

สูตรลัด Jaspr

สร้างเว็บไซต์แบบมือใหม่หัดขับด้วย Jaspr
และ Dart

อนุชิต ชโลธร

สำนักพิมพ์ก๊อปวาง



สูตรลัด Jaspr

สร้างเว็บไซต์แบบมือใหม่หัดขับด้วย
Jaspr และ Dart

อนุชิต ชโลธร

(อัปเดตครั้งที่ 2)

คำนำ

หนังสือเล่มนี้จะนำพาคุณดำดิ่งสู่การสร้างสรรคเว็บแอปพลิเคชันที่น่าทึ่งด้วย Jaspri เฟรมเวิร์กที่ทรงพลังและยืดหยุ่นแห่งยุค ไม่ว่าคุณจะเป็นนักพัฒนาที่เคยสัมผัสประสบการณ์กับ Flutter มาก่อน หรือเพิ่งเริ่มต้นในเส้นทางสายเว็บ Jaspri จะเป็นกุญแจสำคัญที่ปลดล็อกศักยภาพของคุณ ให้ทุกวิสัยทัศน์กลายเป็นจริงบนโลกออนไลน์

เราจะพาคุณสำรวจแก่นแท้ของ Jaspri ตั้งแต่การเริ่มต้นโปรเจกต์แรก การทำความเข้าใจโครงสร้างสถาปัตยกรรม การสร้างคอมโพเนนต์แบบโต้ตอบ ไปจนถึงการจัดการสถานะที่ซับซ้อน และการ Deploy แอปพลิเคชัน นอกจากนี้ เรายังจะเจาะลึกความสัมพันธ์ระหว่าง Jaspri และ Flutter เพื่อให้คุณเห็นภาพที่ชัดเจนว่าเฟรมเวิร์กทั้งสองทำงานร่วมกันอย่างไร และเมื่อใดที่แต่ละตัวคือตัวเลือกที่ดีที่สุดสำหรับงานของคุณ

เป้าหมายสูงสุดของเราคือการให้ความรู้และแรงบันดาลใจ เพื่อให้คุณสามารถเนรมิตเว็บแอปพลิเคชันที่สวยงาม ประสิทธิภาพสูง และใช้งานง่ายด้วย Jaspri เราเชื่อมั่นว่าการเรียนรู้ Jaspri จะเปิดประตูสู่โอกาสใหม่ๆ ในโลกของการพัฒนา

เว็บ และเราตื่นเต้นที่จะได้เป็นส่วนหนึ่งในการเดินทาง
สร้างสรรค์ของคุณ

ขอให้มีความสุขกับการเขียนโค้ด!

สารบัญ

แนะนำ Jaspr	1
แนวคิดเบื้องหลัง Jaspr	1
คุณสมบัติหลัก	2
ทดลองใช้งานใน JasprPad	3
Jaspr และ Flutter Web ความเหมือนที่แตกต่าง	6
ความแตกต่างในกรณีการใช้งาน	6
Jaspr	6
Flutter Web	7
ตัวอย่างการใช้งานจริง	8
ความแตกต่างเชิงแนวคิด	9
แนวทางการออกแบบของ Jaspr	10
ไม่มีคอมโพเนนต์สำเร็จรูป	10
การจัดการข้อความ (Text)	11
การจัดวางเลย์เอาต์ (Layout)	11
การฝังแอป Flutter (Flutter Embedding)	12
การใช้งาน Flutter Plugins	12

เริ่มต้นใช้งาน Jaspr	14
การใช้งานผ่าน VS Code Extension	14
ติดตั้ง Extension	14
สร้างโปรเจกต์ใหม่	15
รันและ Debug โปรเจกต์	16
การหยุดการทำงานของเซิร์ฟเวอร์	18
การใช้งานผ่าน CLI	19
ติดตั้ง Jaspr CLI	20
สร้างโปรเจกต์ใหม่	20
รัน Development Server	21
ตัวเลือกของโปรเจกต์ (Project Options)	21
Rendering Mode	22
Routing	22
Multi-Page Routing (ถ้าเปิดใช้งาน Routing)	22
Flutter Embedding	23
Plugin Support	23
Custom Backend	23
โหมดการเรนเดอร์	25
Static Mode	26

