REACT

THE GOAL FOR THIS SESSION

Be able to recognize common, modern JS features and try them out

AGENDA

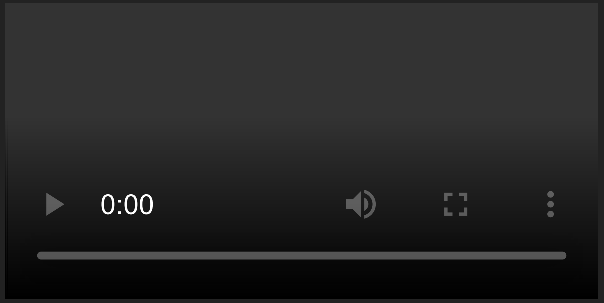
- Crazy Buttons
- Vanilla JS, need to know stuff

CRAZY BUTTONS

1. Challenge
2. Vanilla
3. React

CHALLENGE

HOW WOULD YOU IMPLEMENT THIS



Vanilla

React

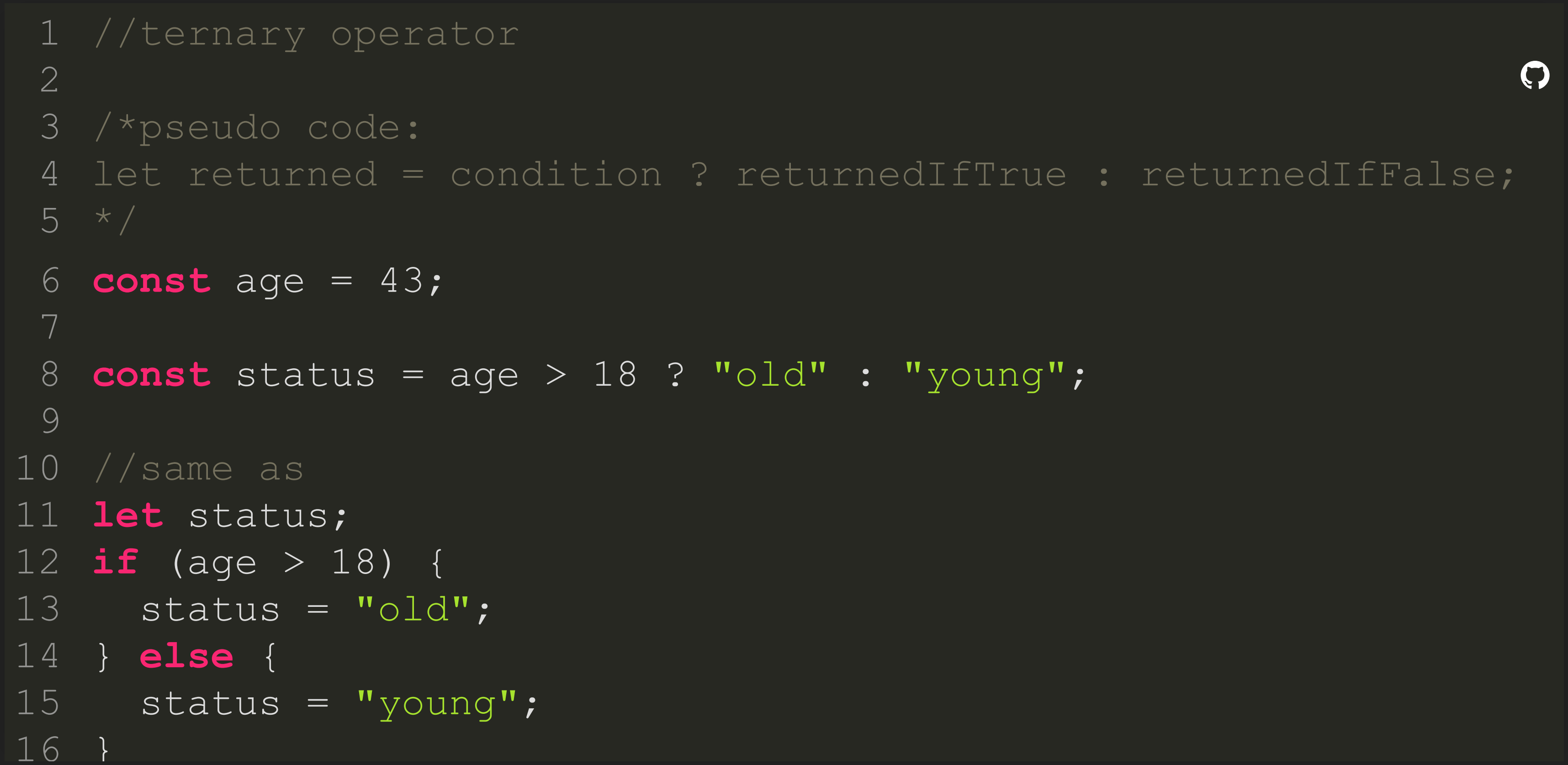
```jsx
import { useState } from "react";

function App() {
  const [masterCount, setMasterCount] = useState(0);
  function updateLocalCount() {
    setMasterCount((prevCount) => prevCount + 1);
  }
  return (
    <div className="App">
      <button onClick={updateLocalCount}>{masterCount}</button>
      <Child masterCount={masterCount} />
      <Child masterCount={masterCount} />
      ...
    </div>
  );
}
```

