНИЕ НА СТОРОНЕ СЕРВЕРА

- Есть несколько реализаций Server Side Rendering

ГДЕ МОЖНО ИСПОЛЬЗОВАТЬ

- Использование ограничивается frontend приложениями.

Структура



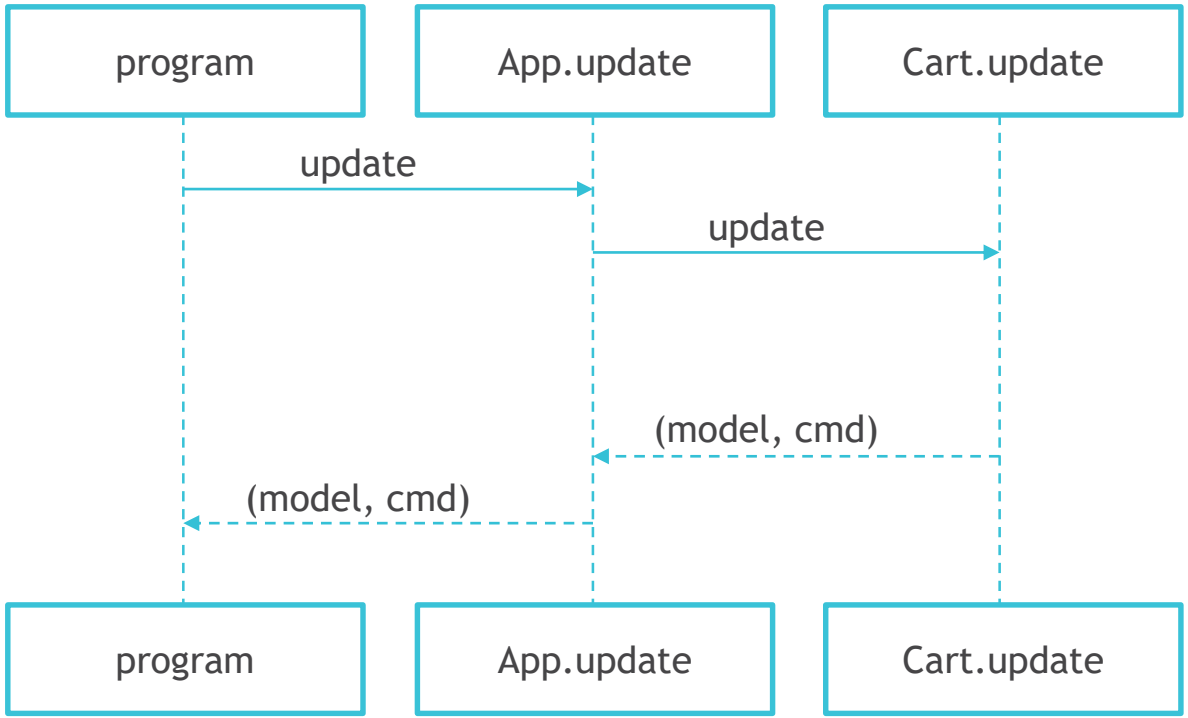
app/app.elm

```
main =
    program
        { init = init
        , view = view
        , update = update
        , subscriptions = (always Sub.none)
        }
```

app/model.elm

```
type alias Model =
    { filter: Filter.Model.Filter,
      adverts: Advert.Model.Adverts,
      cart: Cart.Model.Cart }

init : (Model, Cmd Action)
init =
  let
    (filter, filterCmd) = Filter.Model.init
    (adverts, advertsCmd) = Advert.Model.init
    (cart, cartCmd) = Cart.Model.init
  in
    (Model filter adverts cart,
        Cmd.batch [
            Cmd.map FilterAction filterCmd,
            Cmd.map AdvertsAction advertsCmd,
            Cmd.map CartAction cartCmd])
```

Data flow


program
App.update
Cart.update
update
update
(model, cmd)
(model, cmd)
program
App.update
Cart.update

Data flow (Update)

app/update.elm

```
update: Action -> Model -> (Model, Cmd Action)
update action model =
    case action of
        NoOp ->
            (model, Cmd.none)
        CartAction cartAction ->
            let (newCart, cmd) = Cart.Update.update cartAction model.cart
                newModel = { model | cart = newCart }
            in (newModel, Cmd.map CartAction cmd)
        ...
```

cart/update.elm

```
update: Action -> Cart -> (Cart, Cmd Action)
update action model =
    case action of
        NoOp ->
            (model, Cmd.none)
        Add advert -> ({ model | adverts = advert :: model.adverts }, Cmd.none)
        ...
```

