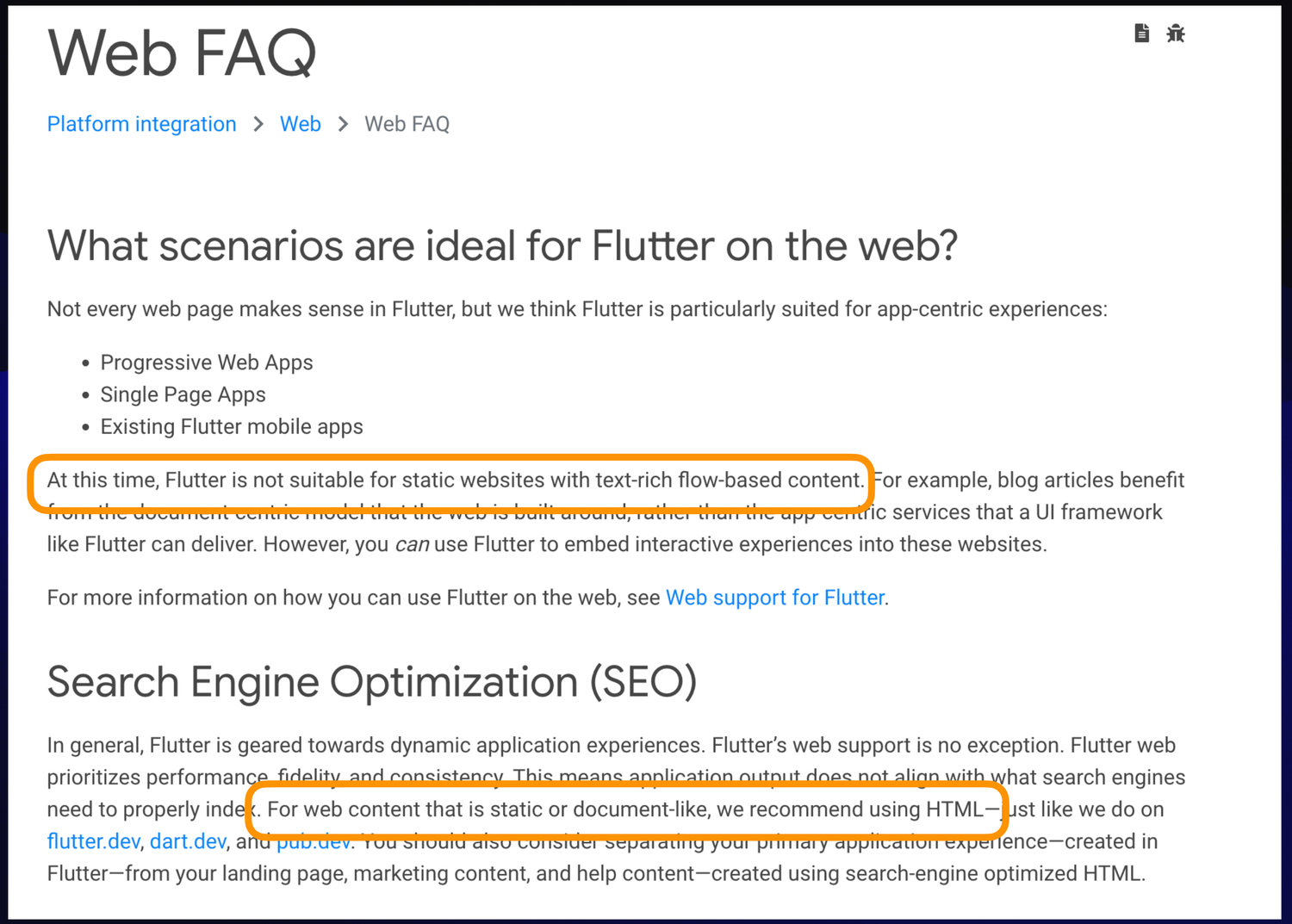
Introducing Jaspr

Darts first full stack web framework

Kilian Schulte
@schultek

Building websites with Dart?

Flutter Web?



Web FAQ

Platform integration > Web > Web FAQ

What scenarios are ideal for Flutter on the web?

Not every web page makes sense in Flutter, but we think Flutter is particularly suited for app-centric experiences:

- Progressive Web Apps
- Single Page Apps
- Existing Flutter mobile apps

At this time, Flutter is not suitable for static websites with text-rich flow-based content. For example, blog articles benefit from the document-centric model that the web is built around, rather than the app-centric services that a UI framework like Flutter can deliver. However, you *can* use Flutter to embed interactive experiences into these websites.

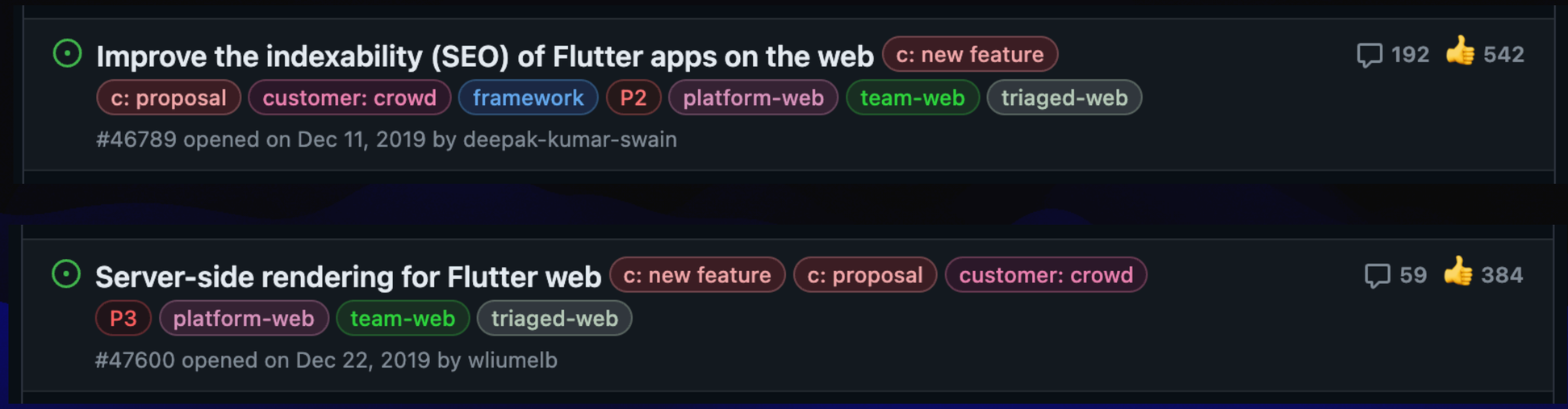
For more information on how you can use Flutter on the web, see Web support for Flutter.

