JavaScript

STUDIA PODYPLOMOWE
POLITECHNIKA BIAŁOSTOCKA

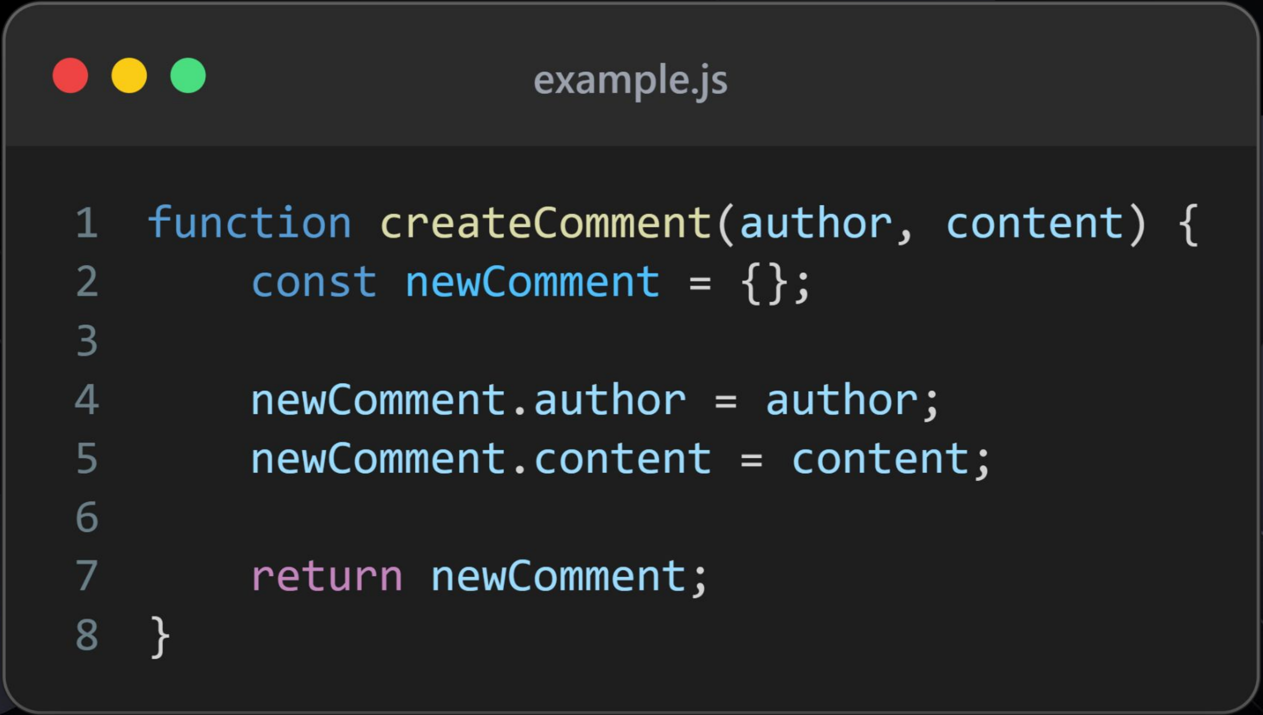
HOMEWORK

Comments

Implementation with prototype

Object creation

- Create a function that takes in 2 params (author, content) and returns an object
- The function should:
  - create an empty object
  - set author and content of the object
  - return the object

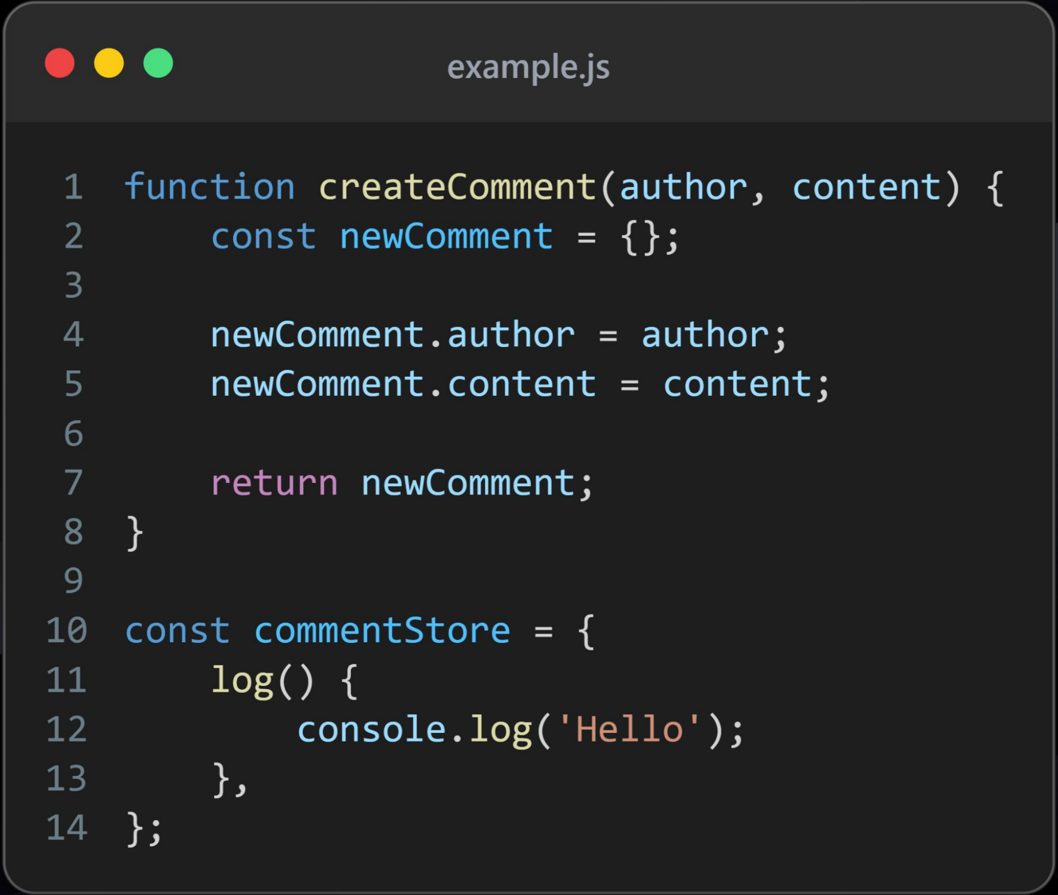


example.js

```
function createComment(author, content) {
    const newComment = {};

    newComment.author = author;
    newComment.content = content;

    return newComment;
}
```

Comment Functions Store

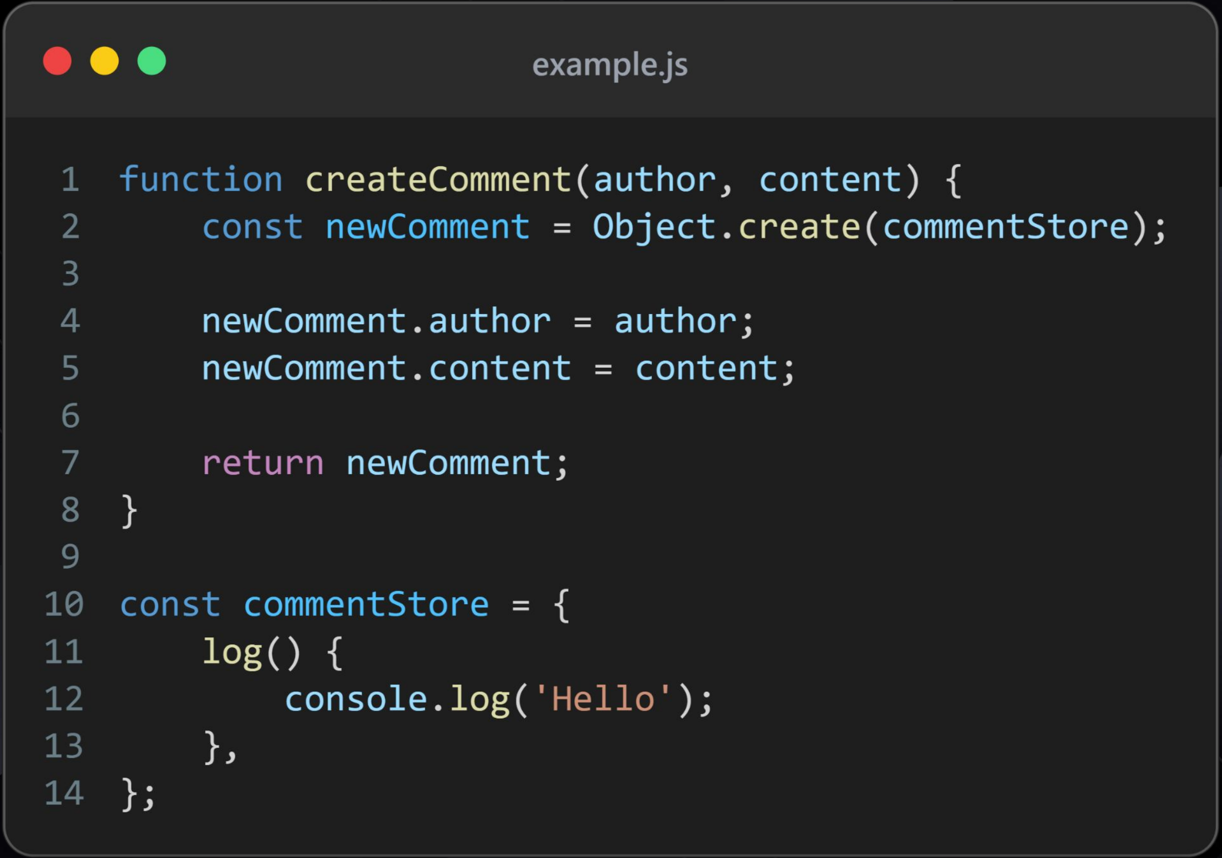
- Create a commentStore object that will hold comments functionality and add a method *log* that will show ‘hello’ in console (for now)



