Entwicklung einer IDE unabhängigen Benutzeroberfläche für Saros

Matthias Bohnstedt

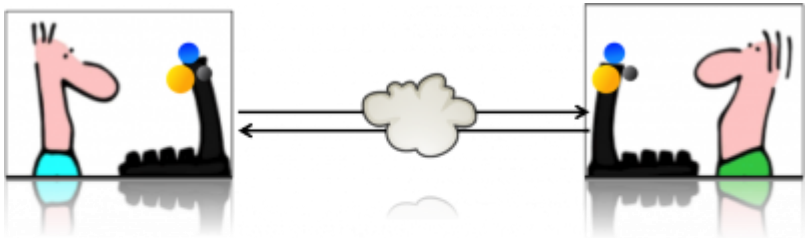
Betreuer: Franz Zieris

Eingereicht bei: Prof. Dr. Prechelt

Aufbau des Vortrags

- Einleitung
  - Motivation: Saros für IntelliJ / Eclipse
  - Problemstellung und Zielsetzung
- Vorgehen
- Ergebnisse
- Ausblick

Saros



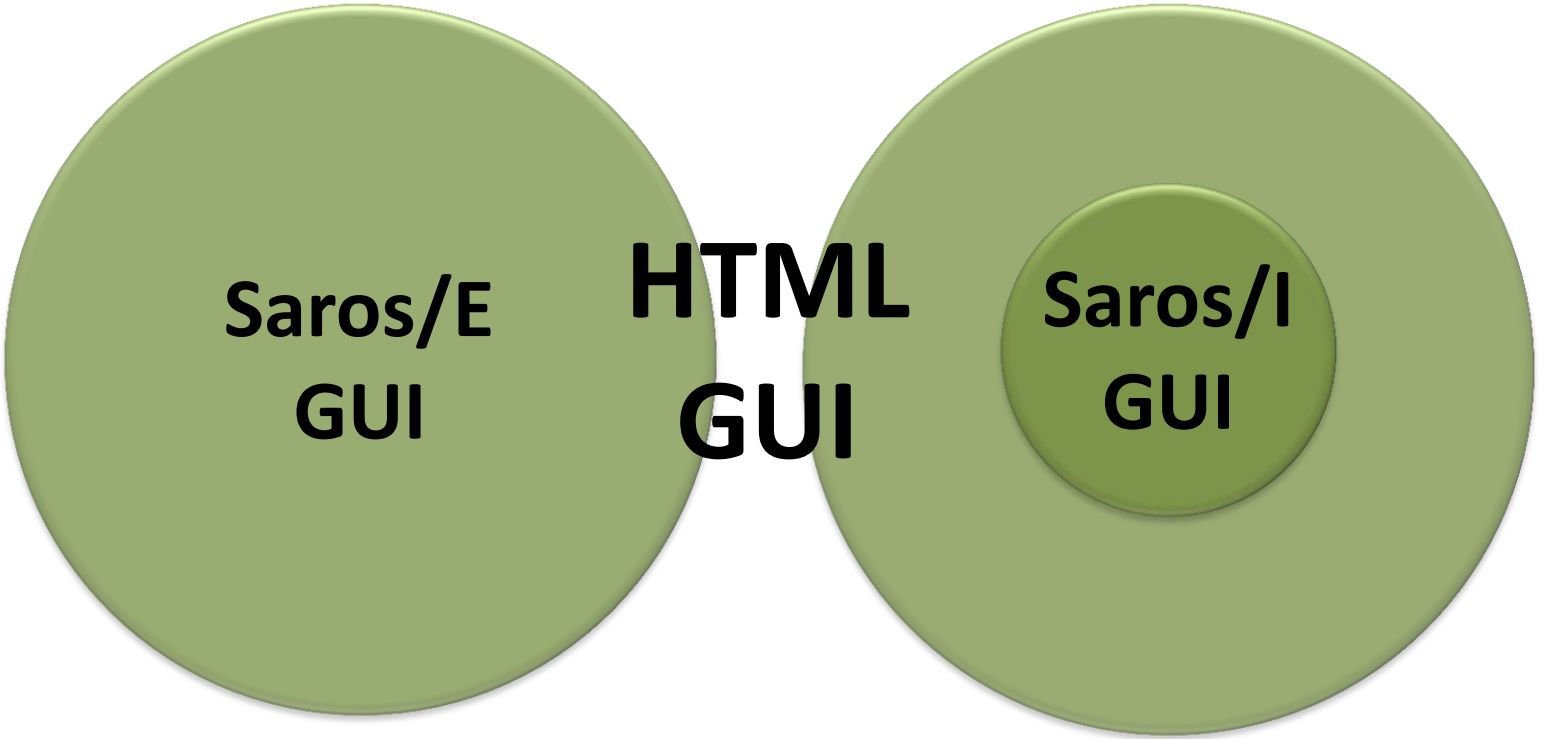
- **OS Eclipseplugin zur kollaborativen Softwareentwicklung**
  - Project Sharing
  - Zeitgleiches bearbeiten einer Datei
  - Syncronisierung des Workspaces zwischen Teilnehmern
  - Chat, Actionawareness, Whiteboard, …
  - XMPP basierte Kommunikation