Data flow (View)

app/view.elm

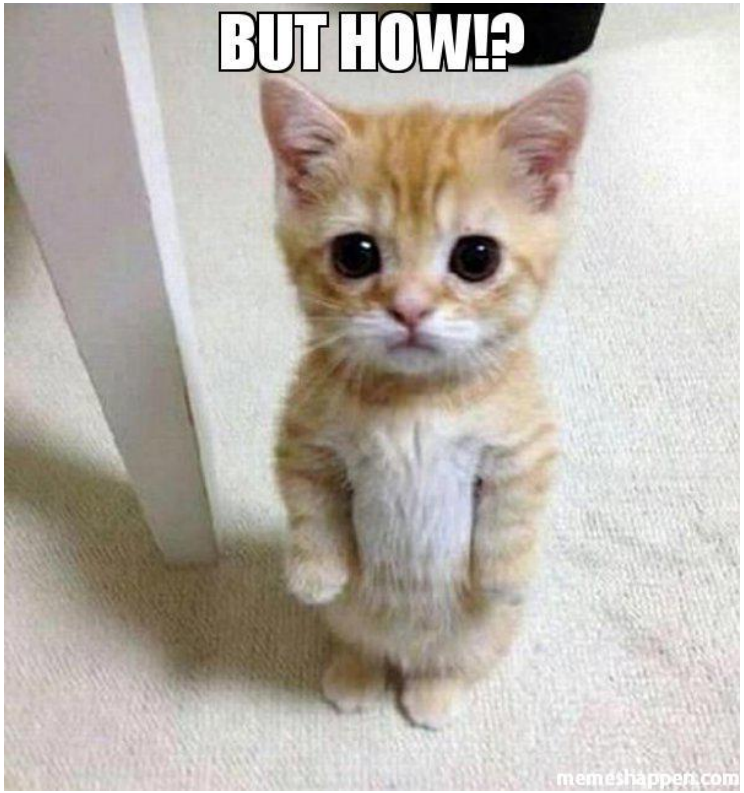
```
view: Model -> Html Action
view model =
    div [] [
        ...
        toolbar model
        ...
    ]
```

```
toolbar: Model -> Html Action
toolbar model =
        nav [ class "light-blue lighten-1", attribute "role" "navigation" ]
            [ div [ class "nav-wrapper container" ]
                [ a [ class "brand-logo", href "#",
                      id "logo-container", onClick CartRoute ]
                    [ text "Cats For Everyone" ]
                , ul [ class "right hide-on-med-and-down" ]
                    [ li []
                        [ a [ class "waves-effect", href "#" ]
                            [ i [ class "large material-icons left" ]
                                [ text "shopping_cart" ]
                            , text <| cartText model
                            ]
                        ]
                    ]
                ]
            ]
```

```
cartText: Model -> String
cartText model =
    if List.isEmpty model.cart.adverts then
        "Cart is empty"
    else
        model.cart.adverts
            |> List.length
            |> toString
```

Ошибки

- В Elm приложениях не может быть рантайм ошибок*
- Только ошибки времени компиляции



Ошибки

app/cart/update.elm

```
Cannot find variable `List.nap`.

16|   List.nap (\advert -> advert.id /= id) model.adverts }, Cmd.none)
      ^^^^^^^^
`List` does not expose `nap`. Maybe you want one of the following?
    List.map
    List.any
    List.map2
    List.map3
```

app/filter/model.elm

```
The 1st argument to function `Filter` is causing a mismatch.

16|      Filter "1.20" 0 0 0 "" "" "",
                ^^^^^^
Function `Filter` is expecting the 1st argument to be:
    Float
But it is:
    String
```

Ошибки

app/filter/update.elm

```elm
update: Action -> Filter -> (Filter, Cmd Action)
update action model =
    case action of
        NoOp ->
            (model, Cmd.none)
        PriceFrom price ->
            ({model | priceFrom = price}, Cmd.none)
        ...
--        Apply ->
--            (model, Cmd.none)


type Action =
    NoOp
    | PriceFrom Float
    ...
    | Apply
```

```
This `case` does not have branches for all possibilities.

 8|>    case action of
 9|>        NoOp ->
10|>            (model, Cmd.none)
11|>        PriceFrom price ->
12|>            ({model | priceFrom = price}, Cmd.none)
...
You need to account for the following values:

Filter.Action.Apply

Add a branch to cover this pattern!
```