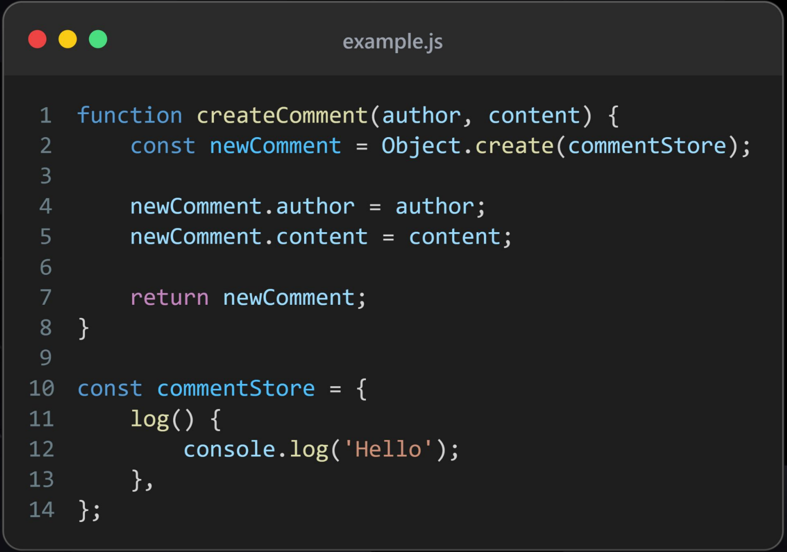
```javascript
function createComment(author, content) {
    const newComment = {};

    newComment.author = author;
    newComment.content = content;

    return newComment;
}

const commentStore = {
    log() {
        console.log('Hello');
    },
};
```

Comment Functions Store

- Modify the createComment function, so that when called, it will create a comment object that has a prototypal link to the commentStore

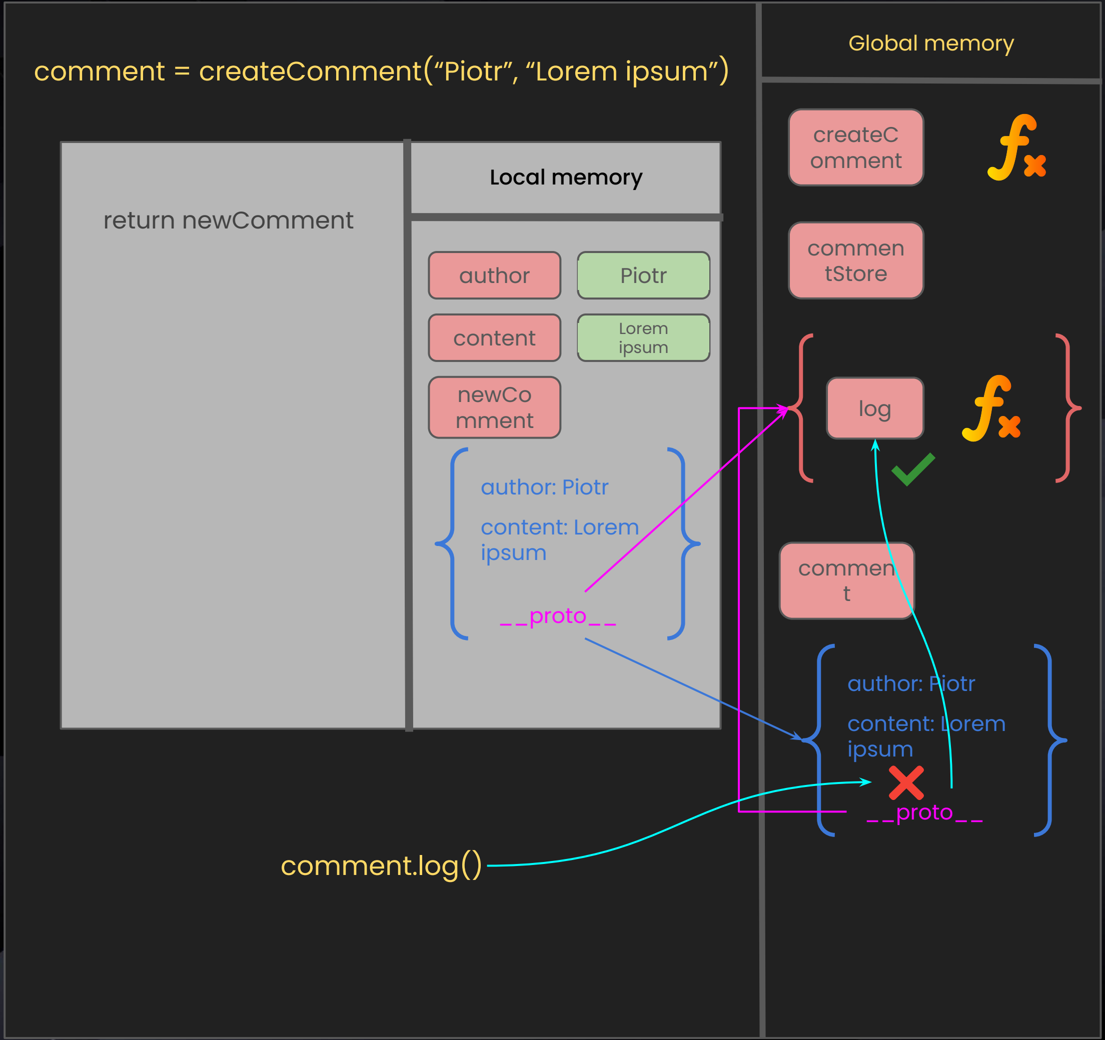


```javascript
function createComment(author, content) {
    const newComment = Object.create(commentStore);

    newComment.author = author;
    newComment.content = content;

    return newComment;
}

const commentStore = {
    log() {
        console.log('Hello');
    },
};
```



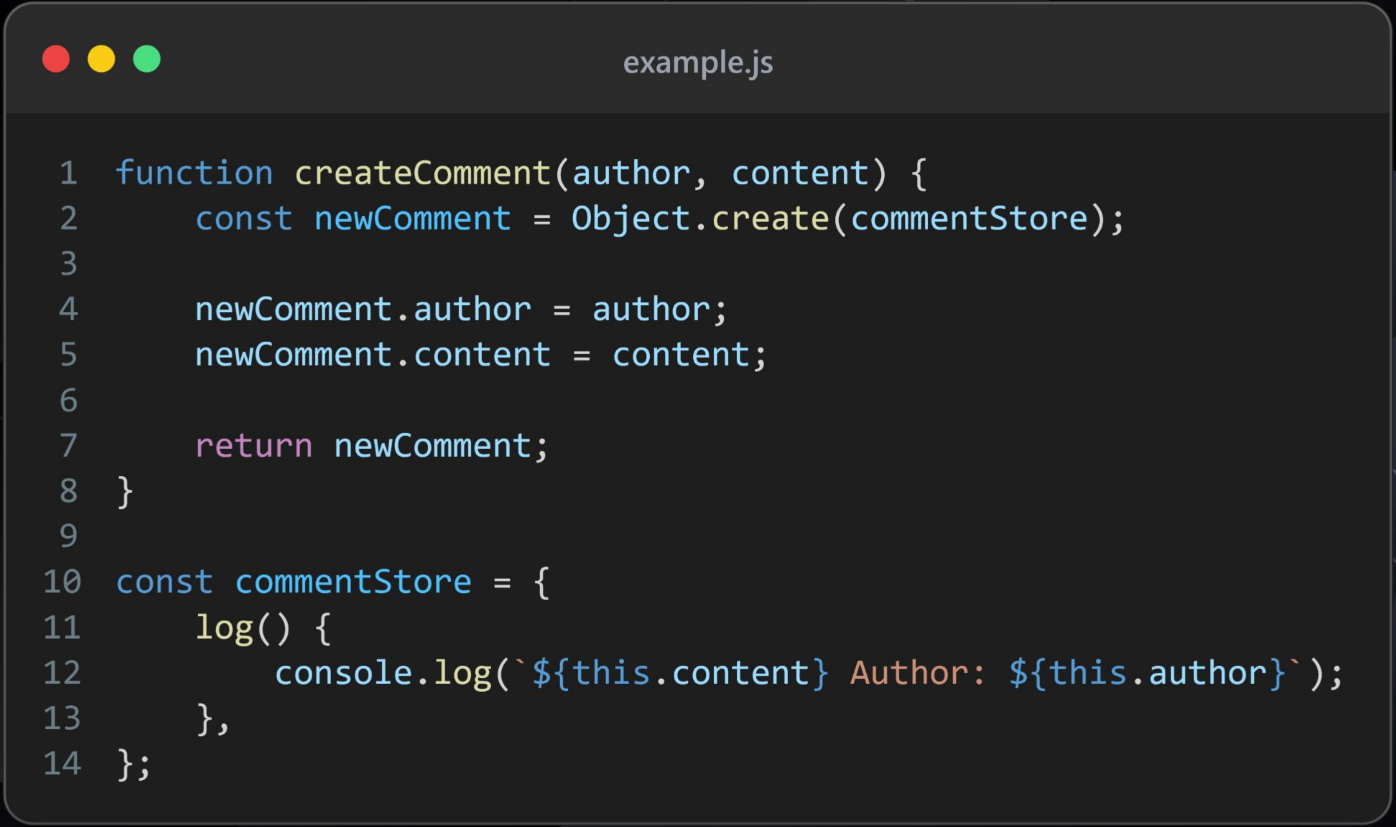
example.js

```javascript
function createComment(author, content) {
    const newComment = Object.create(commentStore);

    newComment.author = author;
    newComment.content = content;

    return newComment;
}

const commentStore = {
    log() {
        console.log('Hello');
    },
};
```

