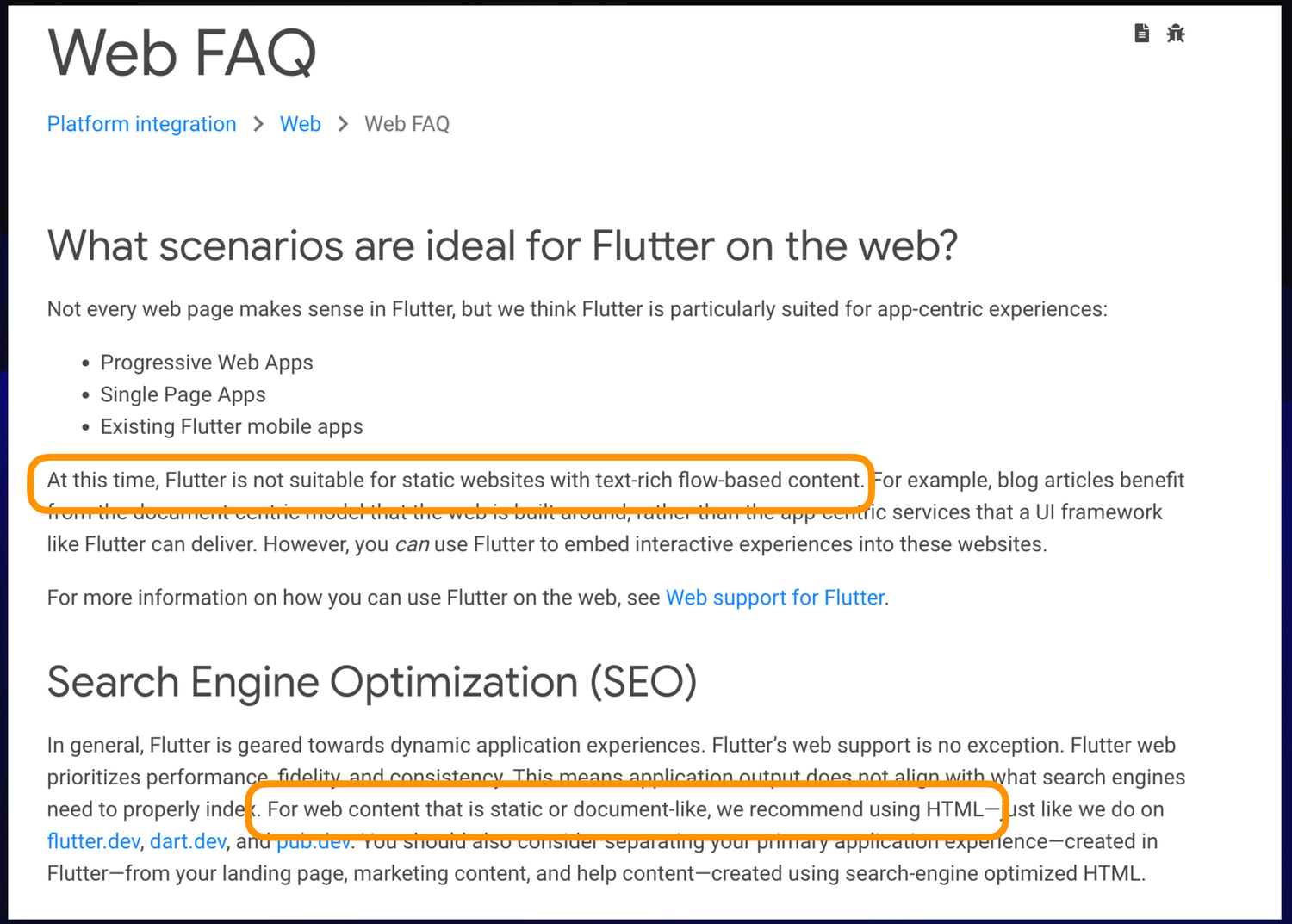
Jaspr: Unleashing the Power of Dart for Modern Web Development

Kilian Schulte
@schultek_dev

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?