- IntelliJ IDEA Version wird zur Zeit entwickelt (Beta)

Motivation und Herausforderung

- Entwicklung von Saro für andere IDEs wie IntelliJ erzeugt neue Herausforderungen.
  - Teilung in IDE spezifische / unspezifische (Kern-)Komponenten
  - Buildprozess
  - Testframework
  - Entwicklungsprozess
  - Guidelines
  - Dokumentation
  - …
- **Entwicklung IDE spezifischer GUI**

Ausgangsbasis der Arbeit



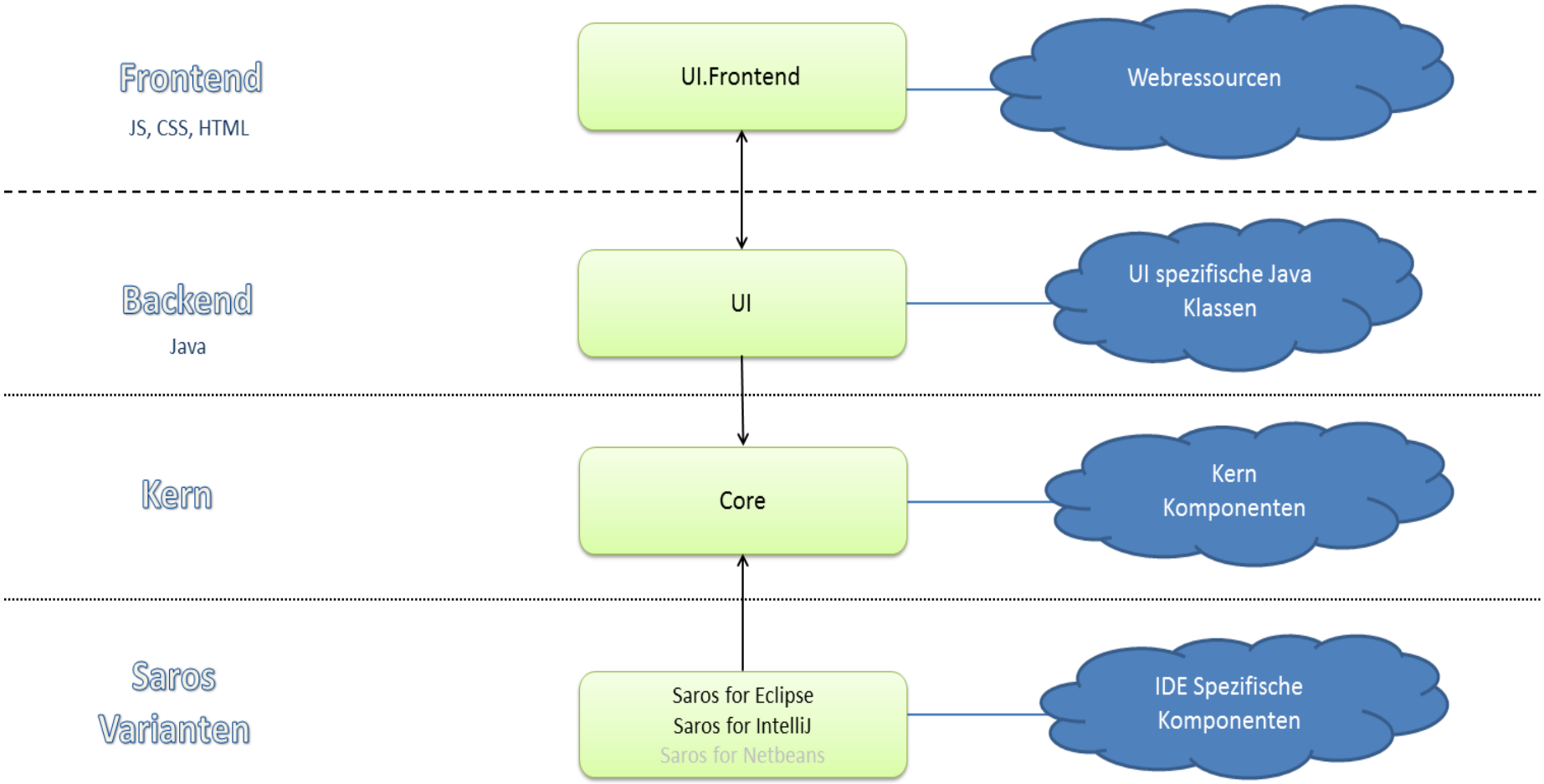
Vorgehen

- Definition von Randbedingungen für neue gemeinsame Oberfläche
  - Vermeidung von Codeduplikation
  - Schärferer Trennung von Anzeige und Geschäftslogik
  - Verbesserung der Codetransparenz
- Analyse welche Anzeigeelemente von Saros können unabhängig von der IDE realisiert werden.
  - Und welchen sollten!?
  - Definition für Kriterien für die Auswahl und Priorisierung