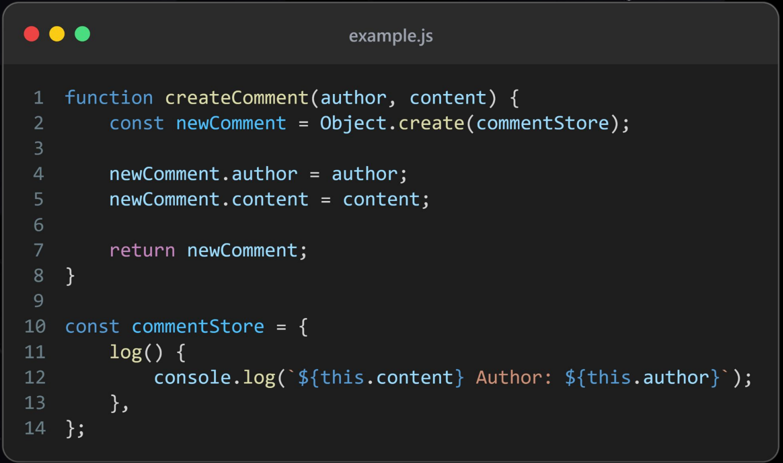


Comment Functions Store

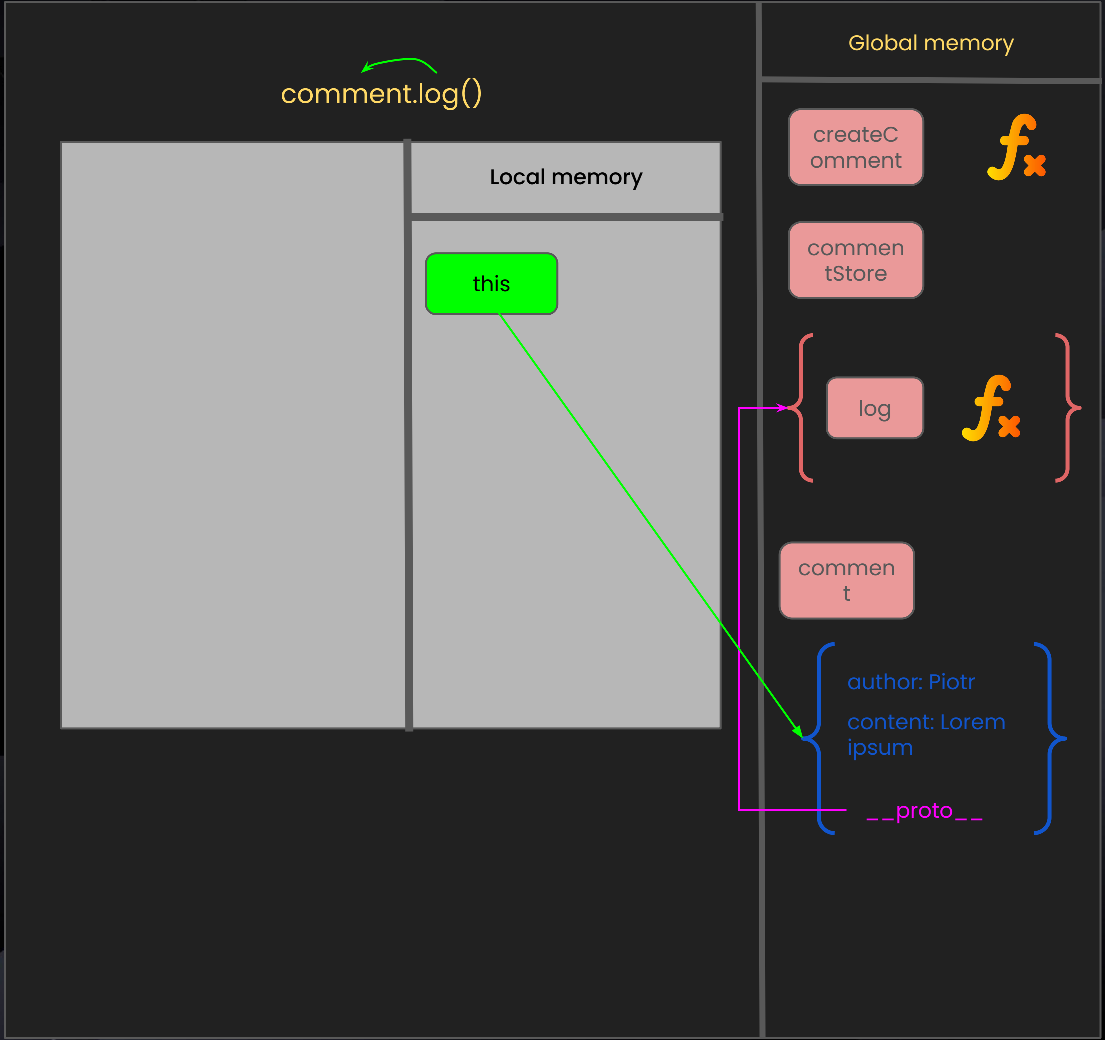
- Modify the log method in commentStore, so it will show the author and the content of the comment


example.js

```js
function createComment(author, content) {
    const newComment = Object.create(commentStore);

    newComment.author = author;
    newComment.content = content;

    return newComment;
}

const commentStore = {
    log() {
        console.log(`${this.content} Author: ${this.author}`);
    },
};
```

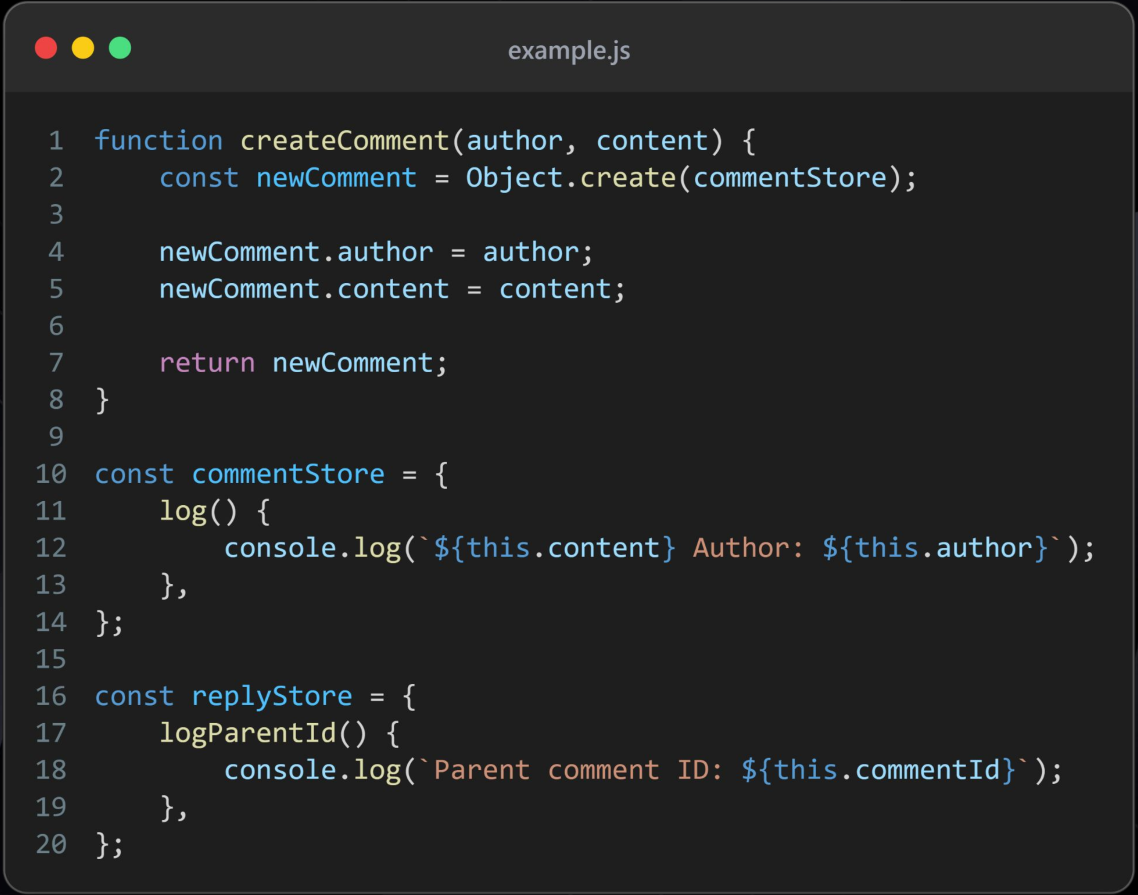

example.js

```
function createComment(author, content) {
    const newComment = Object.create(commentStore);

    newComment.author = author;
    newComment.content = content;

    return newComment;
}

const commentStore = {
    log() {
        console.log(`${this.content} Author: ${this.author}`);
    },
};
```



