

A decorative graphic on the left side of the slide. It consists of a blue parallelogram and a light green parallelogram, both tilted at an angle. The blue shape is in the foreground, and the green shape is partially behind it. They are set against a dark blue background with faint, lighter blue diagonal stripes.

Introduktion till React/webbframverk

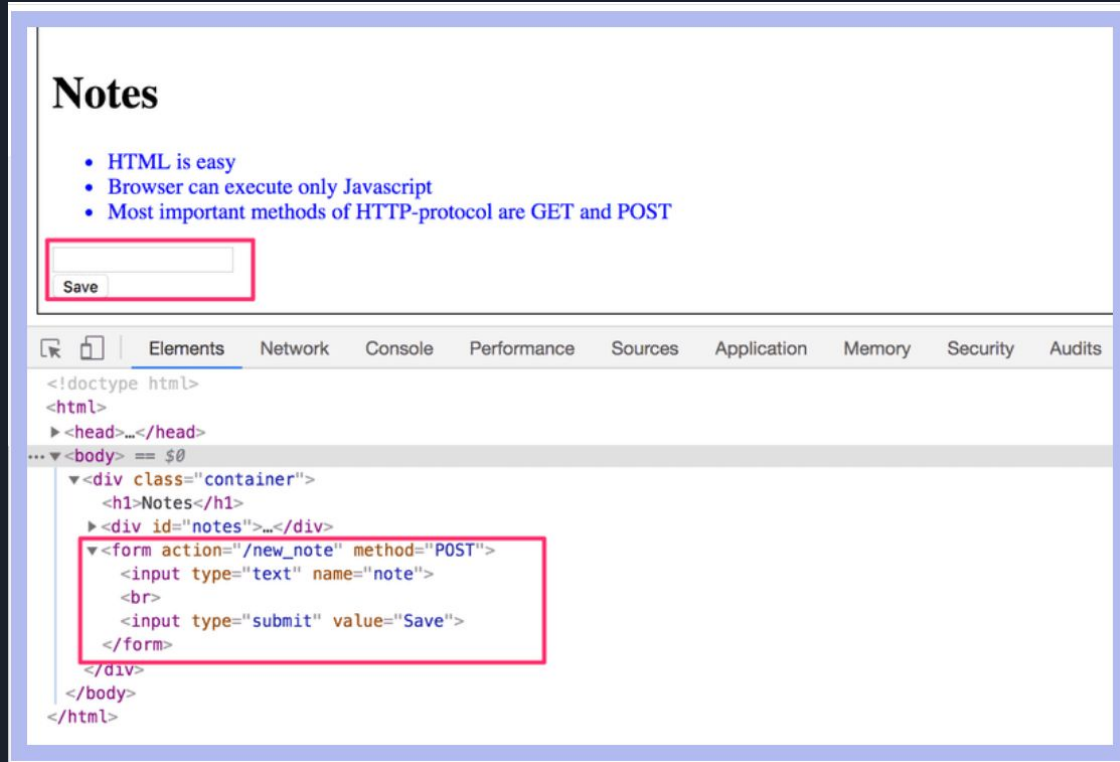


Agenda

- Forms och POST request
- AJAX och SPA
- JavaScript Bibliotek
- Introduktion till React
- Övningar

Forms och POST request

Formulär



The screenshot displays a web browser window with a page titled "Notes". The page content includes a bulleted list and a form. The list contains the following items:

- HTML is easy
- Browser can execute only Javascript
- Most important methods of HTTP-protocol are GET and POST

Below the list is a form with a text input field and a "Save" button. A red rectangle highlights the form elements in the browser view.

The browser's developer tools are open, showing the "Elements" panel. The HTML structure of the page is displayed, with the form's code highlighted by a red rectangle:

```
<!doctype html>
<html>
  <head>...</head>
  <body> == $0
    <div class="container">
      <h1>Notes</h1>
      <div id="notes">...</div>
      <form action="/new_note" method="POST">
        <input type="text" name="note">
        <br>
        <input type="submit" value="Save">
      </form>
    </div>
  </body>
</html>
```

Forms och POST request

Nätverks request

• form data is sent with HTTP POST

Save

Elements Network Console Performance Sources Application Memory Security Audits

View: [Icons] [Icons] Group by frame Preserve log [x] Disable cache [x] Offline No throttling

