

# Systemutveckling

## Ramverk

**25 HP**

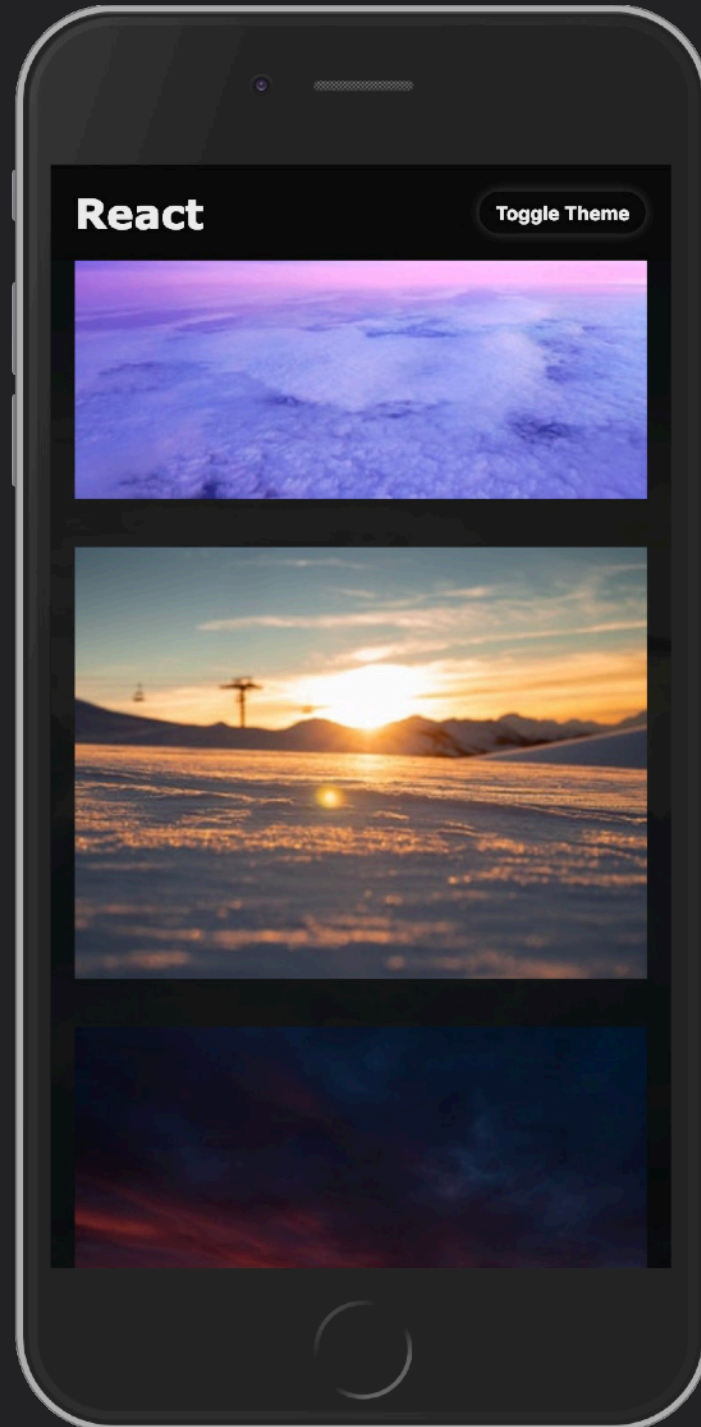
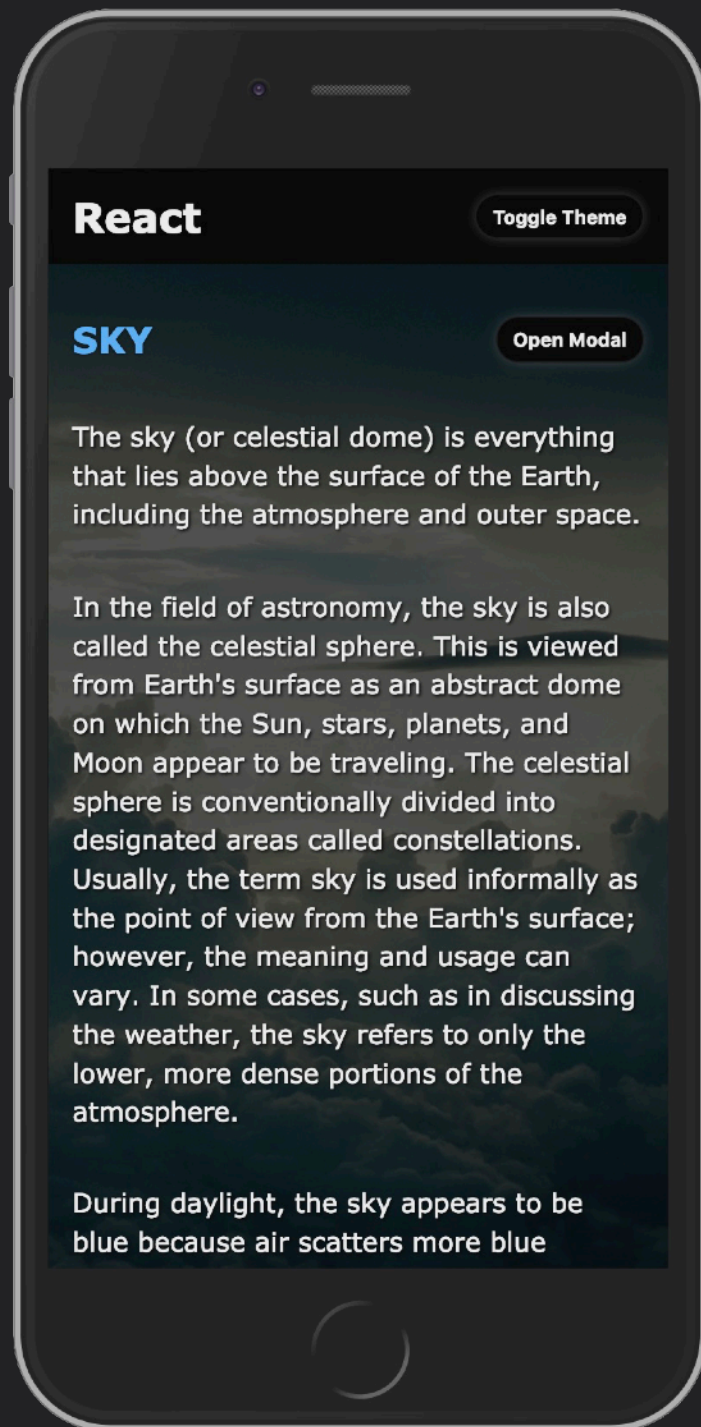
# Kursupplägget

