Certainly! It looks like you've provided a transcript of a lecture on the new class fields proposal in JavaScript, along with some code examples. Let me summarize the key points and provide some additional notes:

### Key Points from the Lecture:

1. \*\*Introduction to Class Fields Proposal:\*\*

- Class fields proposal is aimed at improving and changing JavaScript classes.

- Currently at stage three, indicating a high likelihood of becoming part of the language.

2. \*\*Public Fields:\*\*

- Public fields are properties accessible on all instances of a class.

- Defined using a simple assignment within the class, without the need for keywords like `const` or `let`.

3. \*\*Private Fields:\*\*

- Introduced using the `#` symbol before the field name.

- Truly private, not accessible from the outside of the class.

- Implementation involves using `#` before the field name, making it a private instance property.

4. \*\*Public Methods:\*\*

- Methods declared within the class are automatically public.

- Accessible on instances and part of the class's public API.

5. \*\*Private Methods:\*\*

- Private methods can be used to hide implementation details.

- Similar to private fields, declared using the `#` symbol before the method name.

- Currently, not fully supported in some browsers; the example demonstrated potential future syntax.

6. \*\*Static Methods:\*\*

- Static methods are associated with the class itself, not instances.

- Defined using the `static` keyword before the method name.

### Additional Notes:

- \*\*Browser Support:\*\*

- The class fields proposal may not be fully supported in all browsers. As of the recording, Google Chrome has partial support.

- Developers should test code in environments that support these features.

- \*\*Private vs. Protected:\*\*

- Private fields and methods offer true encapsulation, providing stricter privacy compared to using conventions like underscores for indicating protected properties.

- \*\*Syntax Considerations:\*\*

- The syntax for private fields and methods may undergo changes as the proposal progresses through stages.

- \*\*Encapsulation and Data Privacy:\*\*

- The class fields proposal enhances encapsulation and data privacy in JavaScript classes.

- Developers can now create truly private properties and methods.

- \*\*Static Versions:\*\*

- In addition to public and private fields/methods, there are static versions for each, primarily used for class-level functionality.

### Provided Code Example:

```javascript

class Account {

#movements = [];

#pin;

constructor(locale) {

this.locale = navigator.language;

this.#pin = locale;

}

// Public Fields

locale = this.#pin;

// Public Methods

deposit(val) {

this.#movements.push(val);

}

withdraw(val) {

this.deposit(-val);

}

approveLoan(val) {

return val <= this.\_approveLoanThreshold();

}

// Private Methods

#approveLoanThreshold() {

return this.#movements.reduce((acc, mov) => acc + mov, 0) \* 0.1;

}

}

const account1 = new Account('en-US');

account1.deposit(250);

account1.withdraw(140);

account1.approveLoan(300);

```

### Final Note:

- Stay updated on the progress of the class fields proposal as it moves through different stages. The syntax and features might evolve.

This summary combines the provided lecture transcript with additional insights and notes for better understanding. If you have specific questions or need clarification on any point, feel free to ask!