Server Mode	27
Client Mode	28
การเรนเดอร์ฝั่งเซิร์ฟเวอร์ vs ฝั่งไคลเอนต์	30
ความแตกต่างระหว่างสภาพแวดล้อมเซิร์ฟเวอร์และไคลเอนต์	31
คอมโพเนนต์ของฉันทถูกเรนเดอร์ที่ไหน?	33
เครื่องมือจัดการโปรเจกต์ Jaspr	37
คำสั่งหลัก	37
Development Server	38
การ Build โปรเจกต์	38
Static Mode	39
Server Mode	39
Client Mode	40
การตั้งค่าโปรเจกต์ Jaspr	41
โครงสร้างโปรเจกต์	43
ระบบคอมโพเนนต์	47
StatelessComponent	47
StatefulComponent	48
InheritedComponent	50

คอมโพเนนต์พื้นฐาน	51
Component.element()	52
Component.text()	53
Component.fragment()	54
Component.empty()	56
Component.wrapElement()	56
การจัดรูปแบบช่องว่าง	58
การเขียน HTML	61
HTML Utilities	62
ลองใช้งานบน JasprPad	63
ย่อหน้าพร้อมข้อความ Rich Text	63
หัวข้อสีน้ำเงิน	64
ลิงก์พร้อมรูปภาพ	64
Select Input	65
Progress Bar	66
การจัดสไตล์	67
CSS-in-Dart	67
สไตล์ระดับคอมโพเนนต์	67
สไตล์ชืดส่วนกลาง	69

สไตลแบบ Inline	69
สไตลแบบ Responsive	70
สไตลชีตภายนอก	71
การเพิ่มสไตลชีตแบบไดนามิก	73
การใช้ CSS Frameworks จากภายนอก	73
Tailwind CSS	74
Bulma	74
Dart Sass	75
การใช้ Tailwind CSS กับ Jaspr	76
ข้อกำหนดเบื้องต้น	76
การตั้งค่า	77
การใช้งาน	80
การตั้งค่า	81
ระบบ Routing	83
การตั้งค่า Routing	83
Single-page vs Multi-page Routing	84
Route-based Code Splitting	87
Lazy Routes	88
การนำทางโดยใช้คอมโพเนนต์ Link	89

การนำทางไปยัง URL	89
การนำทางย้อนกลับ	90
การนำทางไปยัง Named Route	90
การใช้ Extra Data	91
การเปลี่ยนเส้นทาง	92
Top-level vs Route-level Redirection	93
การเปลี่ยนเส้นทางไปยัง Named Route	93
ข้อควรพิจารณา	94
การโหลดล่วงหน้า	94
การดึงข้อมูล	96
การโหลดข้อมูลบนเซิร์ฟเวอร์	97
คอมโพเนนต์แบบอะซิงโครนัส	98
การโหลดสถานะล่วงหน้าของ StatefulComponent	100
การชิงข้อมูลไปยังไคลเอนต์	101
การส่งข้อมูลผ่านคอมโพเนนต์ @client	101
การใช้ @sync กับฟิลต์ใน StatefulComponents	103
SEO และ Meta Tags	104
การ Pre-rendering	104

ข้อมูล Meta	105
ข้อมูล Meta แบบคงที่	105
ข้อมูล Meta แบบไดนามิก	106
การลดปริมาณ JavaScript ที่โหลด	107
การแบ่งโค้ด	108
โหมด Client	109
โหมด Server/Static	110
เทคนิคอื่นๆ	112
การสร้างเว็บไซต์แบบ Static ด้วย Jaspr	114
การสร้างเพจ	114
การใช้ jaspr_router	115
การสร้าง Dynamic Routes	115
การใช้งานแบบ Manual	117
การสร้าง Sitemap	117
การทำให้เว็บไซต์ที่ Pre-render ได้ตอบโต้	120
การตั้งค่า Hydration	121
Hydration แบบอัตโนมัติ	122
Hydration แบบ Manual	122
คอมโพเนนต์ @client	127

