

# **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, Virtuell DOM, Kap. 1-6)	Reacts Hemsida Main Concepts inklusive övningar i CodePen Kodövning (RP)	Inlämning 1 ges ut
4	mån 1 apr.	D	React Playground (1-initial-setup + 2-layout)		
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	tors 18 apr.	V	React Playground (10-context)	Kodövning (RP)	
13	tis 23 apr.	D	Algoritmer och välja Ramverk	Jobba med inlämning	Handledning
14	tors 25 apr.	V	Tentaplugg		Inlämning 2 lämnas in
15	fre 26 apr.	V	TENTAMEN		

# Axios

HTTP Request Library

# Axios - Intro

Ett litet bibliotet för att enkelt kunna göra HTTP Requests från klienter.

Det är baserad på Promises istället för Callbacks 

Det finns Typescript typings för biblioteket 

Ett alternativ till Fetch eller Jquery's AJAX.

**Varför ett sätt till?**

Fetch fungerar lite olika i olika webbläsare.

AJAX kommer med Jquery som vi inte vill installera i ett React projekt - varför?

# Axios - Intro

## Browser Support

					
Latest ✓	Latest ✓	Latest ✓	Latest ✓	Latest ✓	11 ✓

 Firefox	 Chrome	 IE	 Edge	 Safari
65  ✓	72  ✓	11  ✓	18  ✓	9  ✓
				10  ✓
				11  ✓

# Axios - Hur används det?

Kolla in deras GitHub repo: <https://github.com/axios/axios>

Där hittar ni **installationsinstruktioner** (npm i axios)

Samt dokumentation på hur det används:

```
// Want to use async/await? Add the `async` keyword to your outer function/method.  
async function getUser() {  
  try {  
    const response = await axios.get('/user?ID=12345');  
    console.log(response);  
  } catch (error) {  
    console.error(error);  
  }  
}
```

# Axios - Hur används det?

Om man hellre vill använda Promises

```
// Make a request for a user with a given ID
axios.get('/user?ID=12345')
  .then(function (response) {
    // handle success
    console.log(response);
  })
  .catch(function (error) {
    // handle error
    console.log(error);
  })
  .then(function () {
    // always executed
 });
```

# Axios - Hur används det?

Om man hellre vill använda Promises

```
// Optionally the request above could also be done as
axios.get('/user', {
  params: {
    ID: 12345
  }
})
.then(function (response) {
  console.log(response);
})
.catch(function (error) {
  console.log(error);
})
.then(function () {
  // always executed
});
```

Man kan även  
skicka url  
parametrarna i ett  
objekt istället.

# Axios - Hur används det?

## Tillgängliga metoder

`axios.request(config)`

`axios.get(url[, config])`

`axios.delete(url[, config])`

`axios.head(url[, config])`

`axios.options(url[, config])`

`axios.post(url[, data[, config]])`

`axios.put(url[, data[, config]])`

`axios.patch(url[, data[, config]])`

**Om man vill exikvera flear  
anrop samtidigt!**

### Concurrency

Helper functions for dealing with concurrent requests.

`axios.all(iterable)`

# Läsanvisningar

## Axios Docs

<https://github.com/axios/axios>

## React Router

<https://reacttraining.com/react-router/web/guides/quick-start>

## React Docs

<https://reactjs.org/docs/getting-started.html>

# Dagens övning

Bygga ut detaljsidan

# Övning - del 1

**Dela upp detalj-sidan i tre delar genom att skapa tre nya sektionskomponenter [HeaderSection, TextSection, ImageSection].**

# Övning - del 1

## React Playground

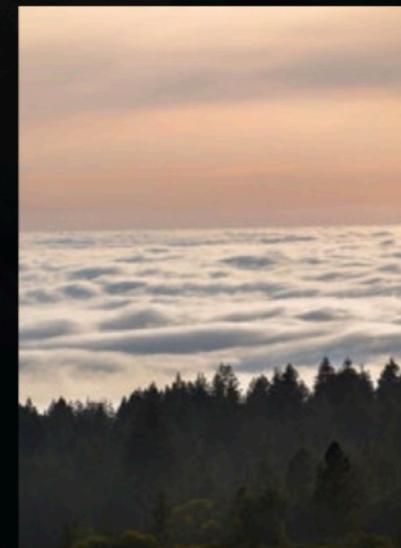
**SKY**

[Open Modal](#)

The sky (or celestial dome) is everything that lies above the surface of the Earth, including the atmosphere and outer space.

In the field of astronomy, the sky is also called the celestial sphere. This is viewed from Earth's surface as an abstract dome on which the Sun, stars, planets, and Moon appear to be traveling. The celestial sphere is conventionally divided into designated areas called constellations. Usually, the term sky is used informally as the point of view from the Earth's surface; however, the meaning and usage can vary. In some cases, such as in discussing the weather, the sky refers to only the lower, more dense portions of the atmosphere.

During daylight, the sky appears to be blue because air scatters more blue sunlight than red. At night, the sky appears to be a mostly dark surface or region spangled with stars. During the day, the Sun can be seen in the sky unless obscured by clouds. In the night sky (and to some extent during the day) the Moon, planets and stars are visible in the sky. Some of the natural phenomena seen in the sky are clouds, rainbows, and aurorae. Lightning and precipitation can also be seen in the sky during storms. Birds, insects, aircraft, and kites are often considered to fly in the sky. Due to human activities, smog during the day and light pollution during the night are often seen above large cities.



# Övning - del 1

**Dela upp detalj-sidan i tre delar genom att skapa tre nya sektionskomponenter [HeaderSection, TextSection, ImageSection].**

**Lägg till en titel text i HeaderSection samt flytta modal-knappen hit och placera den längst ut till höger.**

**Kopiera en text från Wikipedia och lägg till den i TextSection.**

**Samt skapa upp några divar i ImageSection och få till storlek och layout med flexbox.**

**Lägg till overflow scroll/auto på DetailView kcontainern så det går att skrolla på sidan.**

- ▲ detailView
- ⚛ detailView.tsx
- ⚛ headerSection.tsx
- ⚛ imageSection.tsx
- ⚛ textSection.tsx

# Övning - del 2

Hämta bilderna från [unsplash.com](https://unsplash.com) - via deras API!

VAD SOM MÅSTE GÖRAS FÖRST:

Skapa ett konto på Unsplash och skapa sedan en "app" i deras UI.  
Där hittar ni sedan eran **AccessKey** som måste skickas med i varje  
anrop för att hämta bilder.

Kom ihåg att även installera **axios** (npm i axios).

# Övning - del 2

**Utgå ifrån ImageSection komponenten!**

I componentDidMount anropar ni unsplash API'et m.h.a axios för att hämta ett gäng bilder, tex 24 stycken.

**Svaret ni får tillbaka är lämpligt att först logga ut för att kunna se innehållet. Mappa därefter URL'erna i svaret för varje bild till en array som ni sparar i statet.**

Om ni därefter lägger en console.log(this.state.images) i render metoden skall ni kunna se att statet har uppdaterats och att render metod körs två gånger - andra gången med url'en till bilderna.

**Skapa sedan en ny komponent ImageCard med props (url) som renderar ut bilden den får in som props.**

**Mappa sedan alla url'ena sparade i statet till <ImageCard url={...}/> komponenter.**

# Övning - del 3

**Visa bilden i full storlek när användaren klickar på en bild.**

**För att det här skall fungera bra måste ImageCard få tillgång till en mer högupplöst bild som den kan visa.**

**Använd modalen för det här visa bilden på hela skärmen.**

# Nästa lektion

React Playground

(React Context)

Tack