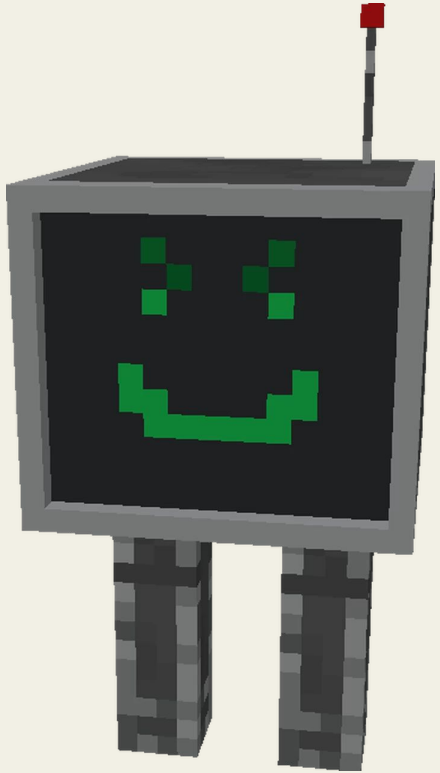

# Kurs i Mineflayer - 04.03.25

# Hva er Mineflayer?

- Mineflayer er et kraftig JavaScript-bibliotek som lar deg lage **programmerbare Minecraft-bots**!



# Hvordan fungerer Mineflayer?

```
const mineflayer = require('mineflayer')

const bot = mineflayer.createBot({
  host: 'localhost', // Server address
  port: 25565, // Server port
  username: 'MyBot', // Bot username
  auth: 'offline' // Auth type
})

bot.on('spawn', () => {
  bot.chat('Hello world!')
})
```

# Sette opp utviklingsmiljøet

- **Forutsetninger:**
  - Node.js v18 eller nyere
  - NPM (følger med Node.js)
  - En kodeeditor (f.eks. VS Code)
  - En *Minecraft: Java Edition*-server
  - Alternativt kan du bruke den medfølgende Docker-filen


# Bruker Docker-filen

- For å bygge Docker-bildet, kjør:
  - `$ docker build -t my-bot .`
- For å kjøre Docker-containeren, kjør:
  - `$ docker run --network="host" my-bot`
- Docker-filen kjører `your_code.js`

# Sentrale konsepter i Mineflayer

# Hendelsesdrevet programmering

- Reager på hendelser i spillet ved hjelp av hendelseslyttere (*event listeners*):

```
bot.on('chat', (username, message) => {
  if (username === bot.username) return
  bot.chat(`You said: ${message}`)
})
```

# async/await

- De fleste Mineflayer-handlingene returnerer *promises*:

```
async function goMining() {
  const block = bot.findBlock({ matching: 'diamond_ore' })
  await bot.pathfinder.goto(block.position)
  await bot.dig(block)
}
```

# Arbeide med Minecraft-data

- Konverteringer mellom blokknavn og ID-er er en vanlig utfordring:

```
// Initialize Minecraft data with your bot's version
const minecraftData = require('minecraft-data')(bot.version)

// Get block ID from name
function getBlockId(blockName) {
  const block = minecraftData.blocksByName[blockName]
  if (!block) return null
  return block.id
}
```

# Arbeide med Minecraft-data

```
// Get item ID from name
function getItemId(itemName) {
  const item = minecraftData.itemsByName[itemName]
  if (!item) return null
  return item.id
}
```

# Arbeide med posisjoner og vektorer

- Bruk alltid **`Vec3`** for posisjonskalkuleringer:

```
const Vec3 = require('vec3').Vec3

// Create a position
const position = new Vec3(0, 64, 0)

// Get position above a block
const blockAbove = position.offset(0, 1, 0)
```

# Arbeide med posisjoner og vektorer

```
// Calculate distance
const distance = position.distanceTo(bot.entity.position)

// Get direction vector
const direction = position.minus(bot.entity.position).normalize()
```

# Plassere blokker

```
/**
 * Places a block at a specific position
 * @param {number} x - X coordinate of reference block
 * @param {number} y - Y coordinate of reference block
 * @param {number} z - Z coordinate of reference block
 * @param {number} face_x - X component of face vector
 * @param {number} face_y - Y component of face vector
 * @param {number} face_z - Z component of face vector
 * @param {object} item - The item to place
 */
async function placeBlock (x, y, z, face_x, face_y, face_z, item)
```

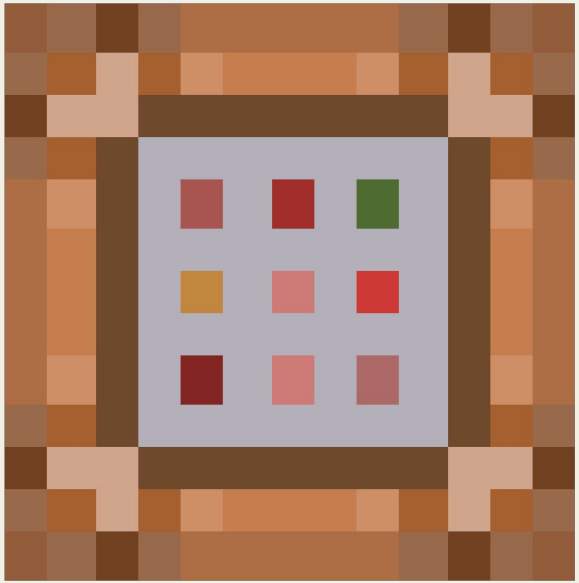
**NB!** Flatevektoren angir hvilken flate av referanseblokken den nye blokken plasseres mot.