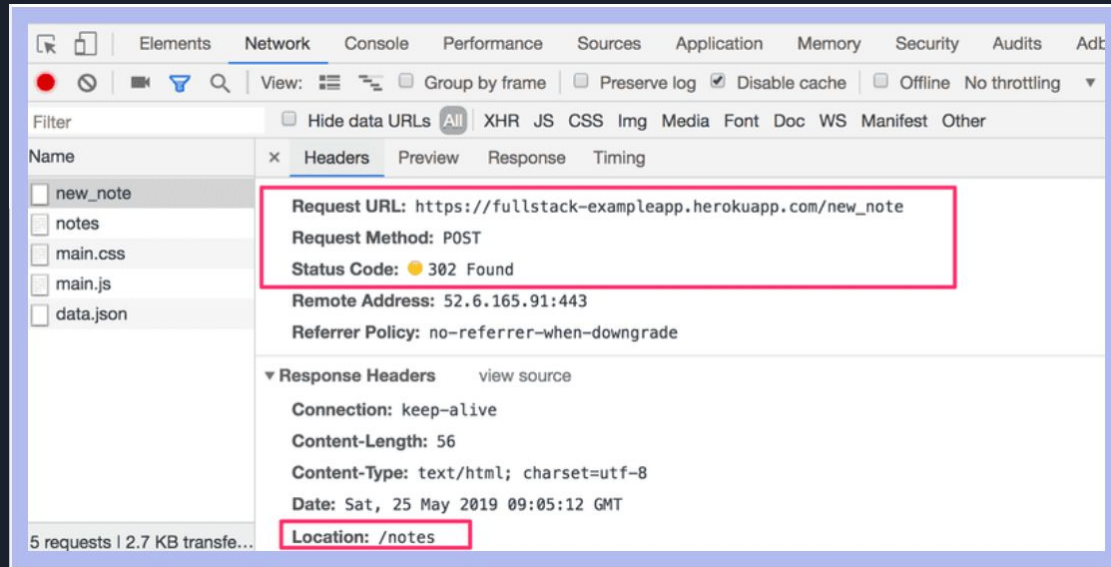
Filter [x] Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

Name	Status	Type	Initiator	Size
<input type="checkbox"/> new_note	302	text/html	Other	20
<input type="checkbox"/> notes	200	document	new_note	64
<input type="checkbox"/> main.css	200	stylesheet	notes	40
<input type="checkbox"/> main.js	200	script	notes	90
<input type="checkbox"/> data.json	200	xhr	main.js:23	57

Forms och POST request

Om vi kikar på requesten

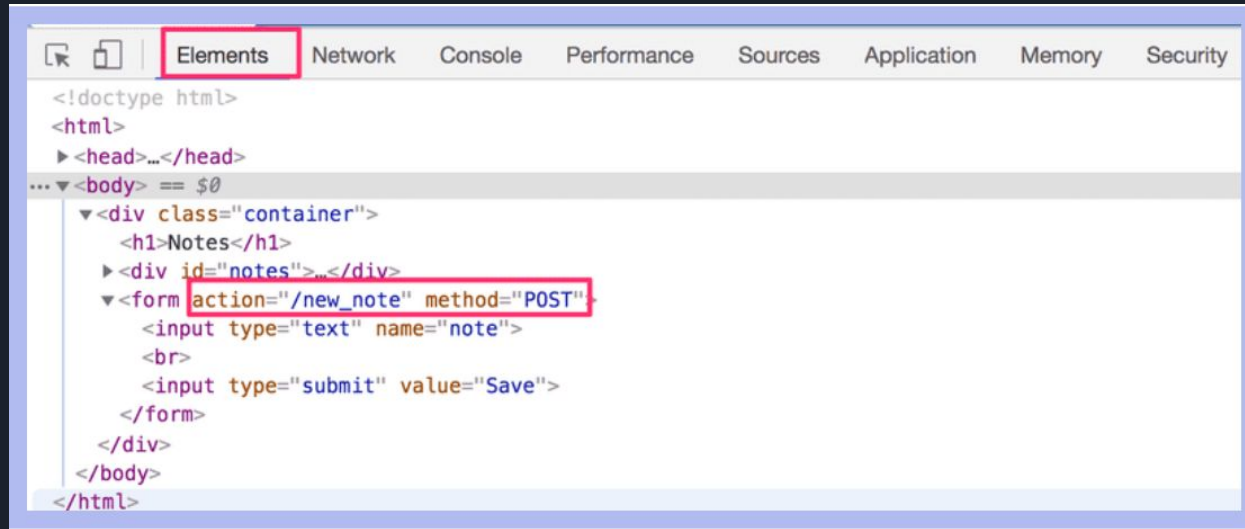
- Response 302 (redirect)
- Återvänder till /notes
- Nu med den nya noten



Forms och POST request

HTML Formulär har

- action attribut
- method attribut



```
<!doctype html>
<html>
  <head>...</head>
  <body> == $0
    <div class="container">
      <h1>Notes</h1>
      <div id="notes">...</div>
      <form action="/new_note" method="POST">
        <input type="text" name="note">
        <br>
        <input type="submit" value="Save">
      </form>
    </div>
  </body>
</html>
```