Search Engine Optimization (SEO)

In general, Flutter is geared towards dynamic application experiences. Flutter's web support is no exception. Flutter web prioritizes performance, fidelity, and consistency. This means application output does not align with what search engines need to properly index. For web content that is static or document-like, we recommend using HTML—just like we do on flutter.dev, dart.dev, and pub.dev. You should also consider separating your primary application experience—created in Flutter—from your landing page, marketing content, and help content—created using search-engine optimized HTML.

„Flutter Web is only for web apps!“

Flutter Web?


Improve the indexability (SEO) of Flutter apps on the web
c: new feature
192
542
c: proposal
customer: crowd
framework
P2
platform-web
team-web
triaged-web
#46789 opened on Dec 11, 2019 by deepak-kumar-swain
Server-side rendering for Flutter web
c: new feature
c: proposal
customer: crowd
59
384
P3
platform-web
team-web
triaged-web
#47600 opened on Dec 22, 2019 by wliumelb

Dart Web!

```
$ dart create --template web
```


```dart
import 'package:web/helpers.dart';

void main() {
  final now = DateTime.now();
  final element = document.querySelector('#output') as HTMLDivElement;
  element.text = 'The time is ${now.hour}:${now.minute}'
      ' and your Dart web app is running!';
}
```

```dart
class FooComponent extends StatelessComponent {
  const FooComponent({super.key});

  @override
  Iterable<Component> build(BuildContext context) sync* {
    yield DomComponent(
      tag: 'div',
      children: [
        DomComponent(
          tag: 'p',
          children: [
            Text('Hello World'),
          ],
        ),
      ],
    );
    yield DomComponent(
      tag: 'img',
      attributes: {
        'src': '/images/logo.png',
      },
    );
  }
}
```

```
class FooComponent extends StatelessComponent {
  const FooComponent({super.key});

  @override
  Iterable<Component> build(BuildContext context) sync* {
    yield DomComponent(
      tag: 'div',
      children: [
        DomComponent(
          tag: 'p',
          children: [
            Text('Hello World'),
          ],
        ),
      ],
    );
    yield DomComponent(
      tag: 'img',
      attributes: {
        'src': '/images/logo.png',
      },
    );
  }
}
```

```
Node a = document.createElement('p');
a.text = 'Hello World';
Node b = document.createElement('div');
b.append(a);
Node c = document.createElement('img');
c.setAttribute('src', '/images/logo.png');
```

```
<div>
    <p>Hello World</p>
</div>
<img src="/images/logo.png" />
```

Flutters Rendering System

Widget

FooWidget

Holds config for a piece of UI

Has a public API

Frequently rebuilds

Element

FooElement

Represents an actual piece of the UI

Rarely rebuilds

RenderObject

RenderFoo

Renders to the screen

Rarely rebuilds

Jasprs Rendering System

Component

FooComponent

Holds config for a piece of UI

Has a public API

Frequently rebuilds

Element

FooElement

Represents an actual piece of the UI

Rarely rebuilds

DOM Node

RenderFoo

Just plain DOM nodes

Rarely rebuilds


I MADE THIS
YOU MADE THIS ?
Widget System
Widget System
Widget System
I MADE THIS
Component System

Island
Architecture

Routing

Server Side
Rendering

Client Side
Hydration

CLI

Theming

Building a framework?!

Javascript
Integrations

Deployment

Using custom
Backends

Development
Server

Testing

CSS Library
Integrations

```
runApp( FooComponent );
```

Jaspr

Pre-rendering

on the
server

HTML
(as a String)