การใช้งาน	128
การเริ่มต้นตรรกะฝั่งไคลเอนต์	129
การแชร์สถานะ	130
การส่งข้อมูล	133
การทำงานเบื้องหลัง	135
การทดสอบ	137
การตั้งค่า	137
ผู้ช่วยทดสอบ	138
testComponents	139
testClient	139
testServer	139
การ Deploy แอปพลิเคชัน Jaspr	140
Static Hosting	140
Firebase Hosting	141
Github Pages	141
Server Hosting	145
Docker	145
Globe	149
การฝั่งแอป Flutter	151

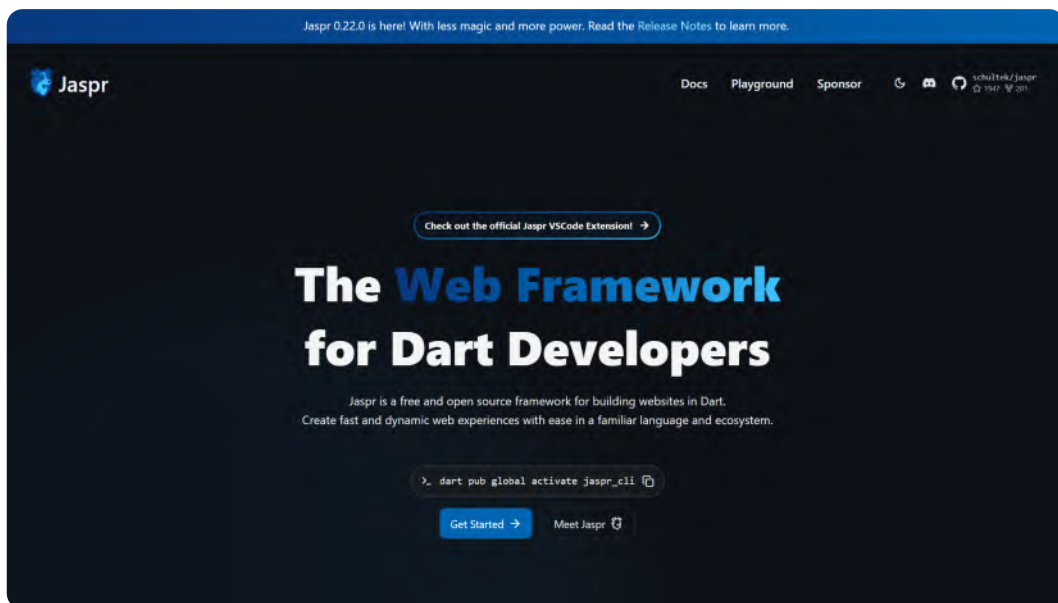
การโหลดแบบช้า	154
การจัดการ Import	154
การใช้งาน Riverpod ใน Jaspr	156
ความแตกต่างจาก flutter_riverpod	156
การเข้าถึง Providers	156
สิ่งทดแทน Consumer	158
การชิงชัยสถานะของ Provider	158
การรอ Async Providers	160
การ Override และ Scoping การชิงชัย	160
ทำไมถึงใช้ Context extensions แทน Consumer	162
ตัวอย่างแอปพยากรณ์อากาศ	163
สร้างโปรเจกต์ Jaspr	163
เรียกใช้บริการพยากรณ์อากาศ wttr.in	164
สร้างโมเดลข้อมูลพยากรณ์อากาศ	166
เพิ่ม Service สำหรับเรียกใช้ API	170
เพิ่มหน้าพยากรณ์อากาศ (Weather Page)	171
กำหนด Route และแก้ไขเมนูนำทาง	173
รันโปรเจกต์และทดลองใช้งาน	175
ตัวอย่างแอปเชื่อมต่อกับ Generative AI	179

สร้างโปรเจกต์ Serverpod	179
สร้างโปรเจกต์ Jaspr Frontend	180
ตั้งค่า Serverpod Backend	181
ติดตั้ง Dependencies และตั้งค่า Database	181
สร้าง Endpoint และโมเดลสำหรับ AI	182
ตั้งค่า Jaspr App ให้ทำงานร่วมกับ Serverpod	185
พัฒนาแอป Jaspr Frontend	187
ตั้งค่า Dependencies	188
สร้าง Service และ UI	188
เพิ่ม Route และเมนูนำทาง	192
ทดลองรันและ Deploy	193
ทดลองรันแอป Jaspr ในโหมดพัฒนา	193
Build และ Deploy ไปยัง Serverpod	194
สร้างคอมโพเนนต์จาก HyperUI	196
สร้างโปรเจกต์	197
ติดตั้ง Tailwind CLI	197
ติดตั้งแพ็คเกจ Tailwind สำหรับ Jaspr	199
เขียนคอมโพเนนต์จาก HyperUI	201
สร้าง Header Component	201

