



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

```
1  function createComment(author, content) {  
2      const newComment = {};  
3  
4      newComment.author = author;  
5      newComment.content = content;  
6  
7      return newComment;  
8  }
```

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)



example.js

```
1  function createComment(author, content) {  
2      const newComment = {};  
3  
4      newComment.author = author;  
5      newComment.content = content;  
6  
7      return newComment;  
8  }  
9  
10 const commentStore = {  
11     log() {  
12         console.log('Hello');  
13     },  
14 };
```

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



example.js

```
1 function createComment(author, content) {  
2     const newComment = Object.create(commentStore);  
3  
4     newComment.author = author;  
5     newComment.content = content;  
6  
7     return newComment;  
8 }  
9  
10 const commentStore = {  
11     log() {  
12         console.log('Hello');  
13     },  
14 };
```

example.js

```
1 function createComment(author, content) {  
2   const newComment = Object.create(commentStore);  
3  
4   newComment.author = author;  
5   newComment.content = content;  
6  
7   return newComment;  
8 }  
9  
10 const commentStore = {  
11   log() {  
12     console.log('Hello');  
13   },  
14 };
```

comment = createComment("Piotr", "Lorem ipsum")

return newComment

Local memory

author

Piotr

content

Lorem ipsum

newComment

author: Piotr

content: Lorem ipsum

__proto__

Global memory

createComment

f_x

commentStore

log

f_x

comment

author: Piotr

content: Lorem ipsum

__proto__

comment.log()

Comment Functions Store

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



example.js

```
1  function createComment(author, content) {  
2      const newComment = Object.create(commentStore);  
3  
4      newComment.author = author;  
5      newComment.content = content;  
6  
7      return newComment;  
8  }  
9  
10 const commentStore = {  
11     log() {  
12         console.log(`${this.content} Author: ${this.author}`);  
13     },  
14 };
```

example.js

```
1 function createComment(author, content) {  
2   const newComment = Object.create(commentStore);  
3  
4   newComment.author = author;  
5   newComment.content = content;  
6  
7   return newComment;  
8 }  
9  
10 const commentStore = {  
11   log() {  
12     console.log(`${this.content} Author: ${this.author}`);  
13   },  
14 };
```

comment.log()

Local memory

this

Global memory

createC
omment

f_x

commen
tStore

log

f_x

commen
t

author: Piotr

content: Lorem
ipsum

__proto__

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

```
1 function createComment(author, content) {  
2     const newComment = Object.create(commentStore);  
3  
4     newComment.author = author;  
5     newComment.content = content;  
6  
7     return newComment;  
8 }  
9  
10 const commentStore = {  
11     log() {  
12         console.log(`${this.content} Author: ${this.author}`);  
13     },  
14 };  
15  
16 const replyStore = {  
17     logParentId() {  
18         console.log(`Parent comment ID: ${this.commentId}`);  
19     },  
20 };
```

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

```
1 function createComment(author, content) {
2   const newComment = Object.create(commentStore);
3
4   newComment.author = author;
5   newComment.content = content;
6
7   return newComment;
8 }
9
10 const commentStore = {
11   log() {
12     console.log(`${this.content} Author: ${this.author}`);
13   },
14 };
15
16 function makeReply(author, content, commentId) {
17   const newReply = createComment(author, content);
18
19   Object.setPrototypeOf(newReply, replyStore);
20   newReply.commentId = commentId;
21
22   return newReply;
23 }
24
25 const replyStore = {
26   logParentId() {
27     console.log(`Parent comment ID: ${this.commentId}`);
28   },
29 };
```

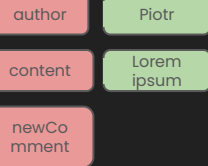
reply = makeReply("Piotr", "Lorem ipsum", "123")

newReply = createComment("Piotr", "Lorem ipsum")

Object.setPrototypeOf(newReply, replyStore)