/ Flutter Web

Sites

(More) text based
(More) linear
(More) static
Little or no user state
(More) consumption focused
Enable short interactions
Code-light

Apps

More *than* text
Not (as) linear
(More) dynamic/interactive
(Potentially) lots of user state
Creation / exploration focused
Focused on long interactions
Code-heavy

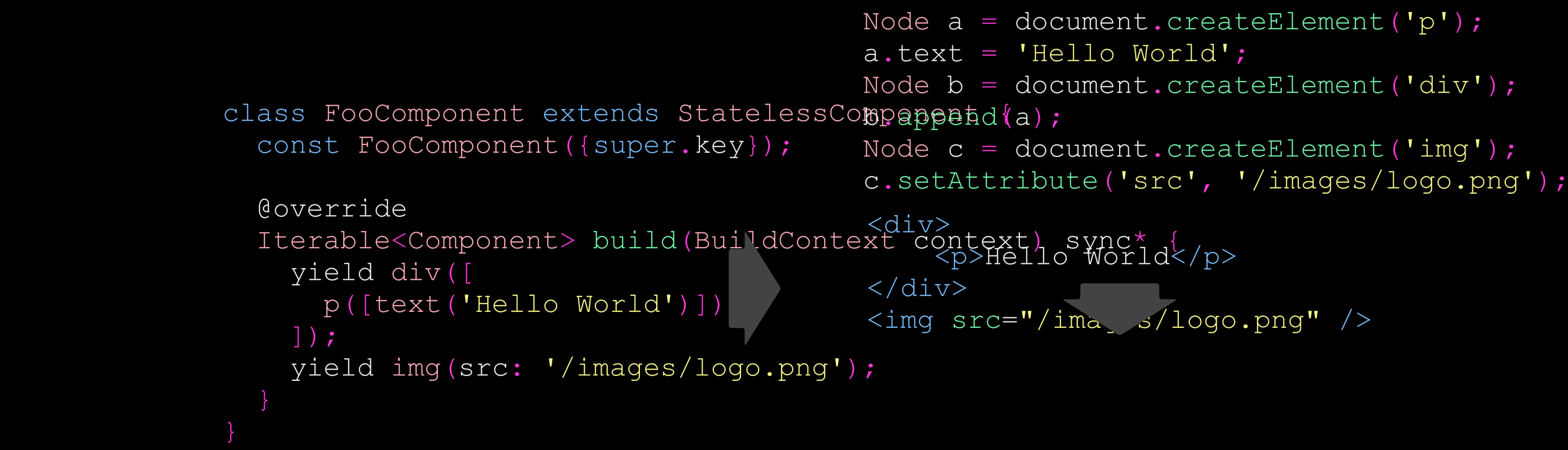
Flutter Web?

/ Flutter Web

Sites	Apps
(More) text based	More *than* text
(More) linear	Not (as) linear
(More) static	(More) dynamic/interactive
Little or no user state	(Potentially) lots of user state
(More) consumption focused	Creation / exploration focused
Enable short interactions	Focused on long interactions
Code-light	Code-heavy

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

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


class FooComponent extends StatelessComponent {
const FooComponent({super.key});
@override
Iterable<Component> build(BuildContext context) sync* {
yield div([
p([text('Hello World')])
]);
yield img(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');

Hello World

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

SEO

A full framework?!

Javascript
Integrations

Using custom
Backends

Deployment

Development
Server

Testing

CSS Library
Integrations

```
runApp( FooComponent );
```

Jaspr

Pre-rendering

on the
server

HTML
(as a String)