Forms och POST request

Servern behöver kunna svara på denna request

- req.body (innehåll)
- res.redirect
- Ingen databas (lista)

```
app.post('/new_note', (req, res) => {  
  notes.push({  
    content: req.body.note,  
    date: new Date(),  
  })  
  
  return res.redirect('/notes')  
})
```



Forms och POST request

Demo: Lägg till POST request hantering



AJAX och SPA

Detta är en ny stil på webbapplikationer 2005 (ny webbläsarteknologi)

- **AJAX** (Asynchronous JavaScript And XML)
- Ladda en JavaScript fil inbäddad i HTML och
 - En url för JavaScript (/)
 - En url för att hämta JSON (/data.json)
 - En url för att skapa ny note (/new_note)
- Hur skulle det se ut om det var REST?



AJAX och SPA

REST

- huvudsida med HTML, JavaScript, CSS (/)
- hämta notes (GET /notes)
- skapa notes (POST /notes)

Detta är standard idag och man talar sällan om att det är AJAX längre

SPA (Single Page Applications)

AJAX och SPA

Skillnad på traditionell [webbapplikation](#) och [SPA](#)





JavaScript Bibliotek

Applikationen vi såg var gjord med Vanilla JavaScript

- Använder DOM-API och JavaScript för att strukturera applikation

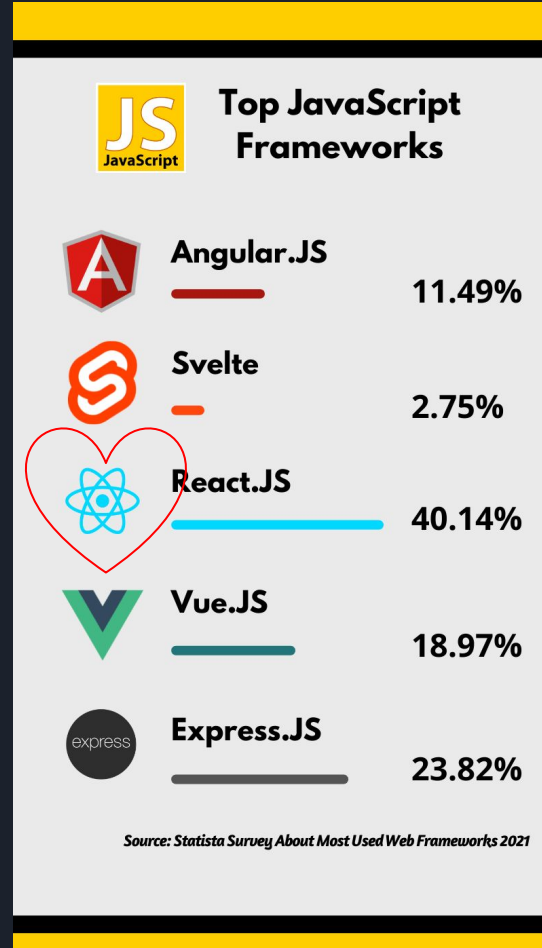
Det finns bibliotek som kan hjälpa

- JQuery (<https://jquery.com/>)
 - exempel på ett gammalt bibliotek med hjälpfunktioner
 - Populärt eftersom det stödde alla webbläsare (idag är detta mindre problem)
- Backbone (<https://backbonejs.org/>)
 - 2010 första ramverk att se ljuset
 - Använde MVC (Model View Controller) för struktur av hela applikation
- Angular (<https://angularjs.org/>)
 - 2012 Blev standard ett tag
 - Använder fejk HTML som kallas komponenter
 - 2014 release av icke-bakåtkompatibel version 2 (stoppade segertåget)

JavaScript Bibliotek

Från 2021

- JavaScript fatigue
 - När man blir trött
- Vi ska fokusera på React





JavaScript Bibliotek

Fullstack web development?