A LITTLE JS BEFORE WE'RE READY

1. Ternary operator
2. Arrow functions
3. Array.prototype.map
4. Spread operator (on objects as well)
5. Destructuring
6. ES6 Classes (so you've seen them)

TERNARY OPERATOR

Because writing "if" takes too long



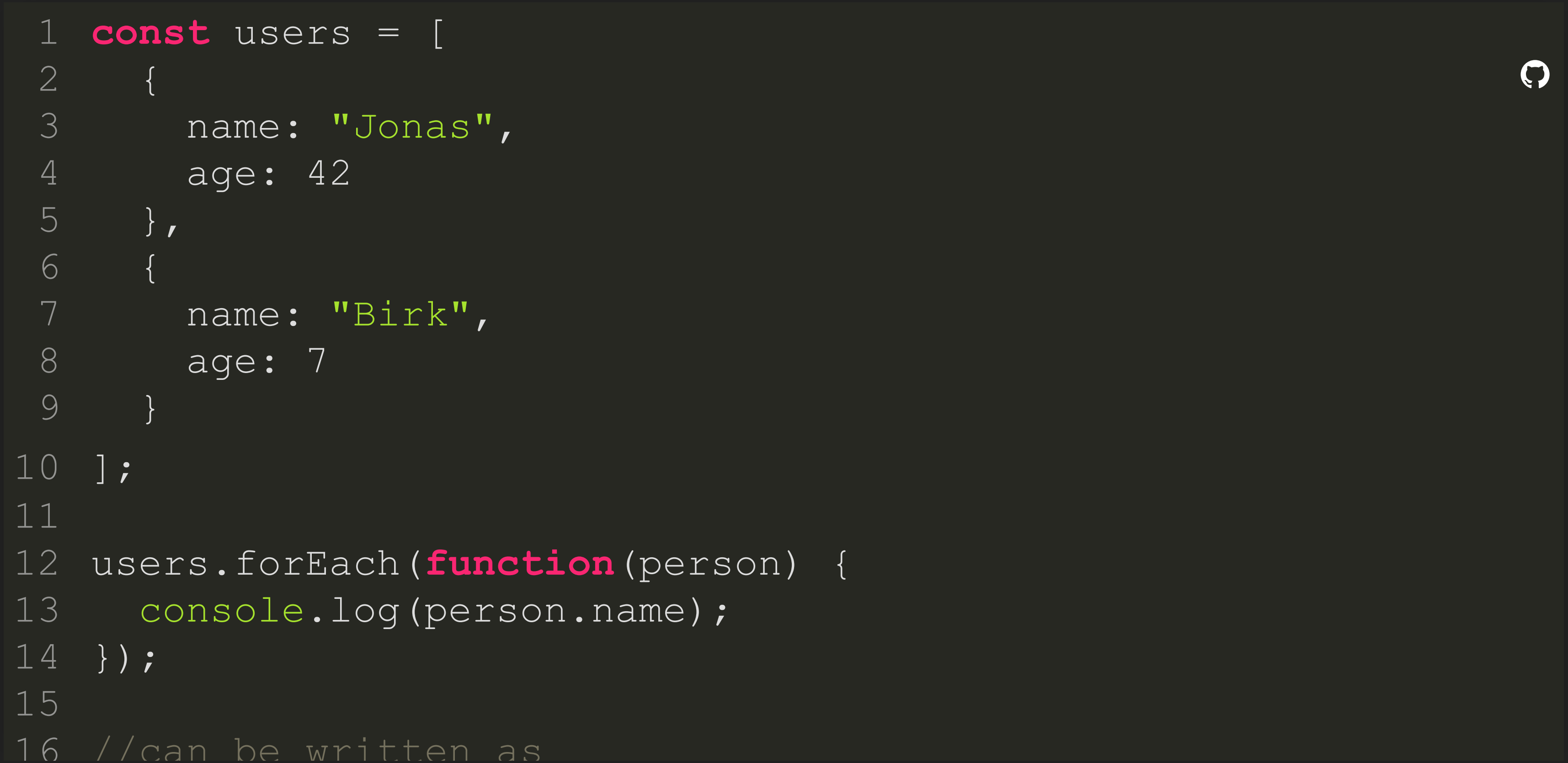
```
//ternary operator

/*pseudo code:
let returned = condition ? returnedIfTrue : returnedIfFalse;
*/
const age = 43;

const status = age > 18 ? "old" : "young";

//same as
let status;
if (age > 18) {
  status = "old";
} else {
  status = "young";
}
```