- Entwicklertest, in wie weit Randbedingungen erfüllt sind
  - Wo sind noch Probleme in der Architektur des Prototypen
- Umsetzung und Bereitstellung der Schnittstelle für Anzeigeelemente

Ergebnis
Analyse der Anzeigeelemente

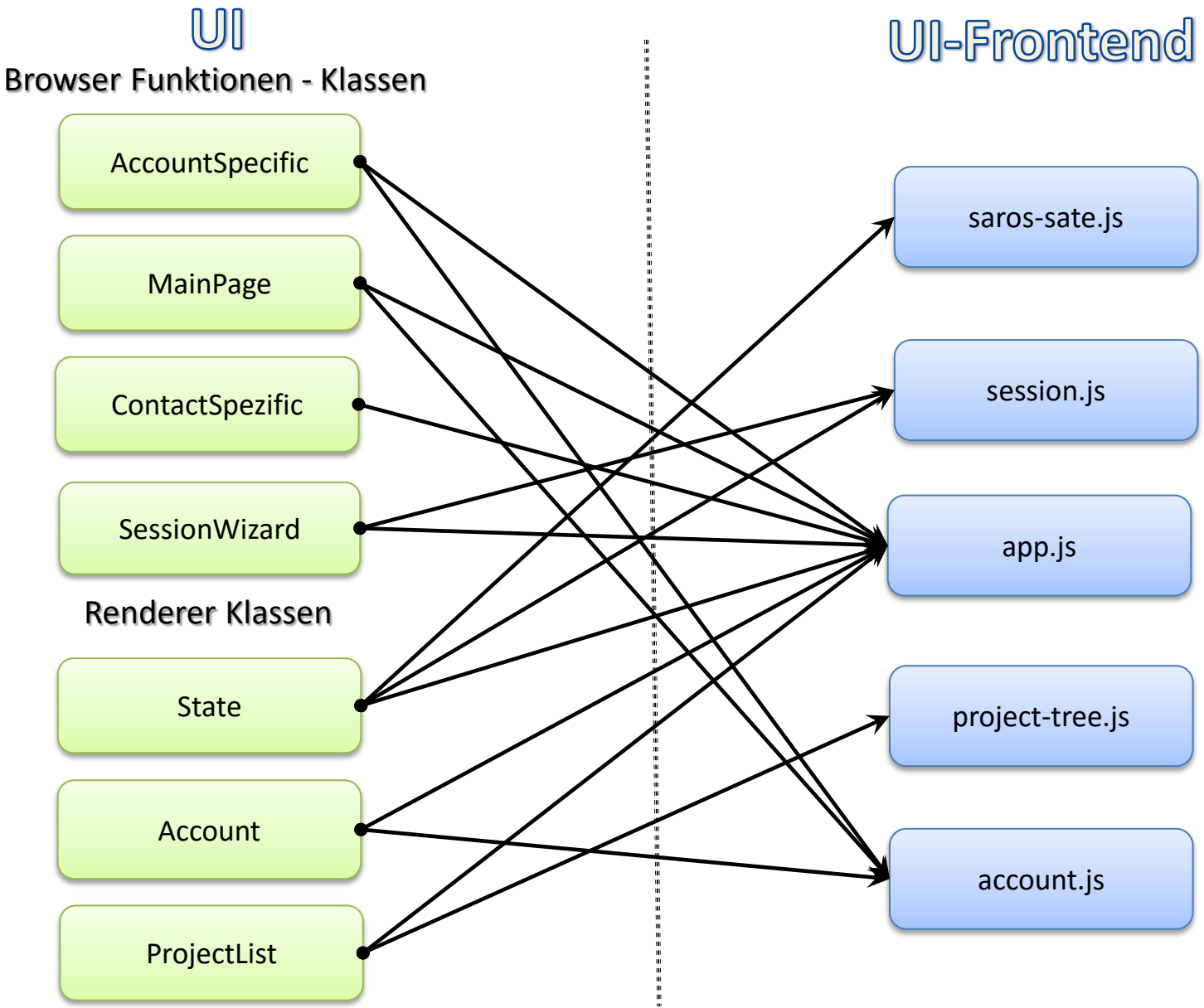
Anzeigeelement	Nutzen	Aufwand
Hauptansicht	**Groß**	**Hoch**
Session-Invitation-Wizard	**Groß**	**Hoch**
Join-Session-Wizard	Mittel	**Hoch**
Fortschrittsanzeige	Mittel	Mittel
Awarness-Informationen	Mittel (Nur Saros-E, Einschränkung in Anzeige)	Mittel bis Hoch
Einstellungsfenster	Gering bis kein (Benutzerfreundlichkeit)	Gering
Konfigurations-Wizard	Gering	Gering
Whitebord	Gering (Nur Saros-E)	Zu evaluieren
Chat	Gering (Nur Saros-E / kaum benutzt)	Zu entscheiden
Weiteres	Gering	Unterschiedlich