Reply

- Create a replyStore object, that has a *logParentId* method on it
- The method should log the parent's comment's ID (*this.commentId*)

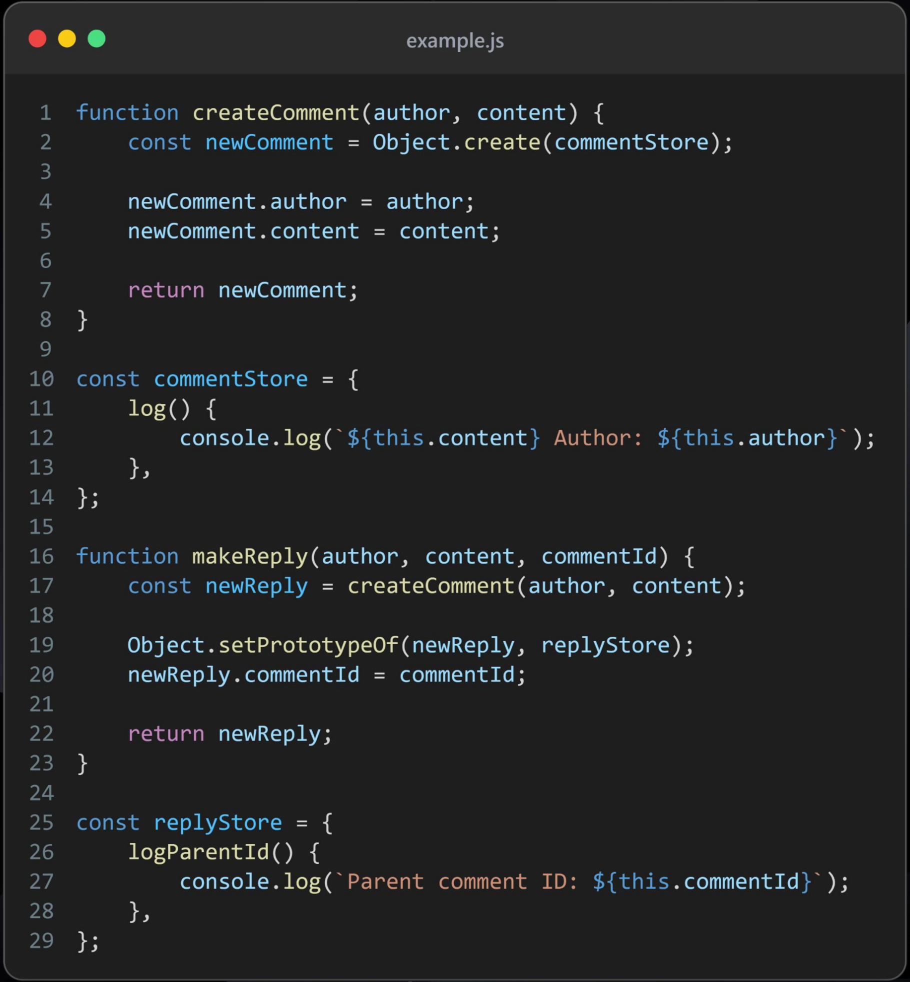


example.js

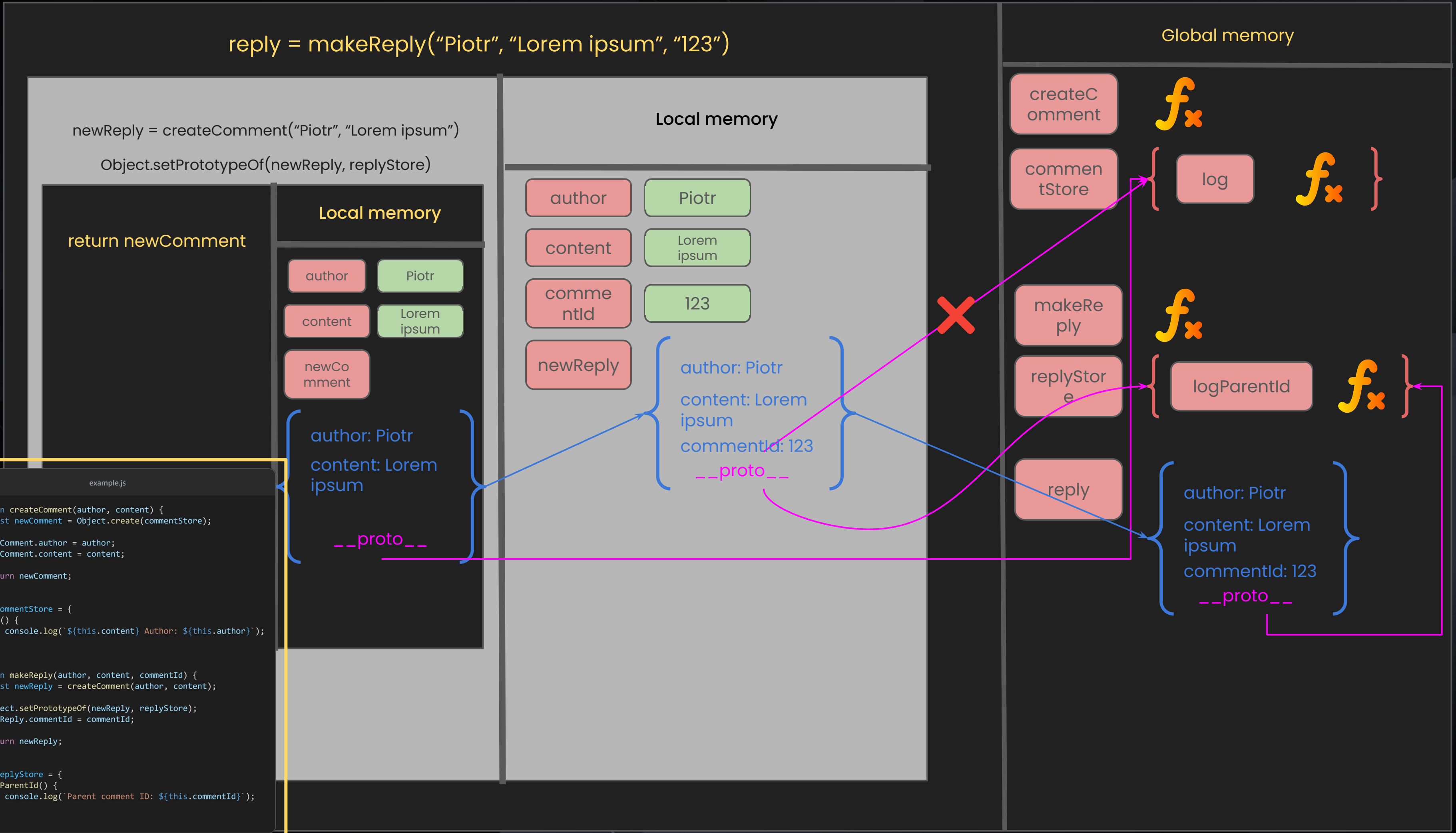
```javascript
function createComment(author, content) {
    const newComment = Object.create(commentStore);

    newComment.author = author;
    newComment.content = content;

    return newComment;
}

const commentStore = {
    log() {
        console.log(`${this.content} Author: ${this.author}`);
    },
};

const replyStore = {
    logParentId() {
        console.log(`Parent comment ID: ${this.commentId}`);
    },
};
```

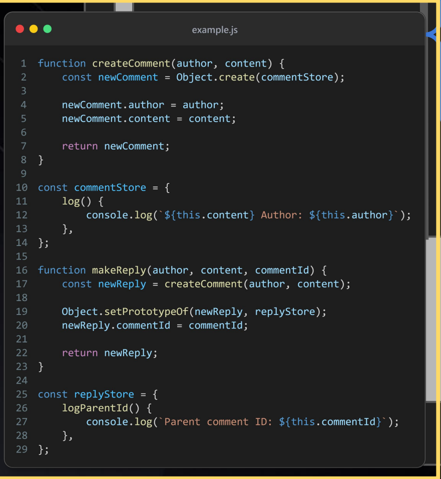
Reply

- Create a makeReply function, that takes in 3 params (author, content, commentId)
- The function should utilize createComment function
- The function should return a reply object with author, content, commentId, that has a prototypal link to replyStore


example.js

```javascript
function createComment(author, content) {
    const newComment = Object.create(commentStore);

    newComment.author = author;
    newComment.content = content;

    return newComment;
}

const commentStore = {
    log() {
        console.log(`${this.content} Author: ${this.author}`);
    },
};

function makeReply(author, content, commentId) {
    const newReply = createComment(author, content);

    Object.setPrototypeOf(newReply, replyStore);
    newReply.commentId = commentId;

    return newReply;
}

const replyStore = {
    logParentId() {
        console.log(`Parent comment ID: ${this.commentId}`);
    },
};
```