สร้าง Footer Component	205
สร้าง Section Component	208
ทดลองใช้งาน	212
บทส่งท้าย	217
ก้าวต่อไปของคุณ	219
ดาวนโหลดซอร์สโค้ด	221

แนะนำ Jaspr

Jaspr คือเฟรมเวิร์กเว็บสมัยใหม่สำหรับการพัฒนาเว็บไซต์ด้วยภาษา **Dart** ที่รองรับทั้งการเรนเดอร์ฝั่งไคลเอนต์ (Client-side Rendering) และฝั่งเซิร์ฟเวอร์ (Server-side Rendering)



เว็บไซต์ jaspr.site

แนวคิดเบื้องหลัง Jaspr

Jaspr ถูกออกแบบขึ้นจากแนวคิดในการสร้างเว็บเฟรมเวิร์กที่ให้ประสบการณ์การพัฒนาและรูปแบบการเขียนโค้ดที่ใกล้เคียงกับ **Flutter** มากที่สุด แต่ยังคงเรนเดอร์ผลลัพธ์ออกมาเป็น **HTML และ CSS มาตรฐาน** ซึ่งสอดคล้องกับธรรมชาติของเว็บ

ทำให้เหมาะสำหรับการพัฒนาเว็บไซต์จริงในเชิงโปรดัคชัน โดยเฉพาะในกรณีที่ Flutter Web อาจไม่ตอบโจทย์ทั้งหมด

เฟรมเวิร์กนี้มุ่งเน้นไปที่นักพัฒนา **Flutter** ที่ต้องการขยายขอบเขตการทำงานของตนเองไปสู่การสร้างเว็บไซต์หลากหลายรูปแบบ ไม่ว่าจะเป็นเว็บไซต์ทั่วไป เว็บไซต์ที่เน้นเนื้อหา หรือระบบที่ไม่จำเป็นต้องใช้สถาปัตยกรรมแบบแอปเต็มรูปแบบของ Flutter Web

เป้าหมายสำคัญของ Jaspr คือการขยายขีดความสามารถของภาษา **Dart** ให้ครอบคลุมทั้งฝั่งเว็บและฝั่งเซิร์ฟเวอร์ ผ่านการเป็น **Full-stack Web Framework** ที่ออกแบบมาอย่างรอบคอบ และพัฒนาด้วย Dart ตั้งแต่ต้นจนจบ เพื่อให้ นักพัฒนาสามารถใช้ภาษาเดียวสร้างเว็บแอปพลิเคชันได้อย่างครบวงจร

คุณสมบัติหลัก

Jaspr ออกแบบมาให้เริ่มต้นใช้งานได้อย่างเป็นธรรมชาติ สำหรับนักพัฒนาที่คุ้นเคยกับ Flutter ด้วยโมเดลคอมโพเนนต์ที่มีแนวคิดและโครงสร้างคล้ายกับ Widget ทำให้สามารถเรียนรู้และนำไปใช้ได้ทันทีโดยไม่ต้องปรับตัวมากนัก ขณะ