ARROW FUNCTIONS

Because writing "function" takes too long

```javascript
function double(amount) {
  return amount * 2;
}
const number = double(4);

//can be written as
const double = amount => amount * 2;
const number = double(4);
```

arrays + arrows > awesome



```
const users = [
  {
    name: "Jonas",
    age: 42
  },
  {
    name: "Birk",
    age: 7
  }
];

users.forEach(function(person) {
  console.log(person.name);
});

//can be written as
```

React does not require us to use arrow functions, but most examples use them

```
function sayHi(person) {
  console.log(person.name);
}
users.forEach(sayHi);
```

```javascript
users.forEach(function (person) {
  console.log(person.name);
});
```

```
users.forEach((person) => {
  console.log(person.name);
});
```

```javascript
users.forEach((person) => console.log(person.name));
users.forEach(console.log);
```

Not 100% the same, but....

IMPLICIT RETURN

If we omit the `{ }` it returns something

```
//implicit return
const double = amount => amount * 2;

const number = double(4);

//explicit return
const double = amount => {
  return amount * 2;
};

const number = double(4);
```

ARRAY.PROTOTYPE.MAP

A "new" JS method that let's us operate on an array and return a new one

```
const numbers = [2, 4, 6, 8];

const newNumbers = numbers.map(number => number * 2);

//same as
const newNumbers = numbers.map(function(number) {
  return number * 2;
});
```

It would also be really beneficial to learn Array.filter, Array.sort, Array.concat (and all the other cool Array methods)

SPREAD OPERATOR

The spread operator `...` allows us to "expand an iterable"

It's pretty confusing at first, but it has a multitude of uses you'll see used in React a lot

SPREADING WITH ARRAYS AND STRINGS

1. Copying (simple) arrays, without reference

```
const ar1 = [1,2,3];
const copy = [...ar1]; // [1,2,3]
```

2. Splitting strings

```
const name="Jonas Holbech";
const asArray = [...name]; // ["J", "o", "n"]...
```

SPREADING WITH ARRAYS AND STRINGS

3. Concatenating arrays

```
const ar1 = [1,2,3];
const ar2 = [...ar1, 4, 5]; // [1,2,3,4,5]
```

4. Converting iterables to arrays

```
const asArray = [...document.querySelectorAll("p
// asArray is now a real array,
// so we can use .map / .filter etc
```

SPREADING OBJECTS

1. Copying (simple) objects (without reference)

```
const me = { name: "Jonas" };
const me2 = { ...me }; // {name: "Jonas"}
```

2. Copying objects (without reference) and modifying

```
const me = { name: "Jonas" };
const me2 = { ...me, wives: 1 }; // {name: "Jonas", wives
```