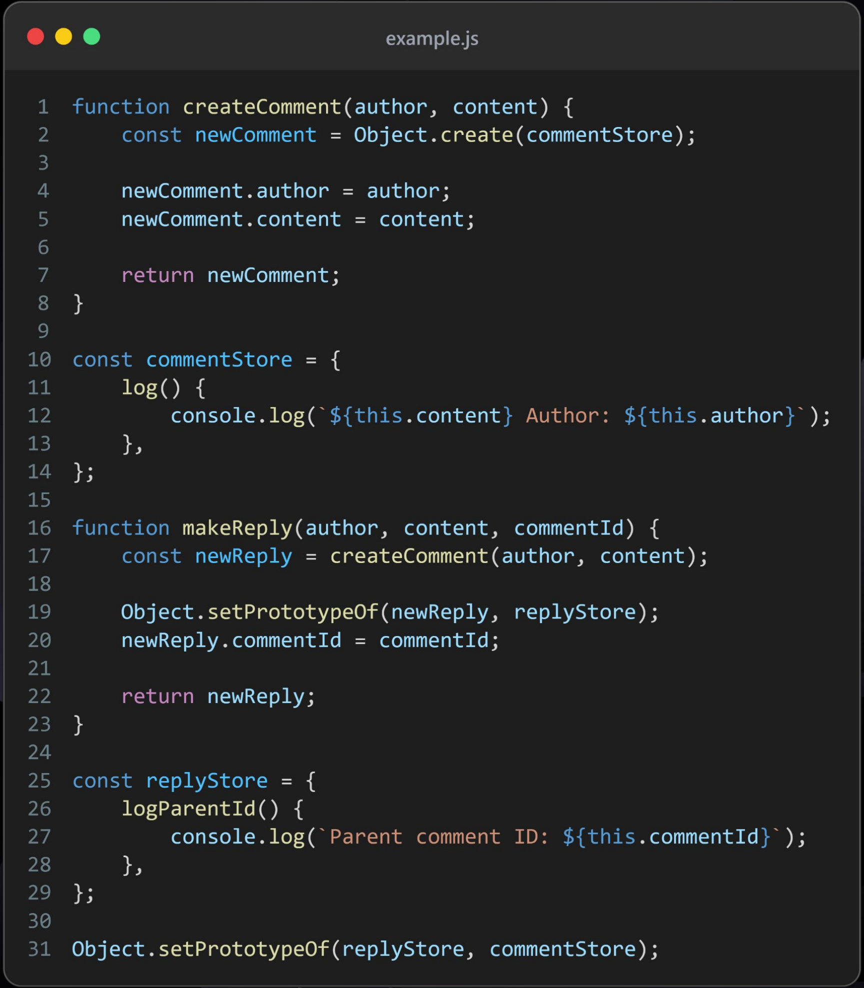




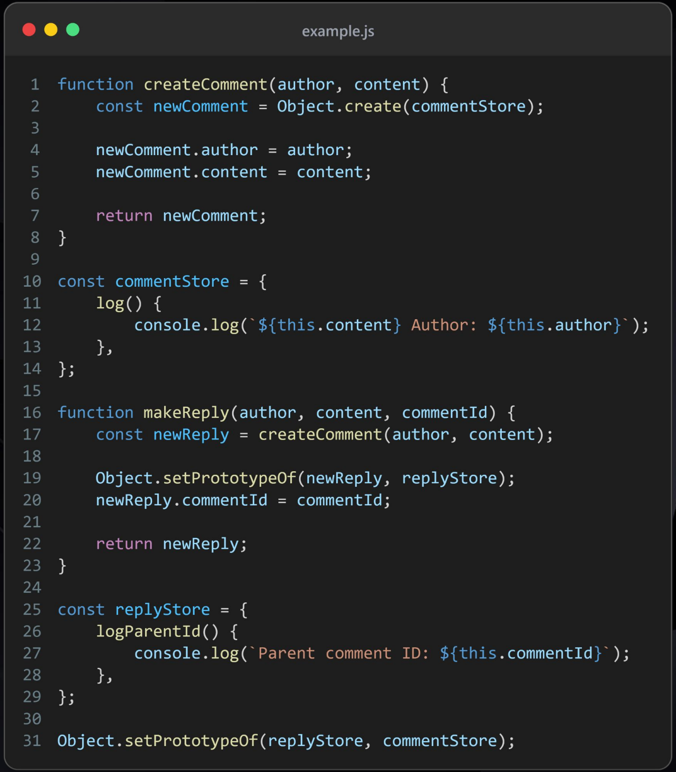
```javascript
function createComment(author, content) {
    const newComment = Object.create(commentStore);

    newComment.author = author;
    newComment.content = content;

    return newComment;
}

const commentStore = {
    log() {
        console.log(`${this.content} Author: ${this.author}`);
    },
};

function makeReply(author, content, commentId) {
    const newReply = createComment(author, content);

    Object.setPrototypeOf(newReply, replyStore);
    newReply.commentId = commentId;

    return newReply;
}

const replyStore = {
    logParentId() {
        console.log(`Parent comment ID: ${this.commentId}`);
    },
};
```

Reply

- Modify the code, so that the reply object could also use the *log* method from commentStore - use prototypal chain

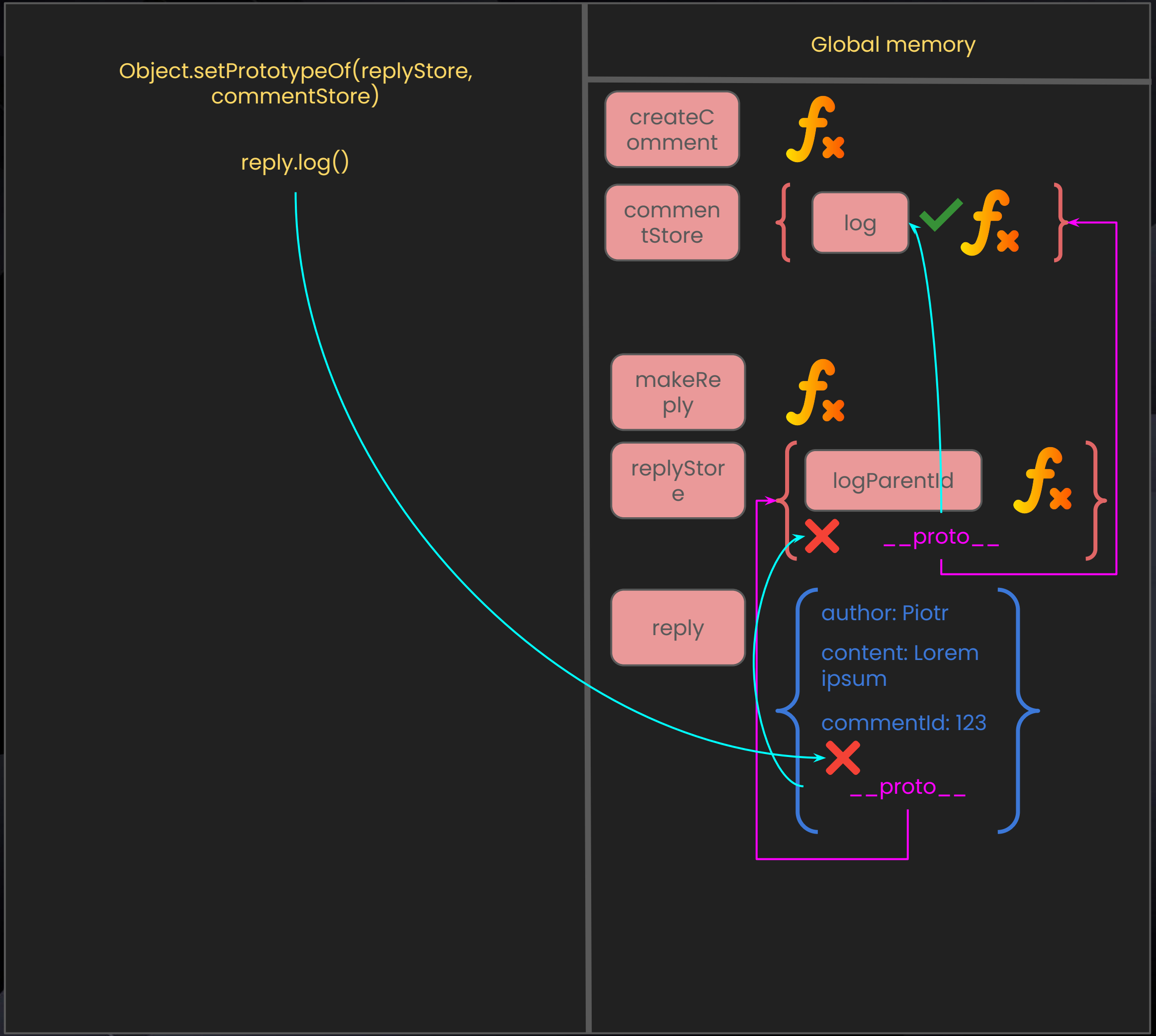

example.js

```javascript
function createComment(author, content) {
    const newComment = Object.create(commentStore);

    newComment.author = author;
    newComment.content = content;

    return newComment;
}

const commentStore = {
    log() {
        console.log(`${this.content} Author: ${this.author}`);
    },
};

function makeReply(author, content, commentId) {
    const newReply = createComment(author, content);

    Object.setPrototypeOf(newReply, replyStore);
    newReply.commentId = commentId;

    return newReply;
}

const replyStore = {
    logParentId() {
        console.log(`Parent comment ID: ${this.commentId}`);
    },
};

Object.setPrototypeOf(replyStore, commentStore);
```