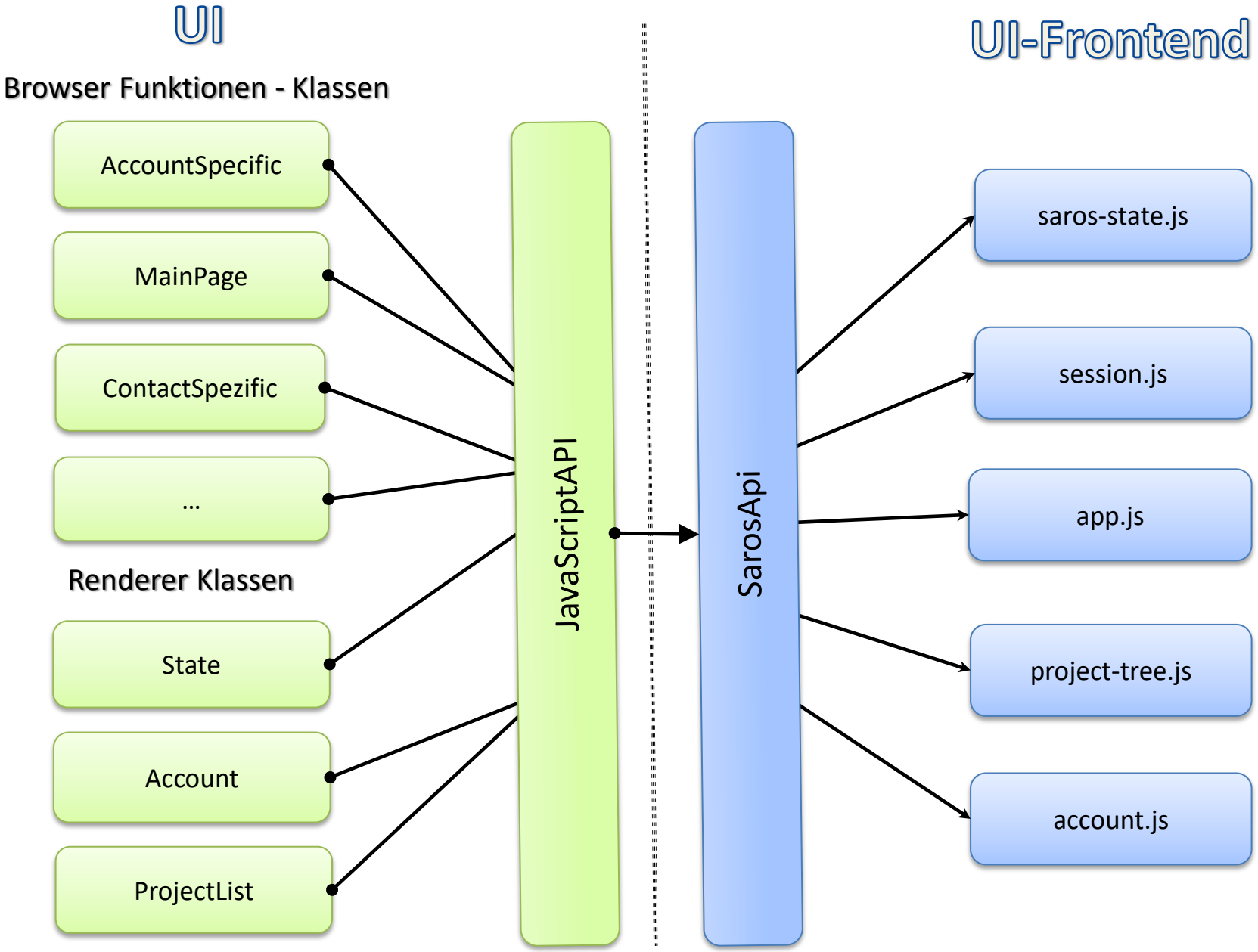
Aufbau der gemeinsamen Oberfläche



Wichtige Begriffe und Klassentypen

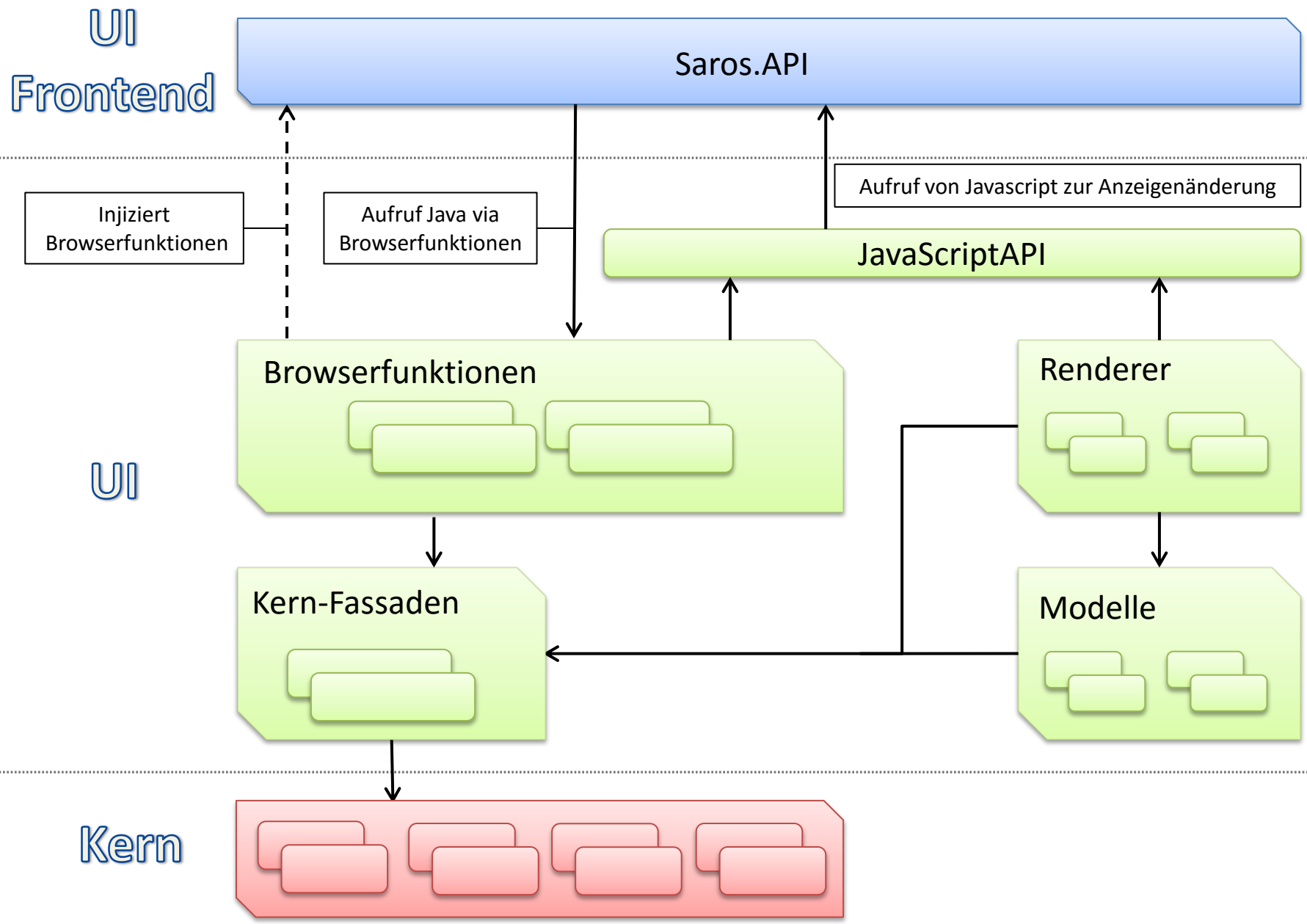
- Modelle
  - Serialisierbare Repräsentation des Zustands von Saros und vom Kern bereitgestellte Informationen
  - Z.B. Account, Kontakte, Zustand des Workspaces, …
- Browserfunktionen
  - Stellt die Schnittstelle für das Frontend dar, um auf Geschäftslogik von Saros zuzugreifen
  - Z.B. AddAccount(), showSessionWizard(), …
- Renderer
  - Aktualisiert das Frontend wenn sich Models ändern