(Continued) rendering

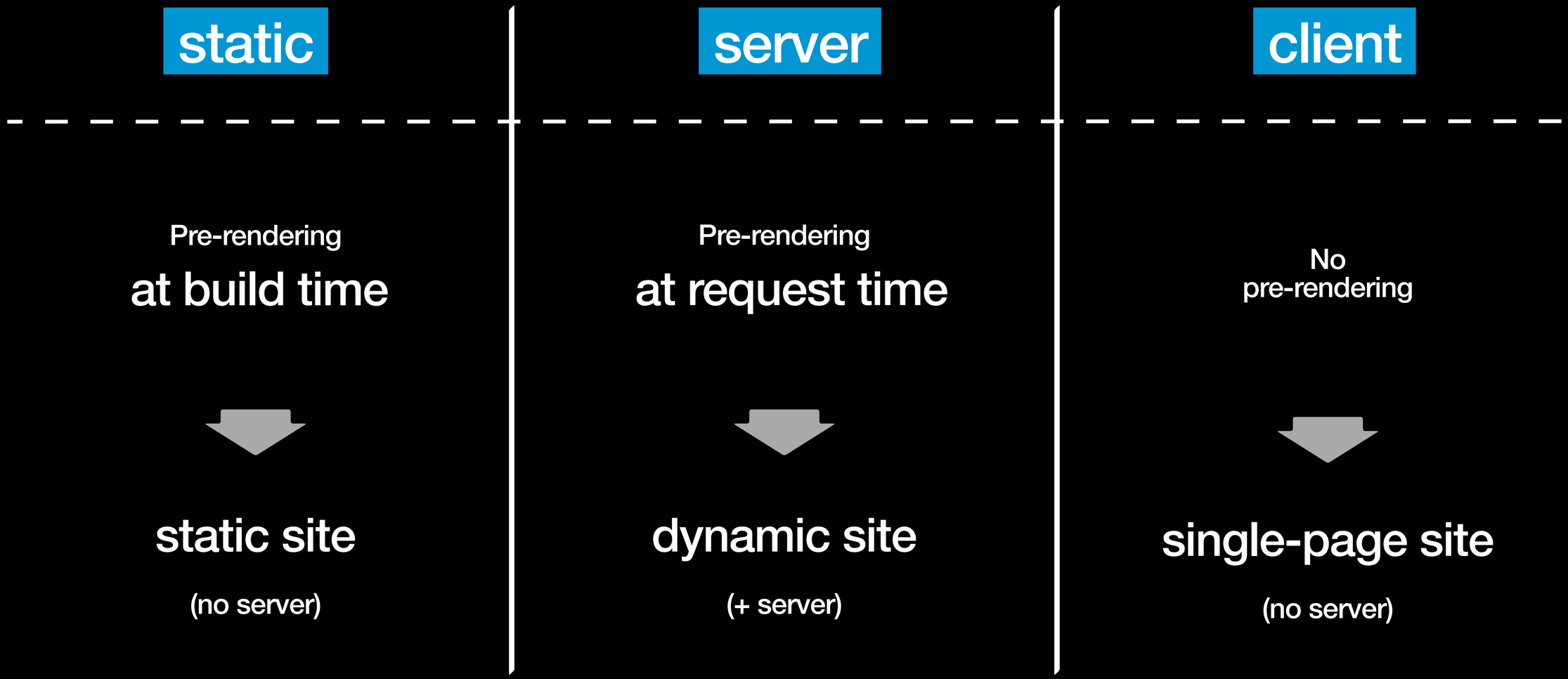
on the
client

DOM instructions

Browser

HTML Elements

Jasprs Rendering Modes



r/dartlang

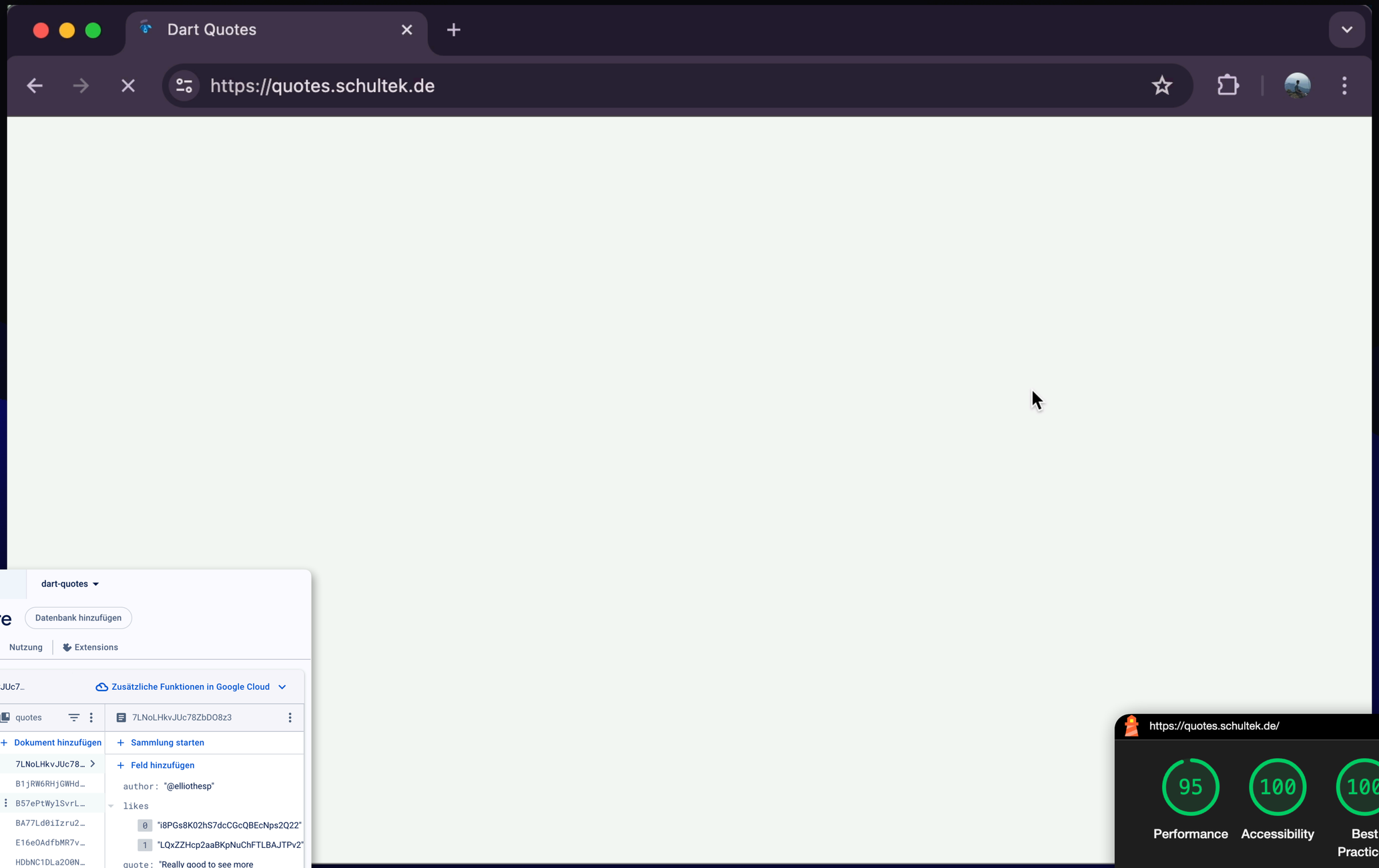
Top All Time

u/schultek • 2 yr. ago

Introducing Jaspr - A complete web-framework written in Dart

Hi fellow developers, I'm really excited to finally share with you what I've been working on over the last few months: Jaspr is a complete web-framework that is written in Dart. It takes inspiration from frameworks like React or Remix, but most importantly Flutter. While it renders normal HTML & CSS, it looks and feels very much like Flutter. So while you get all the benefits of an actual web app (fast page load, seo, server-side rendering, etc.) you get the familiar structure of stateless, stateful and inherited widgets. There is so much to talk about, but here are some highlights: 1. Familiar component model similar to Flutter widgets but adapted for the web It...