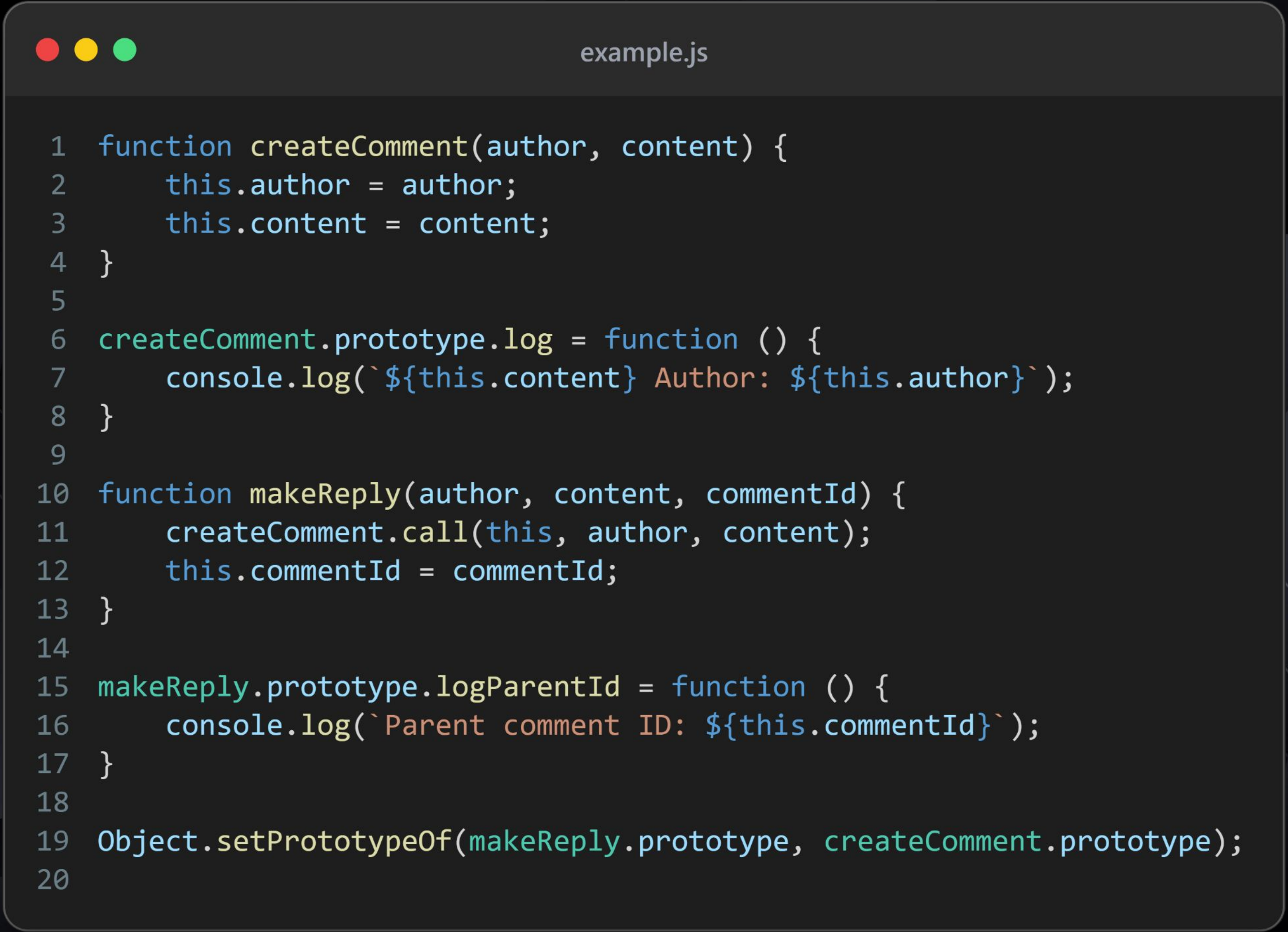
example.js

```
function createComment(author, content) {
    const newComment = Object.create(commentStore);

    newComment.author = author;
    newComment.content = content;

    return newComment;
}

const commentStore = {
    log() {
        console.log(`${this.content} Author: ${this.author}`);
    },
};

function makeReply(author, content, commentId) {
    const newReply = createComment(author, content);

    Object.setPrototypeOf(newReply, replyStore);
    newReply.commentId = commentId;

    return newReply;
}

const replyStore = {
    logParentId() {
        console.log(`Parent comment ID: ${this.commentId}`);
    },
};

Object.setPrototypeOf(replyStore, commentStore);
```

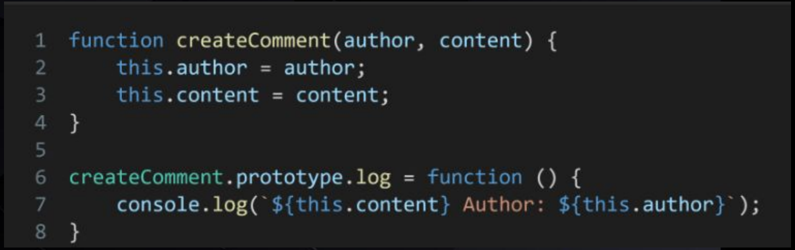


new

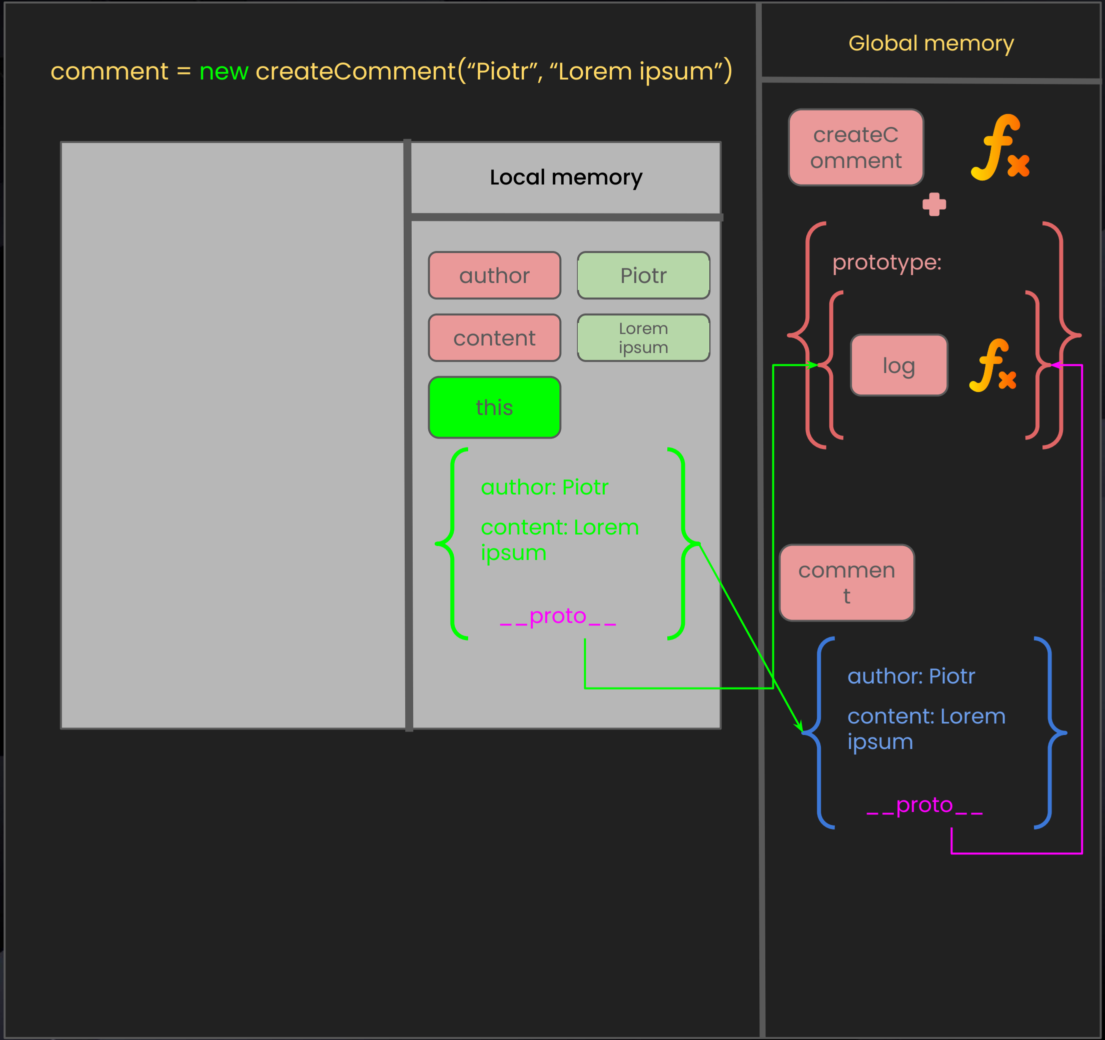
- Modify the code, so we can use *new* keyword to create comments and replies