Föreläsning			Innehåll (FM)	Övningar (EM)	Inlämning
1	mån 25 mars	D	Introduktion Typescript (Basic Types, Type Inference, Variable Declarations, Iterators and Generators)	TypeScripts Hemsida TypeScript in 5 minutes + övning	
2	ons 27 mars	V	Typescript forts. (Functions, Classes, Interfaces, Modules)		
3	fre 29 mars	V	Introduktion React (SPA, Virtuellt DOM, Kap. 1-6)	React's Hemsida Main Concepts inklusive övningar i CodePen Kodövning (RP)	
4	mån 1 apr.	D	React Playground (1-initial-setup + 2-layout)		Inlämning 1 ges ut
5	ons 3 apr.	V	React forts. (Kap. 7-12)		
6	fre 5 apr.	V	Create React App TS Intro	Övning "Todo App"	Handledning
7	tis 9 apr.	D	React Playground (3-navigation-with-state) React Playground (4-navigation-with-routes)	Kodövning (RP)	
8	ons 10 apr.	V	React Playground (5-code-splitting) React Playground (6-error-boundary)	Kodövning (RP)	
9	fre 12 apr.	V	React Playground (7-portals)	Muntlig Presentation Kodövning (RP)	Inlämning 1 in Inlämning 2 ut
10	mån 15 apr.	D	React Playground (8-code-split-app-start)	Kodövning (RP)	
11	tis 16 apr.	D	React Playground (9-api-lib-axios)	Kodövning (RP)	Handledning
12	tis 23 apr.	D	React Playground (10-context)	Kodövning (RP)	
13	tors 25 apr.	V	Tentaplugg		Inlämning 2 lämnas in
14	fre 26 apr.	V	TENTAMEN		

# React Playground

Projektet vi kommer jobba med under kursen

# React Playground














## Branches

- 🔗 master
- 🔗 1-initial-setup
- 🔗 2-layout
- 🔗 3-navigation-with-state
- 🔗 4-navigation-with-routes
- 🔗 5-code-splitting
- 🔗 6-error-boundary
- 🔗 7-portals
- 🔗 8-code-split-app-start
- 🔗 9-api-lib-axios
- 🔗 10-react-context

# React Playground

## Branches

-  master
-  1-initial-setup
-  2-layout
-  3-navigation-with-state
-  4-navigation-with-routes
-  5-code-splitting
-  6-error-boundary
-  7-portals
-  8-code-split-app-start
-  9-api-lib-axios
-  10-react-context