Jaspr in Action


Dart Quotes
https://quotes.schultek.de

Firebase
dart-quotes
Cloud Firestore
Datenbank hinzufügen
Daten
Regeln
Indexe
Nutzung
Extensions
quotes
7LNoLHkvJUc7…
Zusätzliche Funktionen in Google Cloud
(default)
quotes
7LNoLHkvJUc78ZbDO8z3
Sammlung starten
Dokument hinzufügen
Sammlung starten
quotes
7LNoLHkvJUc78…
Feld hinzufügen
B1jRW6RHjGWHd…
author: "@elliothesp"
B57ePtWylSvrL…
likes
BA77Ld0iIzru2…
0 "i8PGs8K02hS7dcCGcQBEcNps2Q22"
E16eOAdfbMR7v…
1 "LQxZZHcp2aaBKpNuChFTLBAJTPv2"
HDbNC1DLa2O0N…
quote: "Really good to see more pushing the backend Dart space forwards."
N7pn1TTNkOihE…

https://quotes.schultek.de/
95
Performance
100
Accessibility
100
Best Practices
100
SEO

```
$ jaspr create

Specify a target directory: myapp

Select a rendering mode:
❯ ◉  static: Build a statically pre-rendered site.
❯ ◉  server: Build a server-rendered site.
  ○  client: Build a purely client-rendered site.

(Recommended) Enable automatic hydration on the client? (Y/n) Yes

Setup routing for different pages of your site? (Y/n) Yes

(Recommended) Use multi-page (server-side) routing?
Choosing [no] sets up a single-page application with client-side routing instead. (Y/n) Yes

Setup Flutter web embedding? (y/N) No

Enable support for using Flutter web plugins in your project? (Y/n) Yes

✓ Generated 25 file(s)
Created project myapp! In order to get started, run the following commands:

cd s
jaspr serve
```

main.dart

```dart
import 'package:jaspr/server.dart';

import 'app.dart';
import 'jaspr_options.dart';

void main() {
  Jaspr.initializeApp(
    options: defaultJasprOptions,
  );

  runApp(Document(
    title: 'Dart Quotes',
    body: App(),
  ));
}
```

main.dart

```dart
import 'package:jaspr/server.dart';

import 'app.dart';
import 'jaspr_options.dart';

void main() {
  Jaspr.initializeApp(
    options: defaultJasprOptions,
  );

  runApp(Document(
    title: 'Dart Quotes',
    lang: 'en',
    meta: {
      "description": "A collection of cool Dart quotes. Built for FullStackFlutter Conference.",
    },
    body: App(),
  ));
}
```

main.dart

```dart
import 'package:jaspr/server.dart';

import 'app.dart';
import 'jaspr_options.dart';

void main() {
  Jaspr.initializeApp(
    options: defaultJasprOptions,
  );

  runApp(Document(
    title: 'Dart Quotes',
    lang: 'en',
    meta: {
      "description": "A collection of cool Dart quotes. Built for FullStackFlutter Conference.",
    },
    styles: [
      // Root styles
      css('html, body')
          .text(fontFamily: const FontFamily.list([FontFamily('Roboto'), FontFamilies.sansSerif]))
          .box(width: 100.percent, minHeight: 100.vh)
          .box(margin: EdgeInsets.zero, padding: EdgeInsets.zero)
          .background(color: Color.hex('#F7F7F7')),
      css('h1').text(fontSize: 4.rem).box(margin: EdgeInsets.unset),
    ],
    body: App(),
  ));
}
```

```dart
  });

  runApp(Document(
    title: 'Dart Quotes',
    lang: 'en',
    meta: {
      "description": "A collection of cool Dart quotes. Built for FullStackFlutter Conference.",
    },
    styles: [
      // Root styles
      css('html, body')
          .text(fontFamily: const FontFamily.list([FontFamily('Roboto'), FontFamilies.sansSerif]))
          .box(width: 100.percent, minHeight: 100.vh)
          .box(margin: EdgeInsets.zero, padding: EdgeInsets.zero)
          .background(color: Color.hex('#F7F7F7')),
      css('h1').text(fontSize: 4.rem).box(margin: EdgeInsets.unset),

      // Include text font
      StyleRule.fontFace(fontFamily: "Roboto", url: "/fonts/Roboto-Regular.ttf"),
      StyleRule.fontFace(fontFamily: "Roboto", fontStyle: FontStyle.italic, url: "/fonts/Roboto-Italic.t

      // Include icon font
      StyleRule.fontFace(fontFamily: "icomoon", url: "/fonts/icomoon.ttf"),
      css('[class^="icon-"], [class*=" icon-"]').text(fontFamily: FontFamily('icomoon')),
      css('.icon-heart-o:before').raw({'content': r'"\e900"'}),
      css('.icon-heart:before').raw({'content': r'"\e901"'}),

    ],
    body: App(),
  ));
}
```

```
<style>


html, body {
    font-family: 'Roboto', sans-serif;
    width: 100%;
    min-height: 100vh;
    padding: 0;
    margin: 0;
    background-color: #F7F7F7;
}
h1 {
    font-size: 4rem;
    margin: unset;
}

@font-face {
    font-family: "Roboto";
    src: url(/fonts/Roboto-Regular.ttf);
}
@font-face {
    font-family: "Roboto";
    font-style: italic;
    src: url(/fonts/Roboto-Italic.ttf);
}


</style>
```

```
@font-face {
    font-family: "icomoon";
    src: url(/fonts/icomoon.ttf);
}
[class^="icon-"], [class*=" icon-"] {
    font-family: 'icomoon';
}
.icon-heart-o:before {
    content: "\e900";
}
.icon-heart:before {
    content: "\e901";
}
```

```dart
  ));

  runApp(Document(
    title: 'Dart Quotes',
    lang: 'en',
    meta: {
      "description": "A collection of cool Dart quotes. Built for FullStackFlutter Conference.",
    },
    styles: [
      // Root styles
      css('html, body')
          .text(fontFamily: const FontFamily.list([FontFamily('Roboto'), FontFamilies.sansSerif]))
          .box(width: 100.percent, minHeight: 100.vh)
          .box(margin: EdgeInsets.zero, padding: EdgeInsets.zero)
          .background(color: Color.hex('#F7F7F7')),
      css('h1').text(fontSize: 4.rem).box(margin: EdgeInsets.unset),

      // Include text font
      StyleRule.fontFace(fontFamily: "Roboto", url: "/fonts/Roboto-Regular.ttf"),
      StyleRule.fontFace(fontFamily: "Roboto", fontStyle: FontStyle.italic, url: "/fonts/Roboto-Italic.t

      // Include icon font
      StyleRule.fontFace(fontFamily: "icomoon", url: "/fonts/icomoon.ttf"),
      css('[class^="icon-"], [class*=" icon-"]').text(fontFamily: FontFamily('icomoon')),
      css('.icon-heart-o:before').raw({'content': r'"\e900"'}),
      css('.icon-heart:before').raw({'content': r'"\e901"'}),

    ],
    body: App(),
  ));
}
```