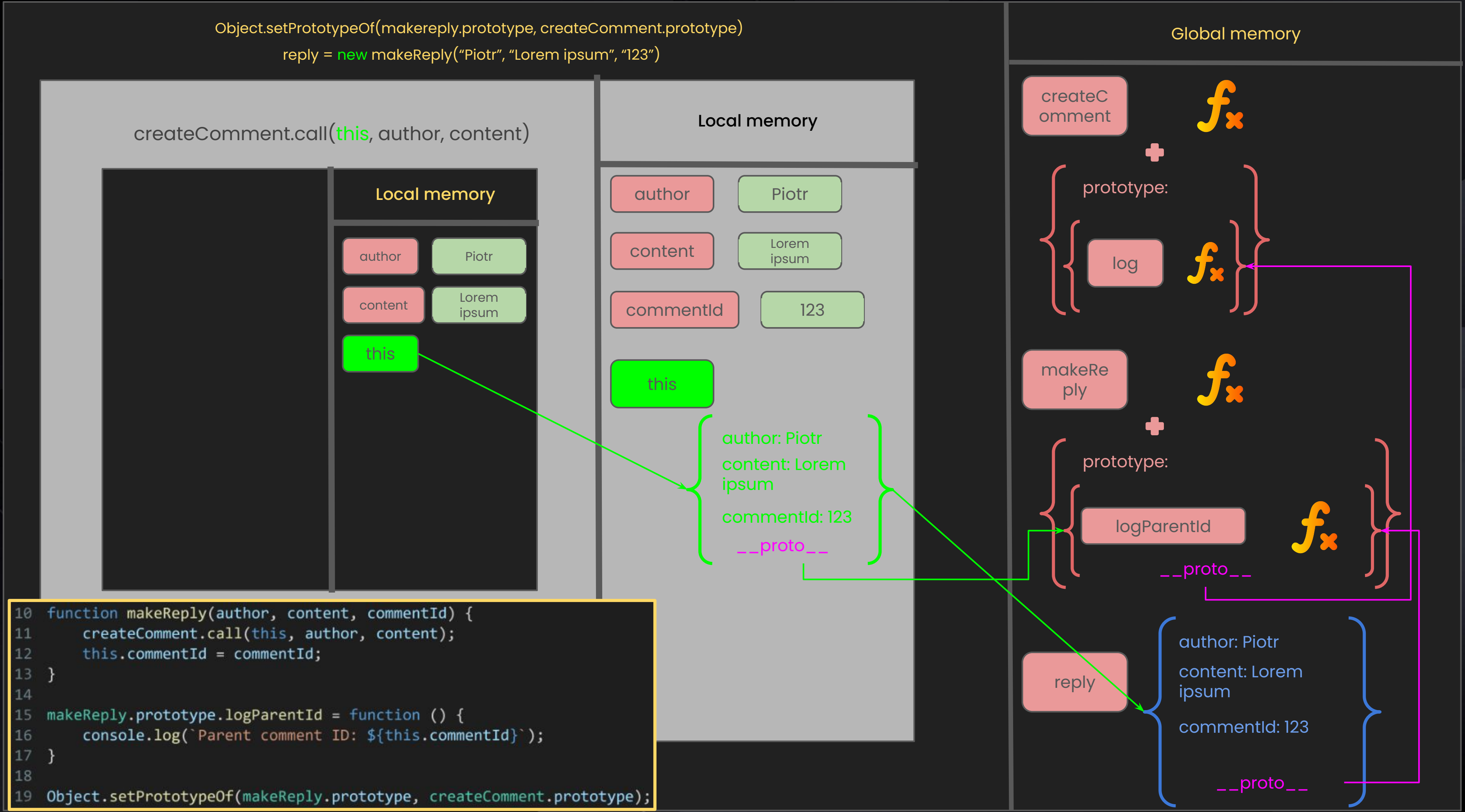

example.js

```javascript
function createComment(author, content) {
    this.author = author;
    this.content = content;
}

createComment.prototype.log = function () {
    console.log(`${this.content} Author: ${this.author}`);
}

function makeReply(author, content, commentId) {
    createComment.call(this, author, content);
    this.commentId = commentId;
}

makeReply.prototype.logParentId = function () {
    console.log(`Parent comment ID: ${this.commentId}`);
}

Object.setPrototypeOf(makeReply.prototype, createComment.prototype);

```



```
function createComment(author, content) {
    this.author = author;
    this.content = content;
}

createComment.prototype.log = function () {
    console.log(`${this.content} Author: ${this.author}`);
}
```

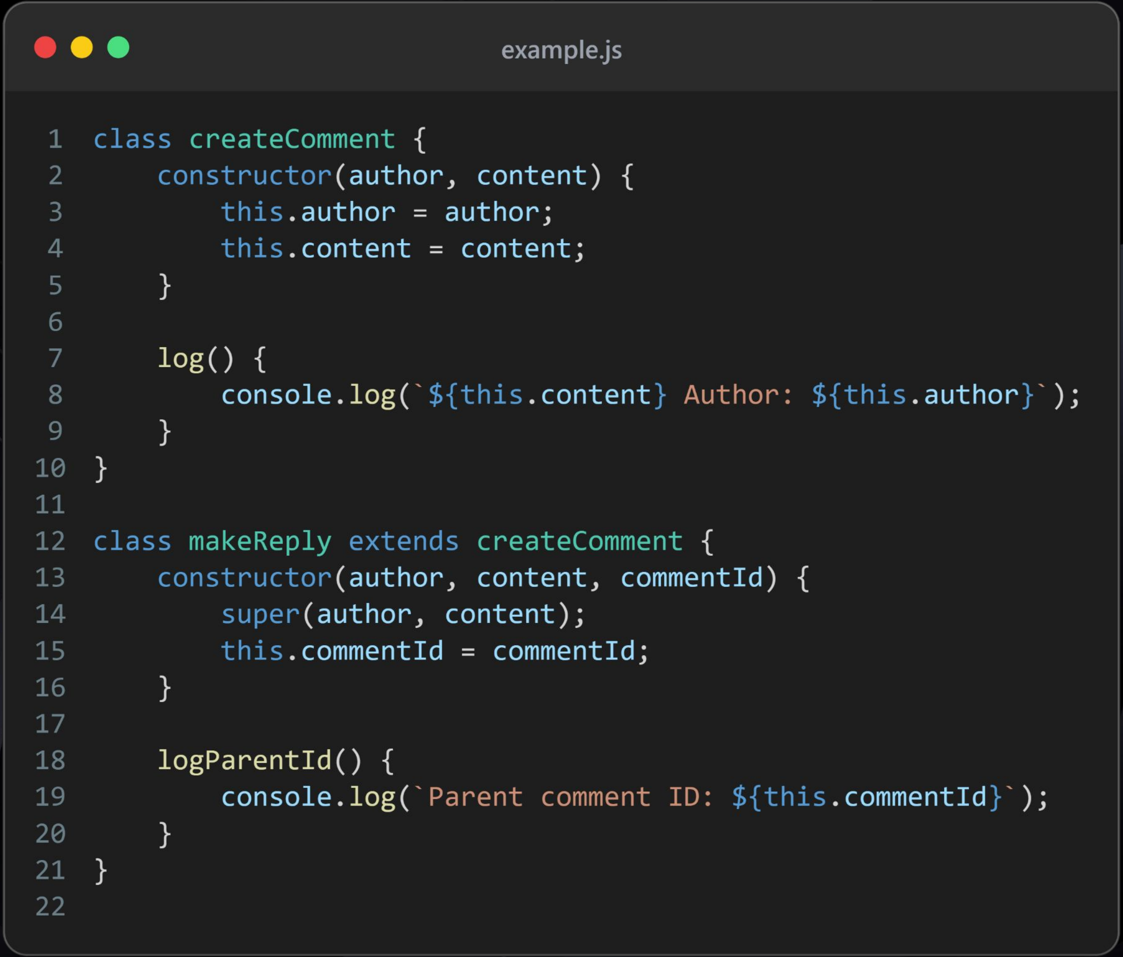




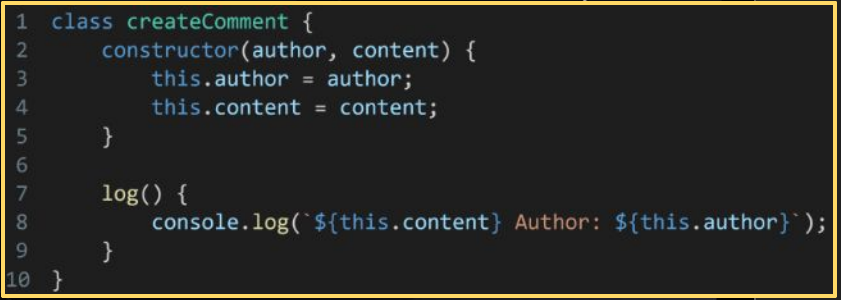
```javascript
function makeReply(author, content, commentId) {
    createComment.call(this, author, content);
    this.commentId = commentId;
}

makeReply.prototype.logParentId = function () {
    console.log(`Parent comment ID: ${this.commentId}`);
}

Object.setPrototypeOf(makeReply.prototype, createComment.prototype);
```