## Innan vi börjar

**1. Starta det ni gjorde i förra veckan och se så att allt fungerar.**

**- npm run app**

**2. Hämta 4-navigation-with-routes koden från #slack och utgå ifrån den.**

**- npm install**

**- npm run app**

# Code Splitting

En snabbare initial laddning av sidan

# Code Splitting - Why?

**Ladda in sidans innehåll dynamisk i webbläsaren.**

**Bättre upplevelse för användare med sämre uppkoppling.**

**Snabbare "PageLoad" (ladda endast in den kod som behövs).**

# Code Splitting - When?

**När delar av sidan bara används av specifika användare ex,**  
en Dashboard.

**När man byter sida ex:**

Listvy — Detaljvy

Produktsida — annan Produktsida



# Code Splitting - How?

**Genom att skapa flera bundle-filer som innehåller olika kod för delar av webbsidan.**

**Webpack eller Browserify används ofta för detta.**

**Utan codesplitting byggs endast en bundle-fil ihop för webbläsaren.**

**Codesplitting möjliggör att det kan finnas flera bundle-filer, med olika innehåll, som kan läsas in på request.**

# Code Splitting - React.Lazy & Suspense

Komponenten "Suspense" tillåter en temporär rendering innan en bundle har laddat klart.

Vanligtvis brukar detta vara "Spinners/Loaders".

Denna temporära rendering implementeras med hjälp utan attributet "fallback" på komponenten <Suspense>.

```
const OtherComponent = React.lazy(() => import('./OtherComponent'));

function MyComponent() {
  return (
    <div>
      <Suspense fallback={<div>Loading...</div>}> ←
        <OtherComponent />
      </Suspense>
    </div>
  );
}
```

Innehållet i fallback kommer renderas då "OtherComponent" inte hunnit laddas in.

# React 17 and Beyond

**I React 17 kommer asynkronisk rendering (Concurrent Rendering), vilket innebär att react kan pausa renderingen till fördel för andra viktigare saker som event's eller i väntan på bundles.**

**Använder sig bland annat av suspense komponenten.**

**Suspense kommer även gå att användas ihop med bilder och JSON-data för att skapa en bättre användarupplevelse.**

**Just nu finns det ett beta bibliotek (react-cache) för att cacha JSON-datan så den inte behöver hämtas på nytt om användaren går tillbaka till en sidan hen redan varit på.**

# Läsanvisningar

## React Today and Tomorrow (video)

<https://www.youtube.com/watch?v=V-QO-KO90iQ>

## Concurrent Rendering (video)

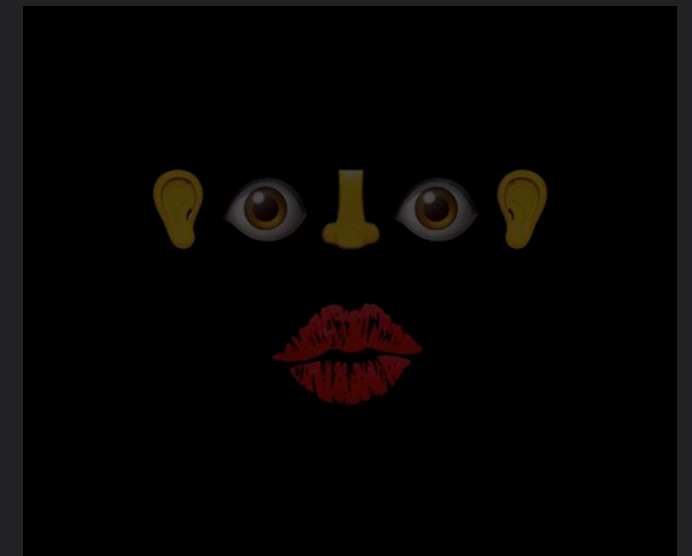
<https://www.youtube.com/watch?v=ByBPyMBTzM0>

# Dagens övning - del 1

Använd er av **Suspense** för att skapa en "splash screen" som alltid laddas in och visas initialt.