app.dart

```dart
import 'package:jaspr/jaspr.dart';
import 'package:jaspr_router/jaspr_router.dart';

import 'pages/home_page.dart';
import 'pages/quote_page.dart';

class App extends StatelessComponent {

  @override
  Iterable<Component> build(BuildContext context) sync* {
    yield div(classes: 'main', [
      Router(
        routes: [
          Route(path: '/', builder: (context, state) {
            return HomePage();
          }),
          Route(path: '/quote/:quoteId', builder: (context, state) {
            return QuotePage(id: state.params['quoteId']!);
          }),
        ],
      ),
    ]);
  }

}
```

home_page.dart

```dart
import 'package:dart_quotes/data/firebase.dart';
import 'package:jaspr/server.dart';

class HomePage extends StatelessComponent {
  const HomePage({super.key});

  @override
  Iterable<Component> build(BuildContext context) sync* {
    yield header([
      img(src: 'images/quote.jpg', alt: "Quote symbol", width: 100),
      h1([text('Dart Quotes')]),
    ]);

    yield ul([
      ...
   ]);
  }

}
```

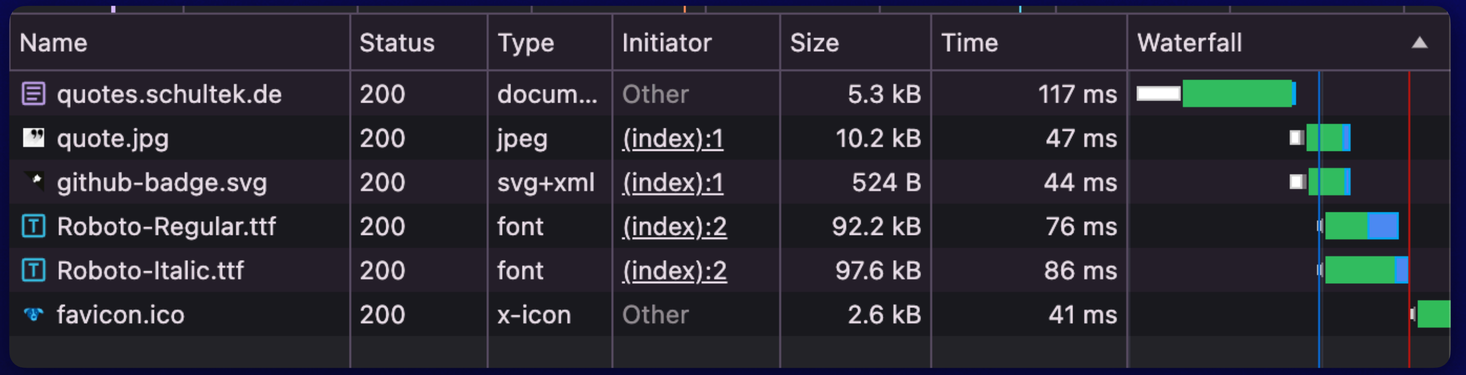
firebase.dart

```dart
import 'dart:io';

import 'package:dart_quotes/data/quote.dart';
import 'package:jaspr/jaspr.dart';

import 'package:dart_firebase_admin/dart_firebase_admin.dart';
import 'package:dart_firebase_admin/firestore.dart';

class FirebaseService {
  static FirebaseService instance = FirebaseService();

  late final FirebaseAdminApp adminApp = FirebaseAdminApp.initializeApp(
    'dart-quotes',
    Credential.fromServiceAccount(File('service-account.json')),
  );

  late final Firestore adminFirestore = Firestore(adminApp);


  Future<List<Quote>> loadQuotes() async {
    var query = await adminFirestore.collection('quotes').get();
    return query.docs.map((doc) {
      return Quote.fromData(doc.id, doc.data());
    }).toList();
  }

}
```

home_page.dart

```dart
import 'package:dart_quotes/data/firebase.dart';
import 'package:jaspr/server.dart';

class HomePage extends StatelessComponent {
  const HomePage({super.key});

  @override
  Iterable<Component> build(BuildContext context) sync* {
    yield header([
      img(src: 'images/quote.jpg', alt: "Quote symbol", width: 100),
      h1([text('Dart Quotes')]),
    ]);

    yield ul([
      ...
   ]);
  }

}
```

```dart
import 'package:dart_quotes/data/firebase.dart';
import 'package:jaspr/server.dart';

class HomePage extends StatelessComponent {
  const HomePage({super.key});

  @override
  Iterable<Component> build(BuildContext context) sync* {
    yield header([
      img(src: 'images/quote.jpg', alt: "Quote symbol", width: 100),
      h1([text('Dart Quotes')]),
    ]);

    yield ul([
      AsyncBuilder(builder: (context) async* {
        var quotes = await FirebaseService.instance.loadQuotes();
        quotes.sort((a, b) => b.likes.length.compareTo(a.likes.length));

        for (var quote in quotes) {
          yield li([
            a(href: '/quote/${quote.id}', [
              p([text(quote.quote)]),
              span([text(quote.author)]),
            ]),
          ]);
        }
      }),
    ]);
  }

}
```