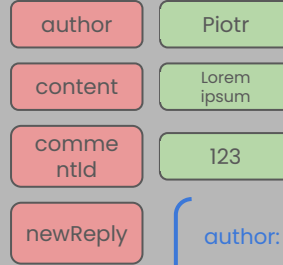
return newComment

Local memory



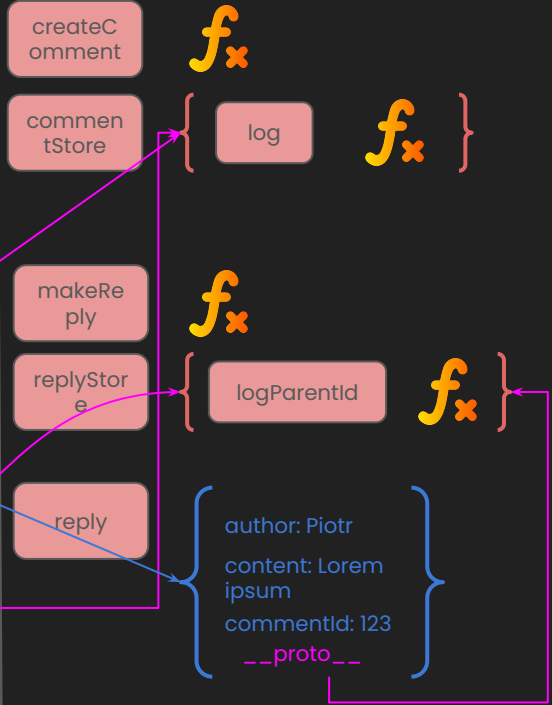
author: Piotr
content: Lorem ipsum
__proto__

Local memory



author: Piotr
content: Lorem ipsum
commentId: 123
__proto__

Global memory



example.js

```
1 function createComment(author, content) {
2   const newComment = Object.create(commentStore);
3
4   newComment.author = author;
5   newComment.content = content;
6
7   return newComment;
8 }
9
10 const commentStore = {
11   log() {
12     console.log(`${this.content} Author: ${this.author}`);
13   },
14 };
15
16 function makeReply(author, content, commentId) {
17   const newReply = createComment(author, content);
18
19   Object.setPrototypeOf(newReply, replyStore);
20   newReply.commentId = commentId;
21
22   return newReply;
23 }
24
25 const replyStore = {
26   logParentId() {
27     console.log(`Parent comment ID: ${this.commentId}`);
28   },
29 };
```

Reply

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

example.js

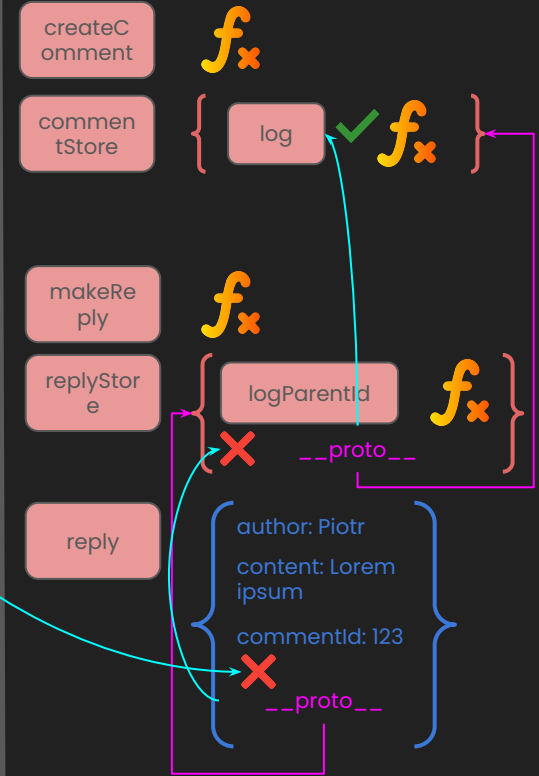
```
1 function createComment(author, content) {
2     const newComment = Object.create(commentStore);
3
4     newComment.author = author;
5     newComment.content = content;
6
7     return newComment;
8 }
9
10 const commentStore = {
11     log() {
12         console.log(`${this.content} Author: ${this.author}`);
13     },
14 };
15
16 function makeReply(author, content, commentId) {
17     const newReply = createComment(author, content);
18
19     Object.setPrototypeOf(newReply, replyStore);
20     newReply.commentId = commentId;
21
22     return newReply;
23 }
24
25 const replyStore = {
26     logParentId() {
27         console.log(`Parent comment ID: ${this.commentId}`);
28     },
29 };
30
31 Object.setPrototypeOf(replyStore, commentStore);
```

```
example.js
1 function createComment(author, content) {
2   const newComment = Object.create(commentStore);
3
4   newComment.author = author;
5   newComment.content = content;
6
7   return newComment;
8 }
9
10 const commentStore = {
11   log() {
12     console.log(`${this.content} Author: ${this.author}`);
13   },
14 };
15
16 function makeReply(author, content, commentId) {
17   const newReply = createComment(author, content);
18
19   Object.setPrototypeOf(newReply, replyStore);
20   newReply.commentId = commentId;
21
22   return newReply;
23 }
24
25 const replyStore = {
26   logParentId() {
27     console.log(`Parent comment ID: ${this.commentId}`);
28   },
29 };
30
31 Object.setPrototypeOf(replyStore, commentStore);
```