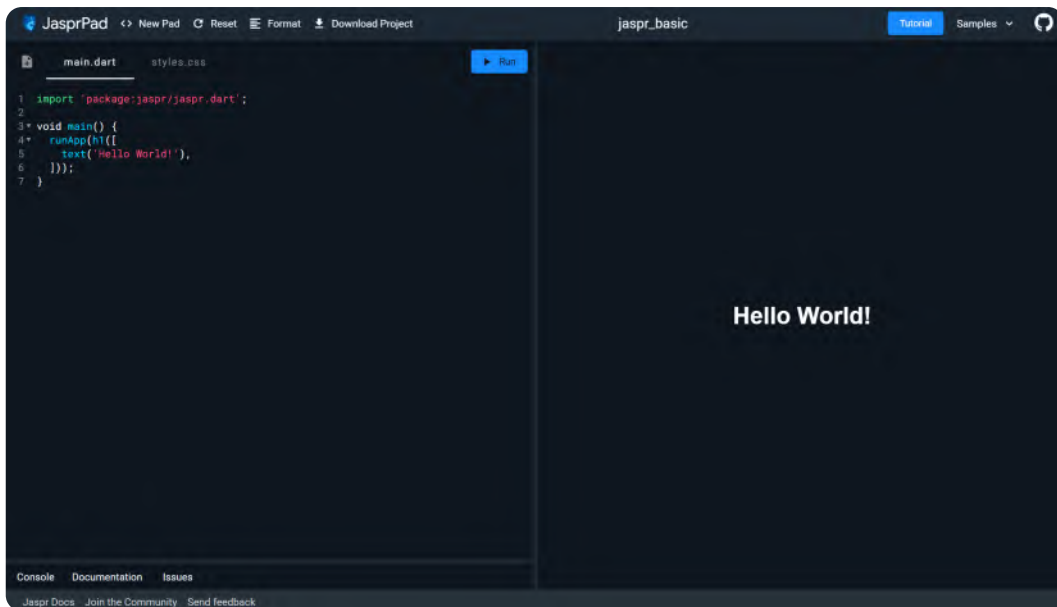
เดียวกันก็มีความทรงพลังด้วยการรองรับ **Server-side Rendering** มาให้ตั้งแต่ต้น ช่วยให้เว็บไซต์พร้อมใช้งานและเป็นมิตรกับ SEO โดยไม่ต้องเสียเวลาในการตั้งค่าเพิ่มเติม

ในด้านการใช้งาน Jaspr ช่วยลดความซับซ้อนของการพัฒนาเว็บด้วยการซิงก์สถานะของคอมโพเนนต์ระหว่างฝั่งเซิร์ฟเวอร์และไคลเอนต์ให้อัตโนมัติ นักพัฒนาไม่จำเป็นต้องจัดการ state ด้วยตนเองมากนัก ส่งผลให้โค้ดอ่านง่ายและดูแลรักษาได้ดีขึ้น นอกจากนี้ยังทำงานได้อย่างรวดเร็วด้วยการอัปเดต DOM เฉพาะส่วนที่มีการเปลี่ยนแปลงจริง ช่วยเพิ่มประสิทธิภาพและความสั่นไหวในการใช้งาน

อีกหนึ่งจุดเด่นคือความยืดหยุ่นสูง Jaspr สามารถรันได้ทั้งบนฝั่งเซิร์ฟเวอร์ ฝั่งไคลเอนต์ หรือทำงานร่วมกันทั้งสองฝั่ง พร้อมรองรับทั้งการตั้งค่าแบบอัตโนมัติสำหรับผู้ที่ต้องการความสะดวก และการปรับแต่งแบบกำหนดเองสำหรับกรณีที่ต้องการควบคุมพฤติกรรมของระบบอย่างละเอียด

ทดลองใช้งานใน JasprPad

Jaspr ได้รับแรงบันดาลใจจาก DartPad และมีเครื่องมือแก้ไขโค้ดออนไลน์ของตัวเองในชื่อ **JasprPad**



playground.jaspr.site

Jaspypad ช่วยให้คุณสามารถ

- ดูตัวอย่างโค้ด (Samples)
- เรียนรู้ผ่านบทเรียน (Tutorial)
- ทดลองใช้งาน Jaspr ได้ทันทีผ่านเบราว์เซอร์

เมื่อคุณต้องการพัฒนาต่อในเครื่องของตนเอง ก็สามารถดาวน์โหลดไฟล์ทั้งหมดออกมาเป็น **โปรเจกต์ Dart ที่พร้อมใช้งานทันที** ได้อย่างสะดวก

ที่สำคัญ Jaspypad เองก็ถูกพัฒนาด้วย **Jaspr** เช่นกัน คุณจึงสามารถศึกษาซอร์สโค้ดเพื่อทำความเข้าใจการใช้งาน Jaspr ในแอปพลิเคชันขนาดใหญ่ได้โดยตรง