UI
Browser Funktionen - Klassen
AccountSpecific
MainPage
ContactSpezific
SessionWizard
Renderer Klassen
State
Account
ProjectList
UI-Frontend
saros-sate.js
session.js
app.js
project-tree.js
account.js


UI
UI-Frontend
Browser Funktionen - Klassen
AccountSpecific
MainPage
ContactSpezific
...
Renderer Klassen
State
Account
ProjectList
JavaScriptAPI
SarosApi
saros-state.js
session.js
app.js
project-tree.js
account.js

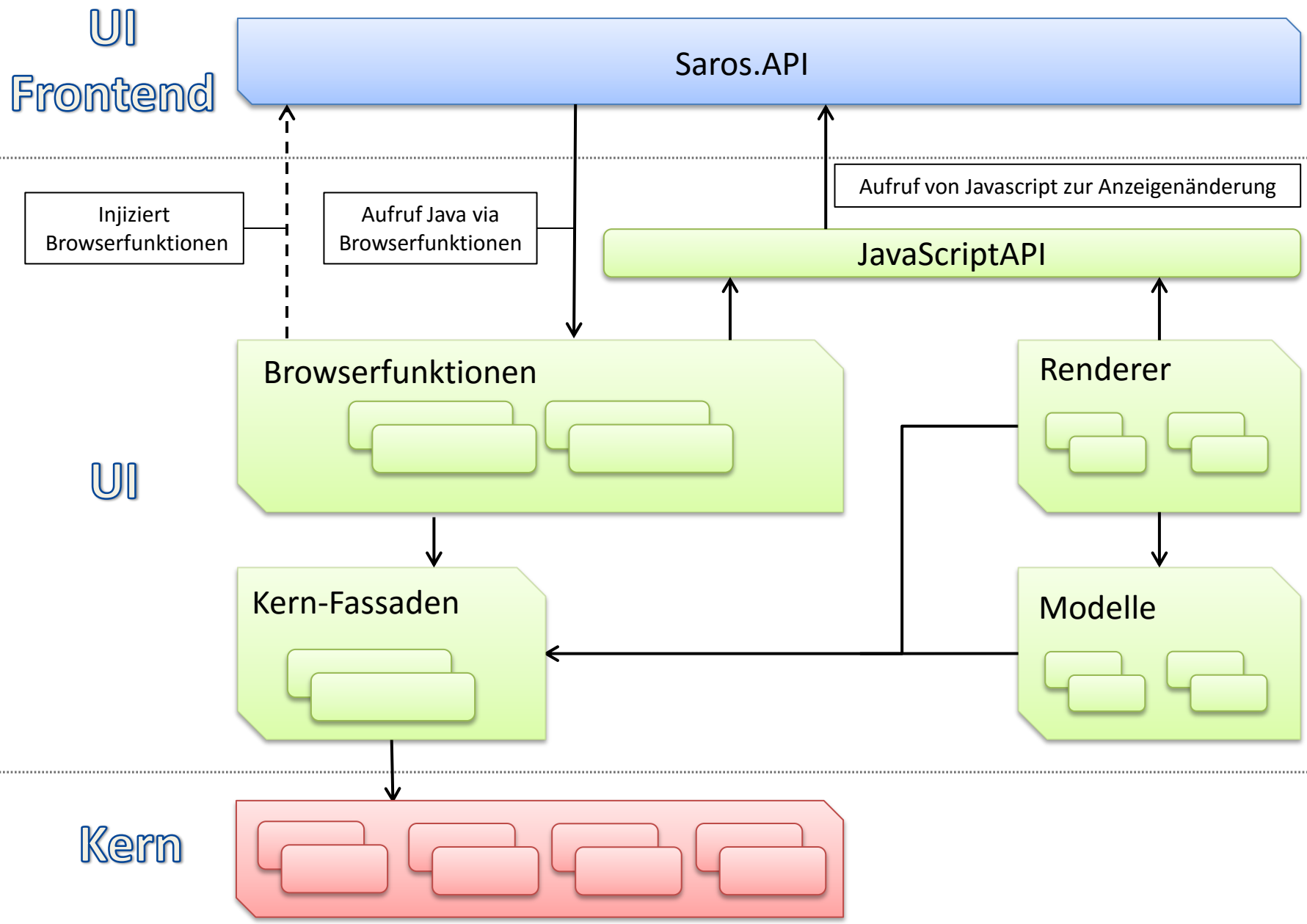
Neugestaltung der Browserfunktionen

- BFs waren anonym und nicht dokumentiert.
  - Frontendentwickler hatte keine Information darüber welche Funktionen verfügbar sind, über Aufgabe und Signatur der injizierten Funktionen
- BFs waren in Klassen gebündelt und an Dialoge gebunden.
  - Wiederverwendung von der selben Browserfunktionen in verschiedenen Dialogen praktisch nicht möglich. Hohe Gefahr von Dopplung von BFs


UI
Frontend
Saros.API
Injiziert
Browserfunktionen
Aufruf Java via
Browserfunktionen
Aufruf von Javascript zur Anzeigenänderung
JavaScriptAPI
UI