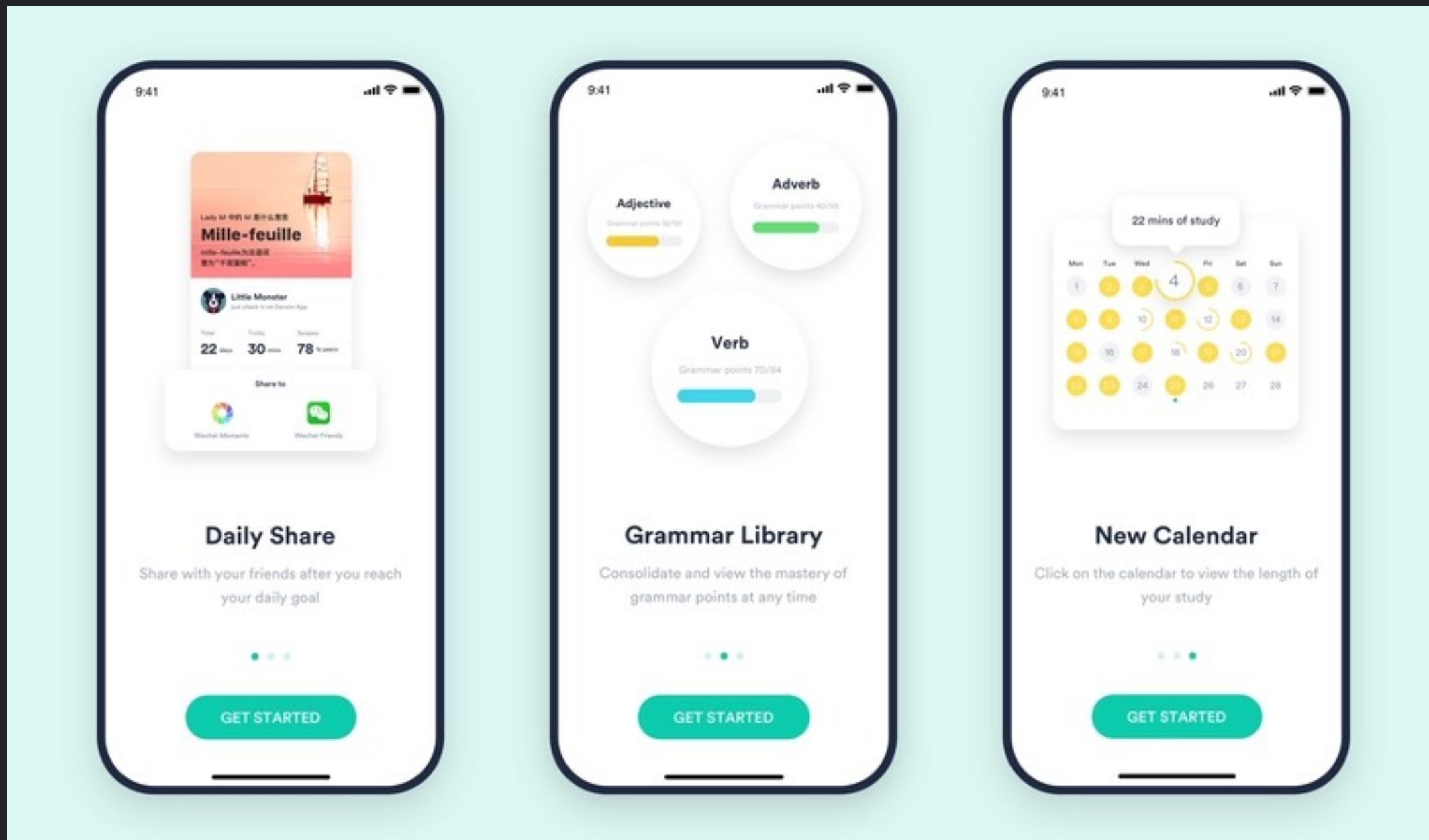
Testa i Chrome att begränsa nätverkshastigheten till 3g

```
render () {  
  return (  
    <Suspense fallback=<Spinner/>>  
      <Router>  
        <ErrorBoundary>  
          <Suspense fallback=<Monkey/>>  
            <Layout/>  
          </Suspense>  
        </ErrorBoundary>  
      </Router>  
    </Suspense>  
  )  
}
```



# Dagens övning - del 2

Skapa en välkomstkärm som kan visas när vi vill presentera nya features för våra användare.



# Dagens övning - del 2

Skapa en välkomstkärm som kan visas när vi vill presentera nya features för våra användare.

Använd Modalen och även state i `<App/>` för att visa/dölja välkomstkärmen. Skapa vilket innehåll du vill, dock måste det finnas en knapp för så användaren kan hoppa över introduktionen.

```
render () {  
  return (  
    <Suspense fallback={<Spinner/>}>  
      <Router>  
        <ErrorBoundary>  
          {this.WelcomeScreen}  
          <Suspense fallback={<Monkey/>}>  
            <Layout/>  
          </Suspense>  
        </ErrorBoundary>  
      </Router>  
    </Suspense>  
  )  
}
```

# Nästa lektion

React Playground

(Axios & Detaljvyn)



**Tack**