Name
Status
Type
Initiator
Size
Time
Waterfall
quotes.schultek.de
200
docum...
Other
5.3 kB
117 ms
quote.jpg
200
jpeg
(index):1
10.2 kB
47 ms
github-badge.svg
200
svg+xml
(index):1
524 B
44 ms
Roboto-Regular.ttf
200
font
(index):2
92.2 kB
76 ms
Roboto-Italic.ttf
200
font
(index):2
97.6 kB
86 ms
favicon.ico
200
x-icon
Other
2.6 kB
41 ms

quote_page.dart

```dart
import 'package:jaspr/server.dart';

class QuotePage extends AsyncStatelessComponent {
  const QuotePage({required this.id, super.key});

  final String id;

  @override
  Stream<Component> build(BuildContext context) async* {
    var quote = await FirebaseService.instance.getQuoteById(id);

    yield nav([
      a(href: "/", [text('Home')]),
    ]);

    yield div(classes: "center", [
      div(classes: "quote-container", [
        img(classes: "quotes-start", src: 'images/quote.jpg', alt: "Starting quote symbol", width: 100),
        h1([text(quote.quote)]),
        p([text(quote.author)]),
        QuoteLikeButton(id: id, initialCount: quote.likes.length),
        img(classes: "quotes-end", src: 'images/quote.jpg', alt: "Ending quote symbol", width: 100),
      ])
    ]);
  }
}
```

firebase.dart

```dart
class FirebaseService {
  ...

  Stream<Quote?> getQuoteById(String id) async* {
    var doc = await adminFirestore.doc('quotes/$id').get();
    if (doc.exists) {
      yield Quote.fromData(doc.id, doc.data()!);
    } else {
      yield null;
    }
  }
}
```

quote_page.dart

```dart
import 'package:jaspr/server.dart';

class QuotePage extends AsyncStatelessComponent {
  const QuotePage({required this.id, super.key});

  final String id;

  @override
  Stream<Component> build(BuildContext context) async* {
    var quote = await FirebaseService.instance.getQuoteById(id).single;

    yield nav([
      a(href: "/", [text('Home')]),
    ]);

    yield div(classes: "center", [
      div(classes: "quote-container", [
        img(classes: "quotes-start", src: 'images/quote.jpg', alt: "Starting quote symbol", width: 100),
        h1([text(quote.quote)]),
        p([text(quote.author)]),
        QuoteLikeButton(id: id, initialCount: quote.likes.length),
        img(classes: "quotes-end", src: 'images/quote.jpg', alt: "Ending quote symbol", width: 100),
      ])
    ]);
  }
}
```

quote_like_button.dart

```dart
import 'package:dart_quotes/data/firebase.dart';
import 'package:jaspr/jaspr.dart';


class QuoteLikeButton extends StatelessComponent {
  const QuoteLikeButton({required this.id, required this.initialCount});

  final String id;
  final int initialCount;

  @override
  Iterable<Component> build(BuildContext context) sync* {
    yield StreamBuilder(
      stream: FirebaseService.instance.getQuoteById(id),
      builder: (context, snapshot) sync* {
        int count = snapshot.data?.likes.length ?? initialCount;
        bool? hasLiked = snapshot.data?.likes.contains(FirebaseService.instance.getUserId());

        yield button(
          classes: "quote-like-btn${hasLiked == true ? ' active' : ''}",
          [
            span(classes: "icon-heart${hasLiked ?? false ? '' : '-o'}", []),
            text(' $count'),
          ],
        );
      },
    );
  }
}
```

quote_like_button.dart

```dart
import 'package:dart_quotes/data/firebase.dart';
import 'package:jaspr/jaspr.dart';

@client
class QuoteLikeButton extends StatelessComponent {
  const QuoteLikeButton({required this.id, required this.initialCount});
```

quote_like_button.client.dart.js

```dart
@override
Iterable<Component> build(BuildContext context) sync* {
  yield StreamBuilder(
    stream: FirebaseService.instance.getQuoteById(id),
```

firebase.dart

```dart
import 'package:dart_firebase_admin/dart_firebase_admin.dart';

import 'package:dart_firebase_admin/firestore.dart';



class FirebaseService {
  ...

  Stream<Quote?> getQuoteById(String id) async* {

      var doc = await adminFirestore.doc('quotes/$id').get();
      if (doc.exists) {
        yield Quote.fromData(doc.id, doc.data()!);
      } else {
        yield null;
      }



  }
}
```

firebase.dart

```dart
@Import.onServer('package:dart_firebase_admin/dart_firebase_admin.dart',
  show: [#FirebaseAdminApp, #Credential])
@Import.onServer('package:dart_firebase_admin/firestore.dart',
  show: [#Firestore])
import 'firebase.imports.dart';

class FirebaseService {
  ...

  Stream<Quote?> getQuoteById(String id) async* {

      var doc = await adminFirestore.doc('quotes/$id').get();
      if (doc.exists) {
        yield Quote.fromData(doc.id, doc.data()!);
      } else {
        yield null;
      }



  }
}
```

firebase.dart

```dart
@Import.onServer('package:dart_firebase_admin/dart_firebase_admin.dart',
  show: [#FirebaseAdminApp, #Credential])
@Import.onServer('package:dart_firebase_admin/firestore.dart',
  show: [#Firestore])
import 'firebase.imports.dart';

class FirebaseService {
  ...

  Stream<Quote?> getQuoteById(String id) async* {
    if (!kIsWeb) {
      var doc = await adminFirestore.doc('quotes/$id').get();
      if (doc.exists) {
        yield Quote.fromData(doc.id, doc.data()!);
      } else {
        yield null;
      }
    } else {

    }
  }
}
```

```dart
@Import.onWeb('package:firebase_core/firebase_core.dart',
  show: [#Firebase, #FirebaseApp])
@Import.onWeb('package:cloud_firestore/cloud_firestore.dart',
  show: [#FirebaseFirestore])
@Import.onServer('package:dart_firebase_admin/dart_firebase_admin.dart',
  show: [#FirebaseAdminApp, #Credential])
@Import.onServer('package:dart_firebase_admin/firestore.dart',
  show: [#Firestore])
import 'firebase.imports.dart';

class FirebaseService {
  ...

  Stream<Quote?> getQuoteById(String id) async* {
    if (!kIsWeb) {
      var doc = await adminFirestore.doc('quotes/$id').get();
      if (doc.exists) {
        yield Quote.fromData(doc.id, doc.data()!);
      } else {
        yield null;
      }
    } else {

    }
  }
}
```