# Fra dokumentasjonen

- **Flatevektor:** En av de seks kardinalretningene, f.eks `new Vec3(0, 1, 0)` for å plassere en blokk på toppen av en annen
  - Den nye blokken vil havne på `referanseblokk.posisjon.pluss(flatvektor)`
- Dersom du skal plassere en blokk som ikke ligger på toppen av en annen, må man eksperimentere litt! :)

# Eksempelfunksjoner

- Jeg har samlet flere eksempelfunksjoner fra et tidligere kurs. De finnes i **`example.js`** og er beskrevet i **`README.md`**
- Alle funksjonene er også importert i **`your_code.js`**



# Eksempler på hva boten din kan gjøre

- Boten kan utføre en rekke handlinger, blant annet:
  - **Svare på kommandoer** via chat
  - **Samle ressurser** automatisk
  - **Lage gjenstander** ved behov
  - **Bygge strukturer** (avansert!)
  - **Kjempe mot fiender** og beskytte spilleren
  - **Utforske verdenen** og kartlegge terrenget
  - **Følge spilleren** rundt
- Merk at disse funksjonene kan kombineres ved å bruke `example.js`

# Pizzapause! 

***Tips:*** *Du kan finne lysbildene våre på vår GitHub-side!*
*Søk etter "Koseprogg GitHub" og naviger deg til* ***`presentation-archive`***.

# Gjennomgang av et eksempel

# Ressurser

- Offisiell dokumentasjon:
  https://github.com/PrismarineJS/mineflayer/blob/master/docs/api.md
- Navn på Minecraft-blokker:
  https://minecraft-ids.grahamedgecombe.com/
- Flere eksempler:
  https://github.com/PrismarineJS/mineflayer/tree/master/examples

# Før vi avslutter...

# Vi trenger nye medlemmer!

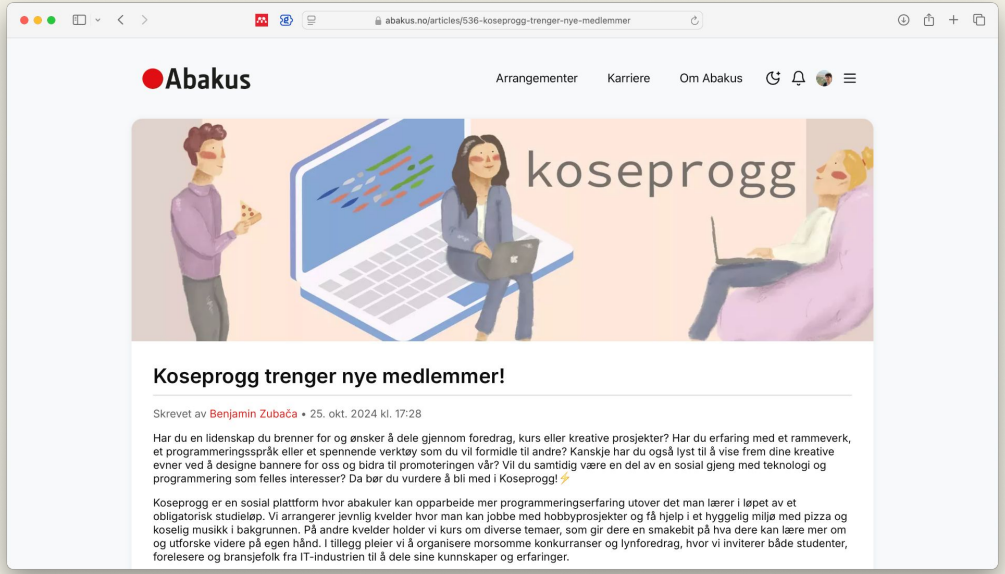
- Vi trenger engasjerte og motiverte folk for å holde Koseprogg i live!
  - Har du et tema du **brenner for** og **ønsker å dele** gjennom foredrag, kurs eller kreative opplegg? ⚡
  - Vil du vise frem dine **kreative evner**? **Design bannere** eller **hjelp oss med promotering!** 🌅
  - Ønsker du å bli en del av en **sosial gjeng** som er glad i teknologi og programmering? 💞
  - Er du en **nysgjerrig** nybegynner som ønsker å **utforske mer** om det vi lærer om? 💭

# Vi trenger nye medlemmer!

- Interessert?
  - Send oss en melding på Slack, ...
  - ... eller send oss en e-post (koseprogg@abakus.no), ...
  - ... og så er du med!

# Vi trenger nye medlemmer!

- Vi har allerede lagt ut mer informasjon på Abakus-hjemmesiden, så gjerne sjekk ut artikkelen nedenfor!



**Neste arrangement:**

*CTF - 8. april*