Object.setPrototypeOf(replyStore, commentStore)

reply.log()

Global memory



new

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

example.js

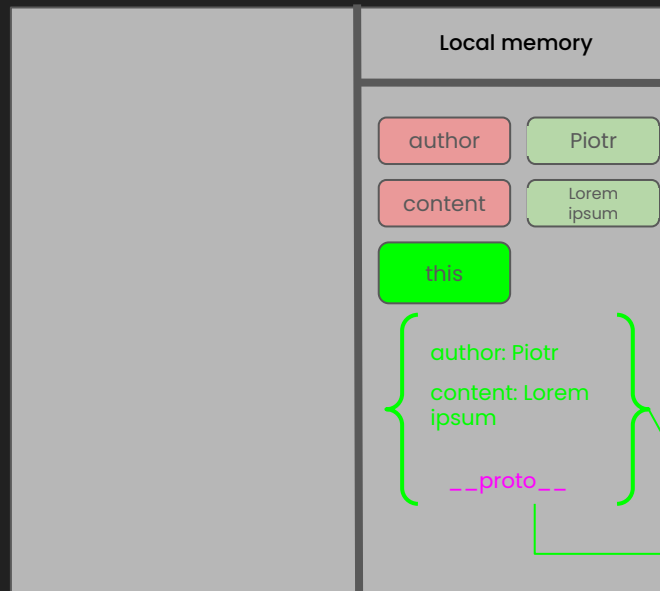
```
1 function createComment(author, content) {
2     this.author = author;
3     this.content = content;
4 }
5
6 createComment.prototype.log = function () {
7     console.log(`${this.content} Author: ${this.author}`);
8 }
9
10 function makeReply(author, content, commentId) {
11     createComment.call(this, author, content);
12     this.commentId = commentId;
13 }
14
15 makeReply.prototype.logParentId = function () {
16     console.log(`Parent comment ID: ${this.commentId}`);
17 }
18
19 Object.setPrototypeOf(makeReply.prototype, createComment.prototype);
20
```

```

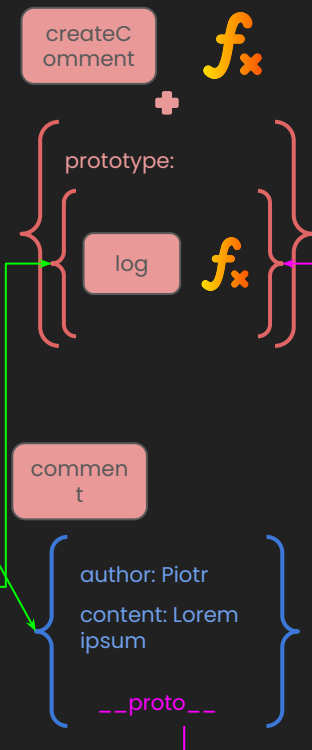
1 function createComment(author, content) {
2   this.author = author;
3   this.content = content;
4 }
5
6 createComment.prototype.log = function () {
7   console.log(`${this.content} Author: ${this.author}`);
8 }