firebase.dart

firebase.dart

```dart
@Import.onWeb('package:firebase_core/firebase_core.dart',
  show: [#Firebase, #FirebaseApp])
@Import.onWeb('package:cloud_firestore/cloud_firestore.dart',
  show: [#FirebaseFirestore])
@Import.onServer('package:dart_firebase_admin/dart_firebase_admin.dart',
  show: [#FirebaseAdminApp, #Credential])
@Import.onServer('package:dart_firebase_admin/firestore.dart',
  show: [#Firestore])
import 'firebase.imports.dart';

class FirebaseService {
  ...

  Stream<Quote?> getQuoteById(String id) async* {
    if (!kIsWeb) {
      var doc = await adminFirestore.doc('quotes/$id').get();
      if (doc.exists) {
        yield Quote.fromData(doc.id, doc.data()!);
      } else {
        yield null;
      }
    } else {
      var snapshots = FirebaseFirestore.instance.collection('quotes').doc(id).snapshots();
      await for (var doc in snapshots) {
        yield Quote.fromData(doc.id, doc.data()!);
      }
    }
  }
}
```

quote_like_button.dart

```dart
import 'package:dart_quotes/data/firebase.dart';
import 'package:jaspr/jaspr.dart';


class QuoteLikeButton extends StatelessComponent {
  const QuoteLikeButton({required this.id, required this.initialCount});

  ...

  @override
  Iterable<Component> build(BuildContext context) sync* {

    ...

        yield button(
          classes: "quote-like-btn${hasLiked == true ? ' active' : ,'}",




          [
            span(classes: "icon-heart${hasLiked ?? false ? '' : '-o'}", []),
            text(' $count'),
          ],
        );
  }
}
```

quote_like_button.dart

```dart
import 'package:dart_quotes/data/firebase.dart';
import 'package:jaspr/jaspr.dart';


class QuoteLikeButton extends StatelessComponent {
  const QuoteLikeButton({required this.id, required this.initialCount});

  ...

  @override
  Iterable<Component> build(BuildContext context) sync* {

    ...

        yield button(
          classes: "quote-like-btn${hasLiked == true ? ' active' : ,'}",
          onClick: () {
            FirebaseService.instance.toggleLikeOnQuote(id, !hasLiked);



          },
          [
            span(classes: "icon-heart${hasLiked ?? false ? '' : '-o'}", []),
            text(' $count'),
          ],
        );
  }
}
```

quote_like_button.dart

```dart
import 'package:dart_quotes/data/firebase.dart';
import 'package:jaspr/jaspr.dart';


class QuoteLikeButton extends StatelessComponent {
  const QuoteLikeButton({required this.id, required this.initialCount});

  ...

  @override
  Iterable<Component> build(BuildContext context) sync* {

    ...

        yield button(
          classes: "quote-like-btn${hasLiked == true ? ' active' : ,'}",
          onClick: () {
            FirebaseService.instance.toggleLikeOnQuote(id, !hasLiked);
            if (!hasLiked) {
              ConfettiService.instance.show(emojis: ['💙', '🎯']);
            }
          },
          [
            span(classes: "icon-heart${hasLiked ?? false ? '' : '-o'}", []),
            text(' $count'),
          ],
        );
  }
}
```

js-confetti

0.12.0 • Public • Published 5 months ago

Readme | Code Beta | 0 Dependencies

npm package 0.12.0 | downloads 27k/week | jsDelivr 4M hits/month | minzipped size 2.41 kB

🎉 JavaScript Confetti library

💥 Supports emojis as confetti
⚡ Zero dependencies used
🦄 Works without any config, yet configurable
🛠️ Has TypeScript typings
🧩 Confetti speed adapts to user screen width

Links: GitHub | NPM | Demo

Install

You can install library from NPM using yarn or npm

```
yarn add js-confetti
```

Alternatively you can download script from CDN

```
<script src="https://cdn.jsdelivr.net/npm/js-confetti@latest/dist/js-confe
```

and then access `JSConfetti` global variable

Usage

js-confetti

0.12.0 • Public • Published 5 months ago

Readme | Code Beta | 0 Dependencies

npm package 0.12.0 | downloads 27k/week | jsDelivr 4M hits/month | minzipped size 2.41 kB

🎉 JavaScript Confetti library

- 💥 Supports emojis as confetti
- ⚡ Zero dependencies used
- 🦄 Works without any config, yet configurable
- 🛠️ Has TypeScript typings
- 🧩 Confetti speed adapts to user screen width

Links: GitHub | NPM | Demo

Install

You can install library from NPM using yarn or npm

```
yarn add js-confetti
```

Alternatively you can download script from CDN

```
<script src="https://cdn.jsdelivr.net/npm/js-confetti@latest/dist/js-confe
```

and then access `JSConfetti` global variable

Usage

quote_page.dart

```
yield Head(
  children: [
    script(src: „https://cdn.jsdelivr.net/npm/js-confetti[...].js“,
      defer: true, []),
  ],
);
```

and then access `JSConfetti` global variable

Usage

Initialize instance of JSConfetti class and call addConfetti method

```
import JSConfetti from 'js-confetti'

const jsConfetti = new JSConfetti()

jsConfetti.addConfetti()
```

NOTE `new JSConfetti()` creates HTML Canvas element and adds it to page, so call it only once!

Customise confetti

Use emojis as confetti:

```
jsConfetti.addConfetti({
   emojis: ['🌈', '⚡', '💥', '✨', '💫', '🌸'],
})
```

confetti_interop.dart

```dart
import 'dart:js_interop';

extension type JSConfetti._(JSObject _) {
  external JSConfetti();

  external void addConfetti([ConfettiOptions? options]);

  static JSConfetti instance = JSConfetti();

  void show({List<String>? emojis}) {
    addConfetti(ConfettiOptions(emojis: emojis.jsify()));
  }
}

extension type ConfettiOptions._(JSObject o) implements JSObject {
  external ConfettiOptions({JSAny? emojis});
}
```

and then access `JSConfetti` global variable

Usage

Initialize instance of JSConfetti class and call addConfetti method

```
import JSConfetti from 'js-confetti'

const jsConfetti = new JSConfetti()

jsConfetti.addConfetti()
```

NOTE `new JSConfetti()` creates HTML Canvas element and adds it to page, so call it only once!