- Buzzword
- En vanlig applikation ses som bestå av 3 lager
 - Frontend (Klient)
 - Backend (Server)
 - Databas (MongoDB)
- En fullstack utvecklare har kompetens
 - Server teknologi (Node)
 - JavaScript
 - Frontend Ramverk (React)
 - Ofta lite cloud (AWS)
 - Lite konfigurerering och administration
- En fullstackutvecklare kan vara bred eller specialiserad



Introduktion till React

Skapa en applikation som heter part1 mha Vite

```
# npm 6.x (outdated, but still used by some):  
npm create vite@latest part1 --template react  
  
# npm 7+, extra double-dash is needed:  
npm create vite@latest part1 -- --template react
```



Introduktion till React

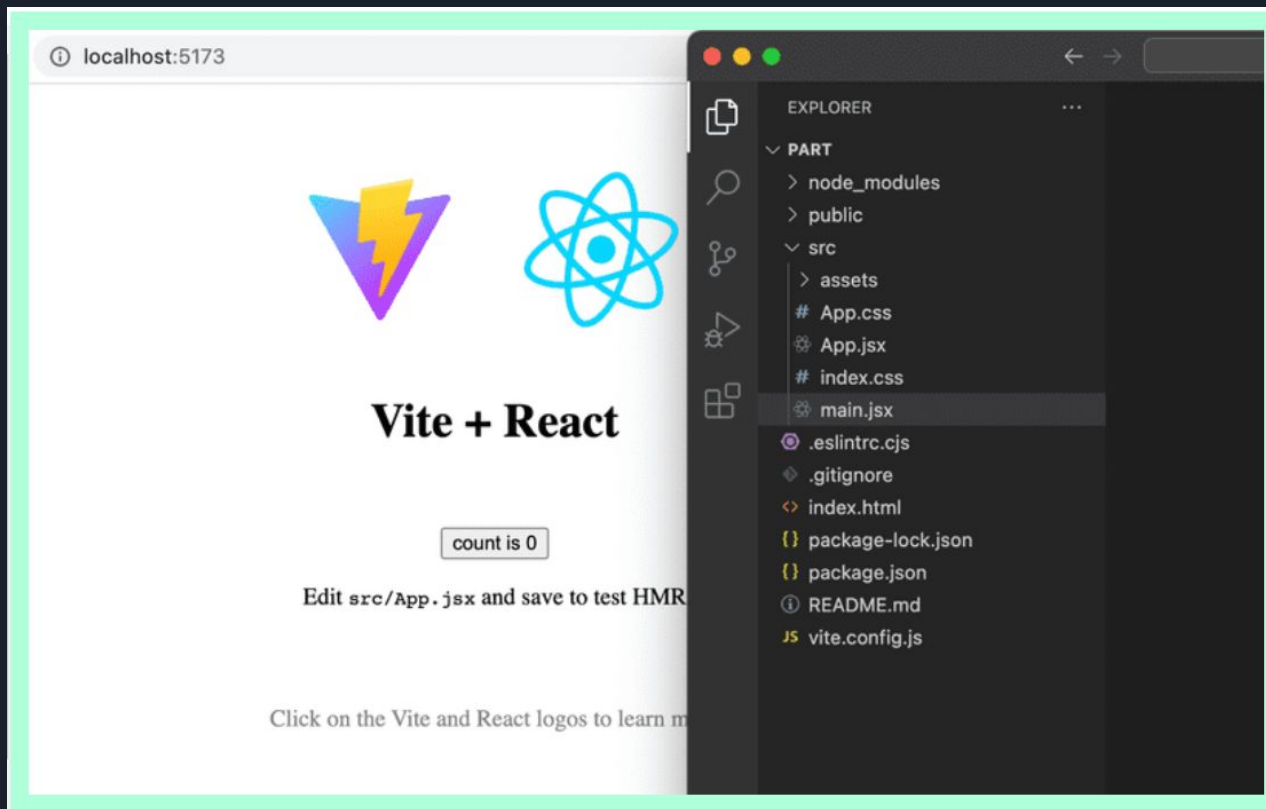
Nu har vi ett byggställning (scaffolding) runt en applikation

```
cd part1  
npm install
```

Vi startar applikation med

```
npm run dev
```


Introduktion till React





Introduktion till React

Create-react-app är äldre alternativ till Vite

Då startar applikation med

```
npm start
```

istället för Vites

```
npm run dev
```



Introduktion till React

Vi ska titta lite på React

- JSX
- ReactDOM.createRoot
- Komponent
- Samarbetande komponenter
- Props



Introduktion till React

Övningar

- Skapa en ny applikation med Vite/create-react-app
 - Ändra så att den säger hej <ditt namn>



Introduktion till React

Övningar

- Fyll i komponenter så att du får en full hemsida
 - Header
 - Content
 - Footer

```
const App = () => {  
  // const-definitions  
  
  return (  
    <div>  
      <Header course={course} />  
      <Content ... />  
      <Total ... />  
    </div>  
  )  
}
```



Introduktion till React

Övningar

- Skapa Content med en komponent Part med props
 - Name
 - Duration
 - Starttime
 - Endtime

```
const Content = ... {  
  return (  
    <div>  
      <Part .../>  
      <Part .../>  
      <Part .../>  
    </div>  
  )  
}
```