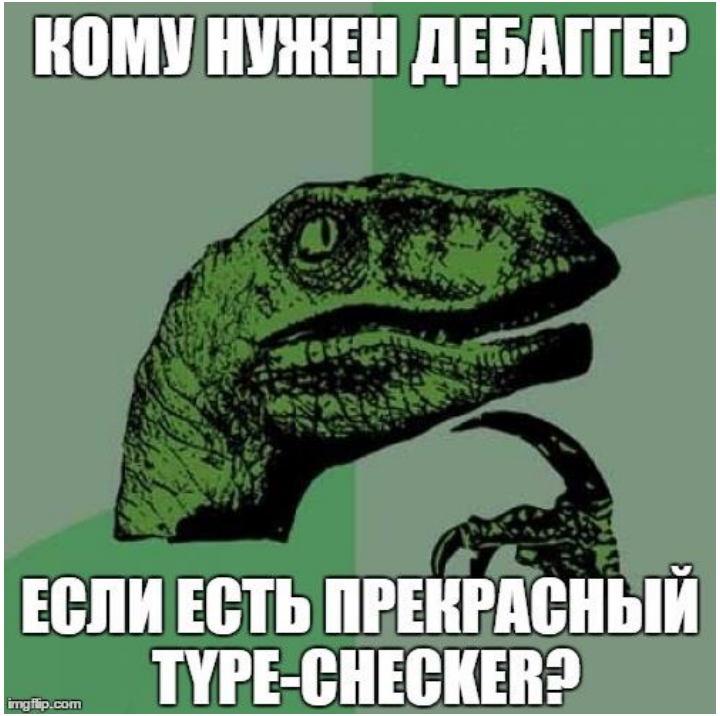
Ошибки

app/advert/model.elm

```
type alias Advert =
    { id: Int,
      title: String,
      description: String,
      price: Maybe Float }
```

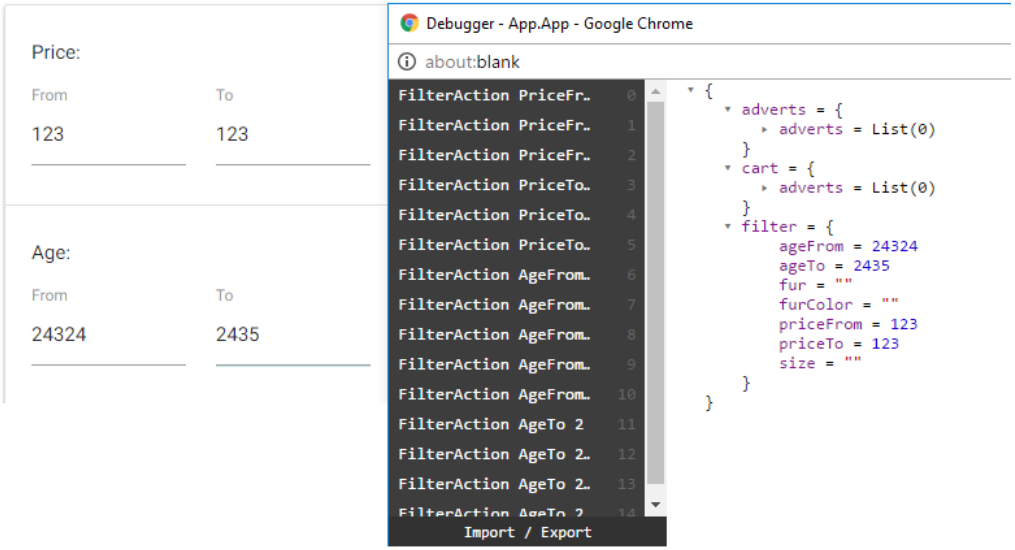
```
advert: Advert
advert = Advert 1 "Cat Title" "Cat Description" (Just 120.00)
```

Отладка


КОМУ НУЖЕН ДЕБАГГЕР
ЕСЛИ ЕСТЬ ПРЕКРАСНЫЙ TYPE-CHECKER?
imgflip.com

Отладка

- Включается флагом --debug в elm-make
- Записывает все действия пользователя
- Time traveling
- Есть возможность импорта/экспорта результатов



Тестирование

- Для тестирования в Elm есть пакет elm-test
- elm-test поддерживает "watch mode". В данном режиме при изменении исходных файлов, ваши тесты будут автоматически запускаться
- Структура стандартна для большинства test фреймворков
- Для BDD тестирования существует надстройка над elm-test с jasmine-like синтаксисом

Тестирование

```
advert: Advert
advert = Advert 1 "Test Title" "Test Description" (Just 120.00)

cartWithAdvert: Cart
cartWithAdvert =
    { adverts = [advert] }

all : Test
all =
    describe "Tests"
        [ describe "Cart Test"
            [ test "Add (cart)" <|
                \() ->
                    cart
                        |> update (Add advert)
                        |> Tuple.first
                        |> Expect.equal cartWithAdvert,
              test "Delete (cart)" <|
                \() ->
                    cartWithAdvert
                        |> update (Delete 1)
                        |> Tuple.first
                        |> Expect.equal cart
            ]
        ]
```

Итог

	Polymer	Meteor	Vue	Elm
Порог входа	низкий	средний	низкий	высокий
Работа на сервере	-	полная	SSR	-
Структура проекта	-	строгая	-	-
Data Flow	гибкий	гибкий	однонаправл.	однонаправл.
Отладка	-	Профили-рование DDP	State debugger	Strict state debugger