Customise confetti

Use emojis as confetti:

```
jsConfetti.addConfetti({
   emojis: ['🌈', '⚡', '💥', '✨', '💫', '🌸'],
})
```

confetti_interop.dart

```dart
import 'dart:js_interop';

extension type JSConfetti._(JSObject _) {
  external JSConfetti();

  external void addConfetti([ConfettiOptions? options]);

  static JSConfetti instance = JSConfetti();

  void show({List<String>? emojis}) {
    addConfetti(ConfettiOptions(emojis: emojis.jsify()));
  }
}
```

confetti.dart

```dart
extension type ConfettiOptions._(JSObject o) implements JSObject {
  external factory ConfettiOptions({JSAny? emojis});
}
@JS.onWeb('confetti_interop.dart', show: [#JSConfetti])
import 'confetti.imports.dart';

class ConfettiService {
  static ConfettiService instance = ConfettiService();

  void show({List<String>? emojis}) {
    JSConfetti.instance.show(emojis: emojis);
  }
}
```

What we have seen.

- Create a new project
- Building components and ui using HTML in Dart
- Styling using CSS in Dart
- Using meta tags for SEO
- `@client` components
- Platform specific `@Imports`
- Integrating firebase plugins

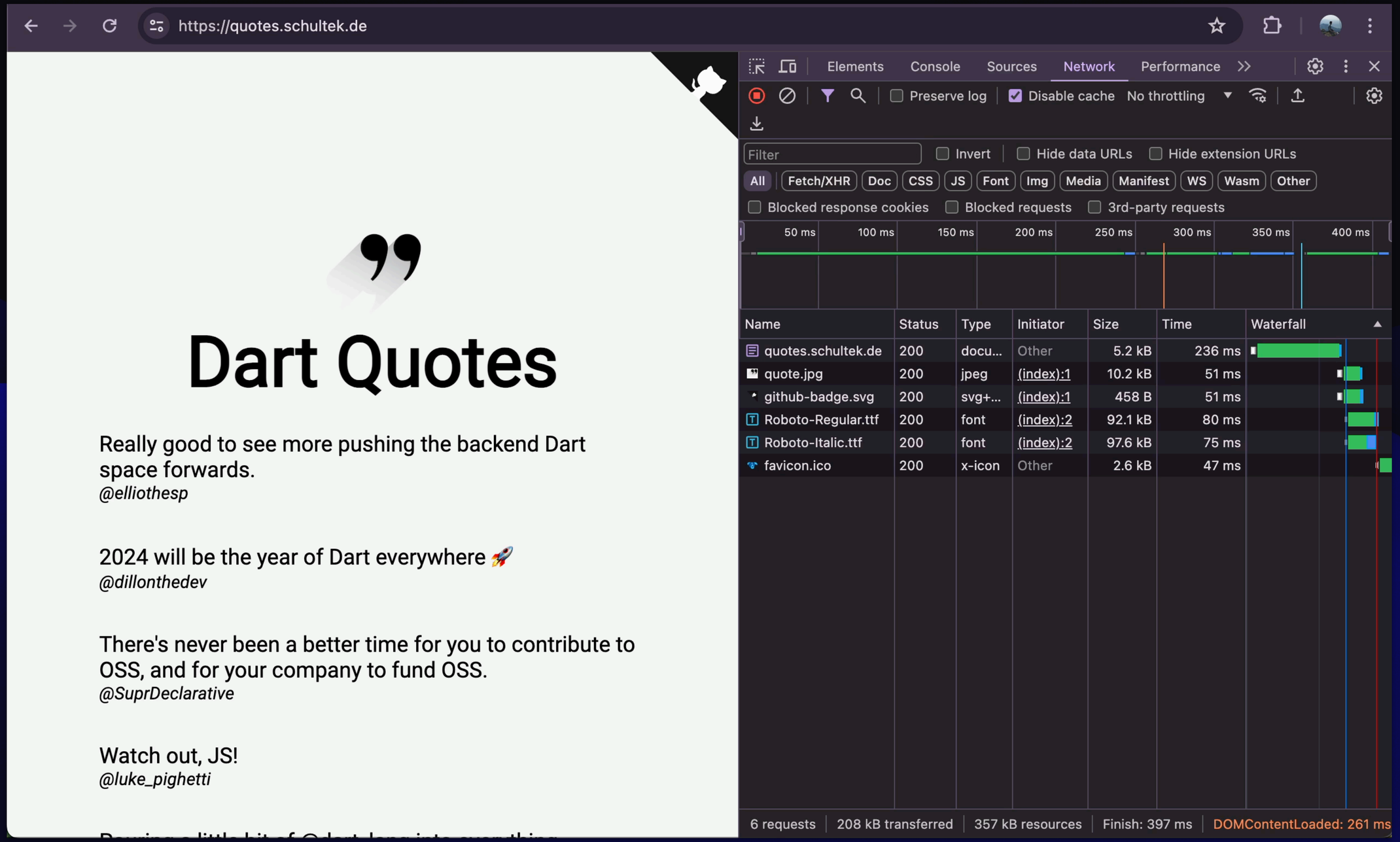
  (both server & client)
- Integrating external js libraries

https://quotes.schultek.de

Introducing Jaspr

Darts first full stack web framework

Kilian Schulte
@schultek


https://quotes.schultek.de
Dart Quotes
Really good to see more pushing the backend Dart space forwards.
@elliothesp
2024 will be the year of Dart everywhere 🚀
@dillonthedev
There's never been a better time for you to contribute to OSS, and for your company to fund OSS.
@SuprDeclarative
Watch out, JS!
@luke_pighetti
Elements
Console
Sources
Network
Performance
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
Hide extension URLs
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
WS
Wasm
Other
Blocked response cookies
Blocked requests
3rd-party requests
50 ms
100 ms
150 ms
200 ms
250 ms
300 ms
350 ms
400 ms
Name
Status
Type
Initiator
Size
Time
Waterfall
quotes.schultek.de 200 docu... Other 5.2 kB 236 ms
quote.jpg 200 jpeg (index):1 10.2 kB 51 ms
github-badge.svg 200 svg+... (index):1 458 B 51 ms
Roboto-Regular.ttf 200 font (index):2 92.1 kB 80 ms
Roboto-Italic.ttf 200 font (index):2 97.6 kB 75 ms
favicon.ico 200 x-icon Other 2.6 kB 47 ms
6 requests
208 kB transferred
357 kB resources
Finish: 397 ms
DOMContentLoaded: 261 ms