(Continued) rendering

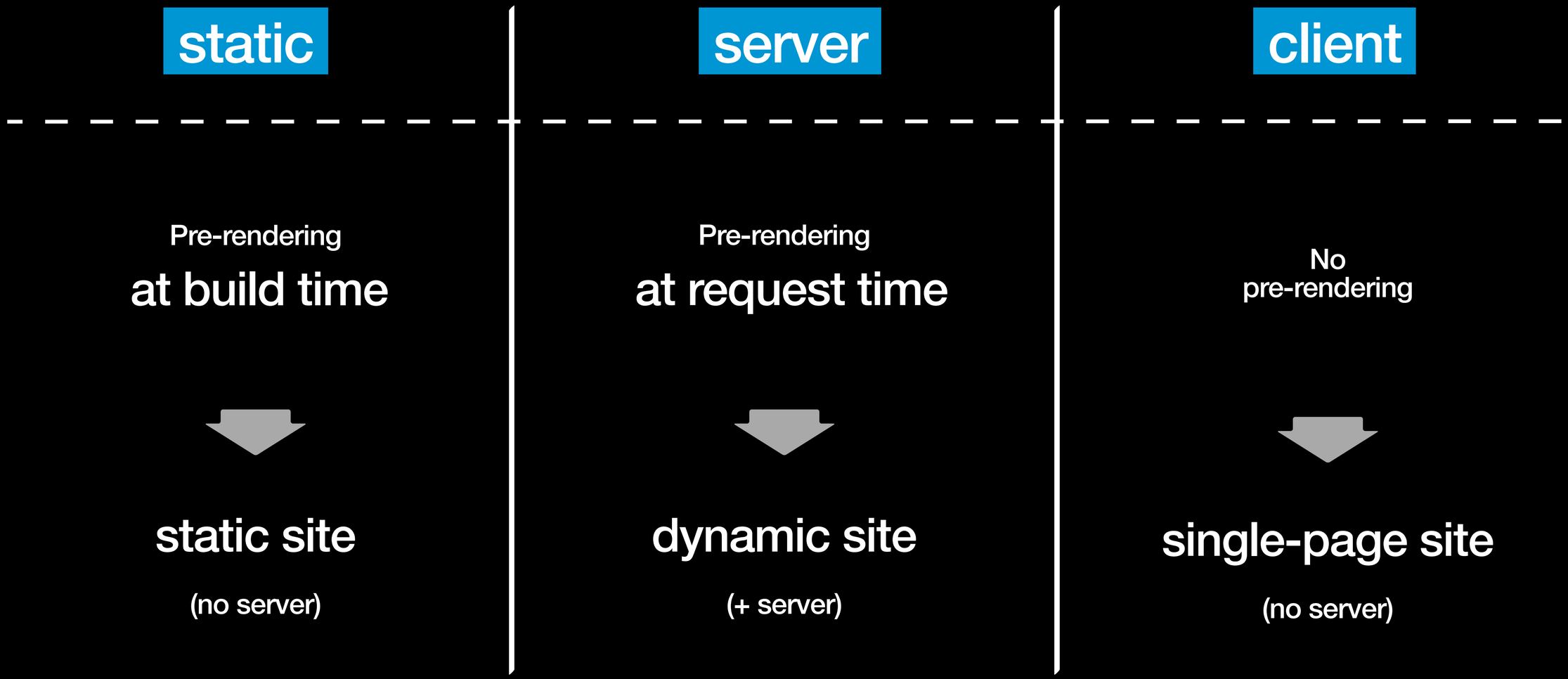
on the
client

DOM instructions

Browser

HTML Elements

Jasprs Rendering Modes



Let’s build!

Recap

- `jaspr <create|serve|build>`
- Build components as you would build widgets
- Styling using CSS in Dart
- Load data on the server using `AsyncStatelessComponent`
- `@client` components for automatic interactivity
- Integrating flutter plugins (`shared_preferences`)
- Platform specific `@Imports`

I don’t want to write CSS manually

A. Use Tailwind with `jaspr_tailwind`

B. Use a css framework (e.g. Bulma, Bootstrap)

I have / need a Dart backend

Use Jaspr in `server` mode with your backend of choice:

`shelf`
`dart_frog`
`serverpod (soon)`

What about state management?

Its the same story as in Flutter!

`jaspr_riverpod`
`jaspr_bloc (community)`
`jaspr_provider (community)`

`(create a fork yourself, its easy)`

Try out Jaspr for yourself!

jasprpad.schultek.de

Kilian Schulte
@schultek_dev