Jaspr เหมาะสำหรับนักพัฒนาที่เขียนภาษา Dart และ Flutter ที่ต้องการสร้างเว็บไซต์ที่ Flutter Web ไม่สามารถทำได้ เช่น เว็บไซต์ที่เน้นเนื้อหา หรือเว็บไซต์ต้องการทำ SEO Jaspr ยังใช้ภาษา Dart เป็นภาษาในการพัฒนา คุณสามารถใช้ภาษา Dart พัฒนาได้ทั้งส่วน web และ server ได้ตามต้องการ

Jaspr และ Flutter Web ความเหมือนที่แตกต่าง

สำหรับนักพัฒนาที่ใช้ภาษา Dart ในการสร้างเว็บแอปพลิเคชัน คำถามสำคัญที่มักเกิดขึ้นคือ: ควรเลือกใช้ **Jaspr** หรือ **Flutter Web**? แม้ว่าทั้งสองจะเป็นเฟรมเวิร์กที่ยอดเยี่ยมและมีรากฐานมาจาก Dart เหมือนกัน แต่ทั้งสองถูกออกแบบมาเพื่อวัตถุประสงค์ที่แตกต่างกันอย่างชัดเจน

บทความนี้จะพาคุณไปสำรวจความแตกต่างที่สำคัญระหว่าง Jaspr และ Flutter Web ตั้งแต่ปรัชญาการออกแบบไปจนถึงกรณีการใช้งานจริง เพื่อช่วยให้คุณตัดสินใจได้อย่างมั่นใจว่าเครื่องมือใดจะตอบโจทย์โปรเจกต์เว็บของคุณได้ดีที่สุด

ความแตกต่างในกรณีการใช้งาน

Jaspr

Jaspr ถูกออกแบบมาเพื่อเป็นเฟรมเวิร์กสำหรับการพัฒนาเว็บไซต์ด้วยภาษา **Dart** ในทุกรูปแบบ ไม่ว่าจะเป็น

- เว็บไซต์แบบสถิต (Static Websites)
- เว็บไซต์ที่เรนเดอร์ฝั่งเซิร์ฟเวอร์ (Server-Rendered Websites)
- Single-Page Applications (SPA)

Jaspr ให้ประสบการณ์การพัฒนาที่ได้รับแรงบันดาลใจจาก Flutter แต่ทำงานบนเทคโนโลยีเว็บมาตรฐานอย่าง **HTML, DOM และ CSS** โดยตรง ทำให้การสร้างเว็บไซต์มีความเป็นธรรมชาติและสอดคล้องกับแพลตฟอร์มเว็บอย่างแท้จริง

Flutter Web

ทีมพัฒนา Flutter ได้ให้คำจำกัดความไว้อย่างชัดเจนว่า

Flutter Web ถูกออกแบบมาเพื่อสร้าง “Web App” ไม่ใช่ “Web Site”

กล่าวคือ Flutter Web เป็นเทคโนโลยีที่ยอดเยี่ยมสำหรับการขยายแอปพลิเคชันที่มีอยู่ให้สามารถทำงานบนเว็บได้ (Cross-platform) แต่ไม่ได้ถูกออกแบบมาเพื่อทดแทนเทคโนโลยีเว็บแบบดั้งเดิมทั้งหมด

เอกสารอย่างเป็นทางการของ Flutter ระบุว่า

ในปัจจุบัน Flutter ยังไม่เหมาะกับการใช้งาน HTML ทุกประเภท ตัวอย่างเช่น เนื้อหาที่เน้นข้อความเป็นหลัก จัดเรียงตามการไหลของเนื้อหา และเป็นเนื้อหาแบบคงที่อย่างบทความบล็อก จะทำงานได้ดีกว่าบนแนวคิดแบบเอกสารของเว็บโดยตรง มากกว่าการใช้แนวคิดแบบแอปที่เฟรมเวิร์ก UI อย่าง Flutter ออกแบบมาเพื่อรองรับ

ตัวอย่างการใช้งานจริง

เว็บไซต์ **dart.dev** และ **docs.flutter.dev** ซึ่งเป็นเว็บไซต์ทางการของ Dart และ Flutter ล้วนพัฒนาขึ้นด้วย **Jaspr** ทั้งสองเป็นตัวอย่างที่ดีของการใช้ Jaspr เพื่อสร้างเว็บไซต์ที่มีเนื้อหาเป็นข้อความจำนวนมาก (Content-heavy) และเป็นแบบสแตติกด้วยภาษา Dart