ES6 - classes

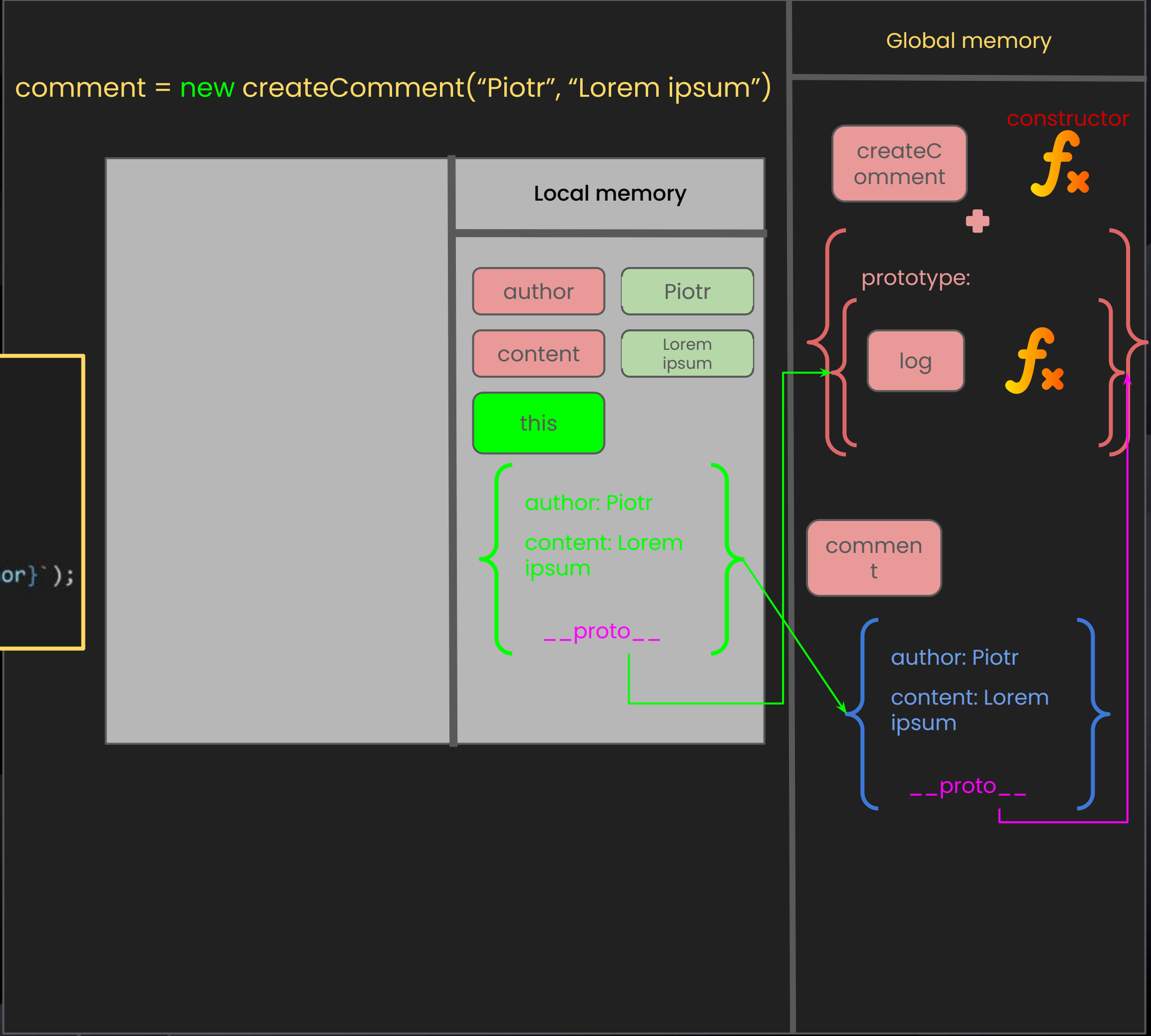
- Refactor the code with ES6 classes


example.js

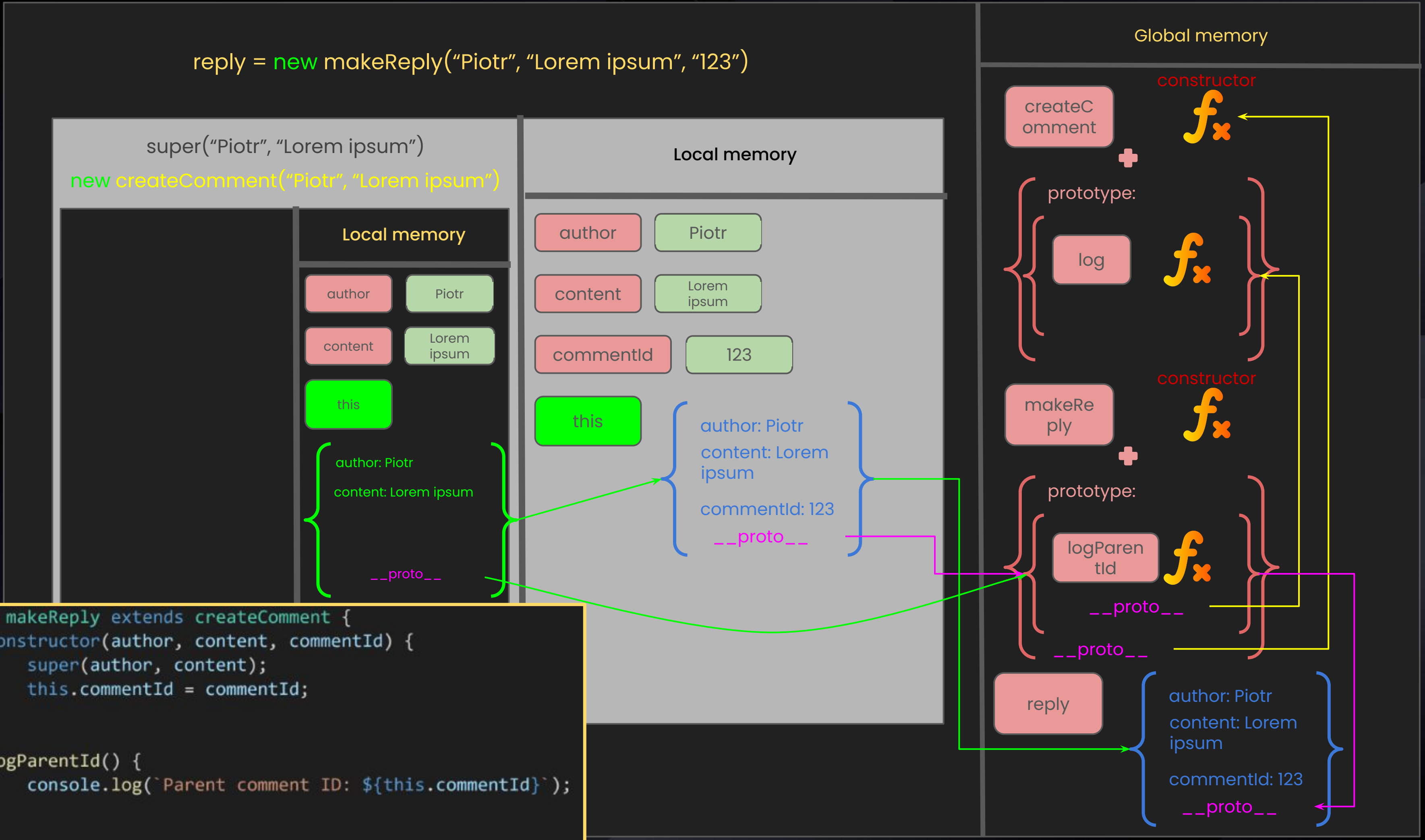
```
class createComment {
    constructor(author, content) {
        this.author = author;
        this.content = content;
    }

    log() {
        console.log(`${this.content} Author: ${this.author}`);
    }
}

class makeReply extends createComment {
    constructor(author, content, commentId) {
        super(author, content);
        this.commentId = commentId;
    }

    logParentId() {
        console.log(`Parent comment ID: ${this.commentId}`);
    }
}

```



```
class createComment {
    constructor(author, content) {
        this.author = author;
        this.content = content;
    }

    log() {
        console.log(`${this.content} Author: ${this.author}`);
    }
}
```



comment = new createComment(“Piotr”, “Lorem ipsum”)



```
class makeReply extends createComment {
    constructor(author, content, commentId) {
        super(author, content);
        this.commentId = commentId;
    }

    logParentId() {
        console.log(`Parent comment ID: ${this.commentId}`);
    }
}
```

HOMEWORK

USE CLASSES

- English to Morse code
- Palindromes
- Longest common subsequence
- Matrices multiplication