Browserfunktionen
Renderer
Kern-Fassaden
Modelle
Kern

Außerdem…

- Erweiterung des Kerns für Projektlist Datenmodel
- Entwicklung des Projektlist Models
- Prototyp für HTML basierten JTourBus
- Anpassungen an der Code-Convention
- Unittest für Modelerzeugung
- Ständige Abstimmung mit anderen laufenden Arbeiten.


UI
Frontend
Saros.API
Injiziert
Browserfunktionen
Aufruf Java via
Browserfunktionen
Aufruf von Javascript zur Anzeigenänderung
JavaScriptAPI
UI
Browserfunktionen
Renderer
Kern-Fassaden
Modelle
Kern

Ausblick

- Vervollständigung der Schnittstelle
  - Implementieren fehlender Browserfunktionen
  - Implementieren fehlender Datenmodel
    - IDE unabhängige Einstellungen
    - Awareness Informationen
    - Progression Monitor
  - Erweitern der Javascript API
    - Senden von Awareness informationen ans Frontend
  - Fassaden für Kern
    - Negotiationhandler

Ausblick II

- Testframework für gemeinsame UI
  - STF für HTML
- Erstellen der Dialoge und Anzeigen im HTML Frontend
- Beheben bekannter Probleme

Danke!

Fragen!?