1. Record Nr. UNINA9910511471103321

Autore	Boduch Adam
Titolo	JavaScript at scale : build enduring JavaScript applications with scaling insights from the front-line of JavaScript development / / Adam Boduch
Pubbl/distr/stampa	Birmingham, [England] ; ; Mumbai, [India] : , : Packt Publishing, , 2015 ©2015
ISBN	1-78528-487-8
Descrizione fisica	1 online resource (267 p.)
Collana	Community Experience Distilled
Disciplina	005.2762
Soggetti	JavaScript (Computer program language) Electronic books.
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Includes index.
Nota di contenuto	Cover; Copyright; Credits; About the Author; About the Reviewers; www.PacktPub.com; Table of Contents; Preface; Chapter 1: Scale from a JavaScript Perspective; Scaling influencers; The need for scale; Growing user base; Building new features; Hiring more developers; Architectural perspectives; The browser is a unique environment; Component design; Component communication; Load time; Responsiveness; Addressability; Configurability; Making architectural trade-offs; Define your constants; Performance for ease of development; Configurability for performance; Performance for substitutability Ease of development for addressabilityMaintainability for performance; Less features for maintainability; Leveraging frameworks; Frameworks versus libraries; Implement patterns consistently; Performance is built in; Leverage community wisdom; Frameworks don't scale out-of-the-box ; Summary; Chapter 2: Influencers of Scale; Scaling users; License fees; Subscription fees; Consumption fees; Ad-supported; Open source; Communicating users; Support mechanisms; Feedback mechanisms; Notifying users; User metrics; Scaling users example; Scaling features; Application value Killer features versus features that killData-driven features; Competing with other products; Modifying existing features; Supporting user groups and roles; Introducing new services; Consuming real-time data; Scaling features example; Scaling development; Finding development

resources; Development responsibilities; Too many resources; Scaling development example; Influencer checklist; User checklist; What's the business model of our software?; Does our application have different user roles?; Do our users communicate with each other using our software?; How do we support our application?
How do we collect feedback from users?How do we notify users with relevant information?; What type of user metrics should we collect?; Feature checklist; What's the core value proposition of our software?; How do we determine the feasibility of a feature?; Can we make informed decisions about our features?; Who's our competition?; How do we make what we have better?; How do we integrate user management into our features?; Are our features tightly coupled to backend services?; How does the front-end stay synchronized with back-end data?; Developer checklist
How do we find the right development resources?How do we allocate development responsibilities?; Can we avoid hiring too many resources?; Summary; Chapter 3: Component Composition; Generic component types; Modules; Routers; Models/Collections; Controllers/Views; Templates; Application-specific components; Extending generic components; Identifying common data and functionality; Extending router components; Extending models/collections; Extending controllers/views; Mapping features to components; Generic features; Specific features; Decomposing components; Maintaining and debugging components
Re-factoring complex components

Sommario/riassunto

Have you ever come up against an application that felt like it was built on sand? Maybe you've been tasked with creating an application that needs to last longer than a year before a complete re-write? If so, JavaScript at Scale is your missing documentation for maintaining scalable architectures. There's no prerequisite framework knowledge required for this book, however, most concepts presented throughout are adaptations of components found in frameworks such as Backbone, AngularJS, or Ember. All code examples are presented using ECMAScript 6 syntax, to make sure your applications are ready