DESTRUCTURING ASSIGNMENT

- MDN
- A bit simpler

Allows us to pick out properties from arrays and objects

Really nice with objects, maybe less so with arrays

WITH OBJECTS

```javascript
const me = {
  name: "Jonas",
  email: "jofh@kea.dk",
  age: 43,
};

//pick out name and email
const { name, email } = me;

//name === "Jonas"
//email === "jofh@kea.dk"

//commonly used with arguments
function sayHi({ name }) {
  //we get the entire object, but pick out name
  console.log(`hi ${name}`);
}

sayHi(me);
```

WITH ARRAYS

```javascript
const names = ["Jonas", "Dannie", "Peter"];

//pick out two names
const [boss, sidekick] = names;
//boss === "Jonas"
//sidekick === "Dannie"


//pick out 1 & 3
const [boss, , henchman] = names;
//boss === "Jonas"
//henchman === "Peter"

//pick out Jonas, and put the rest in an array
const [boss, ...theOthers] = names;
//boss === "Jonas"
//theOthers === ["Dannie", "Peter"];
```

ES6 CLASSES

Just so you've seen them

Older versions of React use these

Simply put, a class is an object containing properties and methods (functions)

These objects are "invoked" using "new"

```
class User {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
  sayHi() {
    console.log(this.name + " says hi");
  }
}

const po = new User("Po", 35);
po.sayHi();
```

The constructor is a method that's automatically called

+3/4

1. Re-write the following if statements to use the ternary operator

```javascript
const age = 43;
const email = "jofh@kea.dk";

let isEven;
if (age % 2 === 0) {
  even = true;
} else {
  even = false;
}

let role = "visitor";
if (email === "jofh@kea.dk.dk") {
  if (age > 40) {
    role = "admin";
  }
}
```

2. Re-write the following to arrow functions

```
function setTitle(title) {
  document.body.title = title;
}

function isEven(number) {
  if (number % 2 === 0) {

    return true;
  }
  return false;
}

function getRandBetween1and10() {
  return Math.floor(Math.random() * 10) + 1;
}
```

3. Re-write the following to arrow functions

```javascript
const movies = [
  {
    name: "Hard Boiled",
    genre: "Hong Kong Action",
  },
  {
    name: "LOTR",
    genre: "Fantasy",
  },
];
function showMovie(movie) {
  console.log(`The movie ${movie.name} is a typical ${movie.genre} m
}

movies.forEach(showMovie);
```

4. Re-write the following to use `map`

Can you do implicit returns?

```javascript
const searchengines = [
  { target: "https://google.com", name: "Google" },

  { target: "https:bing.com", name: "Bing" },
];
let links = "";
searchengines.forEach((link) => {
  links.push(`<a href="${link.target}">${link.name}</a>`);
});
```

5. Follow the instructions

```javascript
const names = ["Jonas", "Dannie", "Peter", "Klaus"];
/* start by making a "bad" copy of this array
by writing something like const myBadCopy = names;
then modify either array by changing one name

then console log both, what happened
*/

/* Then create a copy using the ... spread operator
modify either array and console log them, what happens now?
*/
```

6. Follow the instructions

```javascript
const me = {
  name: "Jonas",
  age: 43,
  kids: 3,
};

/* start by making a bad copy (reference) with something like
const myCopy = me;
change something, see what happens

Then make a new copy using the spead operator,
change something and see what happens
*/
```

7. Watch this

8. Follow the instructions

```
/* this functions receives an object with stuff it does not ne
make it easier for the next developer by destructuring out the
parts we need
Then clean up the function by removing person.
*/

function personCard(person) {
  return `<div class="person">
        <h2>${person.name}</h2>
        <ul>
            <li><a href="tel:+${person.phone}">Call</a></li>
            <li><a href="mailto:${person.email}">E-mail</a></l
        </ul>
    </div>`;
}
const myPerson = {
  name: "Clark Kent",
  marriage: null,
  glasses: true,
  phone: 911,
  street: "secret",
  email: "clark@dailyplanet.com",
};
```

9. Follow the instructions

```javascript
/* this function return 4 random numbers in an array. Use destruc
to pick out the two first items */

function getNums() {
  return [Math.random(), Math.random(), Math.random(), Math.rando
}
```