```

comment = new createComment("Piotr", "Lorem ipsum")



Global memory




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

```
reply = new makeReply("Piotr", "Lorem ipsum", "123")
```

createComment.call(this, author, content)

Local memory

author	Piotr
content	Lorem ipsum
this	

Local memory

author	Piotr
content	Lorem ipsum
commentId	123
this	

author: Piotr
content: Lorem ipsum
commentId: 123
__proto__

Global memory

createComment

f_x

prototype:

log

f_x

makeReply

f_x

prototype:

logParentId

f_x

reply

author: Piotr
content: Lorem ipsum
commentId: 123

__proto__

```
10 function makeReply(author, content, commentId) {  
11   createComment.call(this, author, content);  
12   this.commentId = commentId;  
13 }  
14  
15 makeReply.prototype.logParentId = function () {  
16   console.log(`Parent comment ID: ${this.commentId}`);  
17 }  
18  
19 Object.setPrototypeOf(makeReply.prototype, createComment.prototype);
```

ES6 – classes

- Refactor the code with ES6 classes

example.js

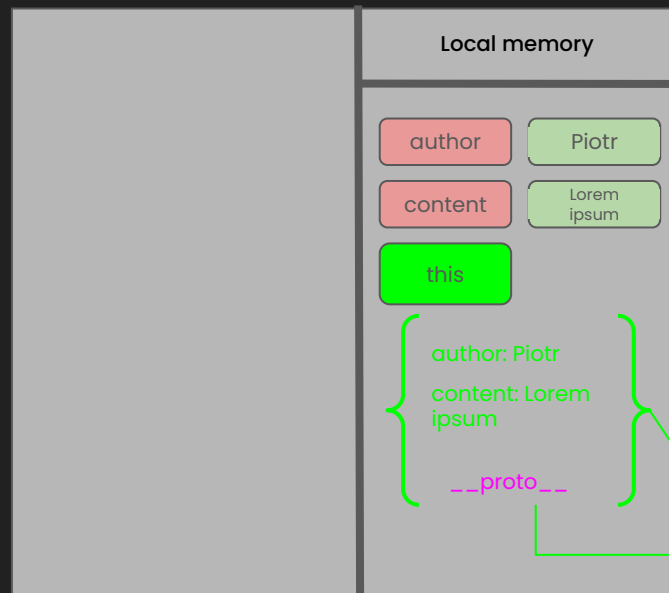
```
1 class createComment {
2   constructor(author, content) {
3     this.author = author;
4     this.content = content;
5   }
6
7   log() {
8     console.log(`${this.content} Author: ${this.author}`);
9   }
10 }
11
12 class makeReply extends createComment {
13   constructor(author, content, commentId) {
14     super(author, content);
15     this.commentId = commentId;
16   }
17
18   logParentId() {
19     console.log(`Parent comment ID: ${this.commentId}`);
20   }
21 }
22
```

```

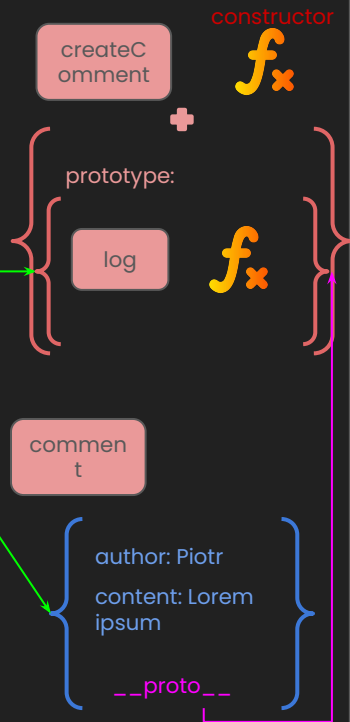
1 class createComment {
2   constructor(author, content) {
3     this.author = author;
4     this.content = content;
5   }
6
7   log() {
8     console.log(`${this.content} Author: ${this.author}`);
9   }
10 }

```

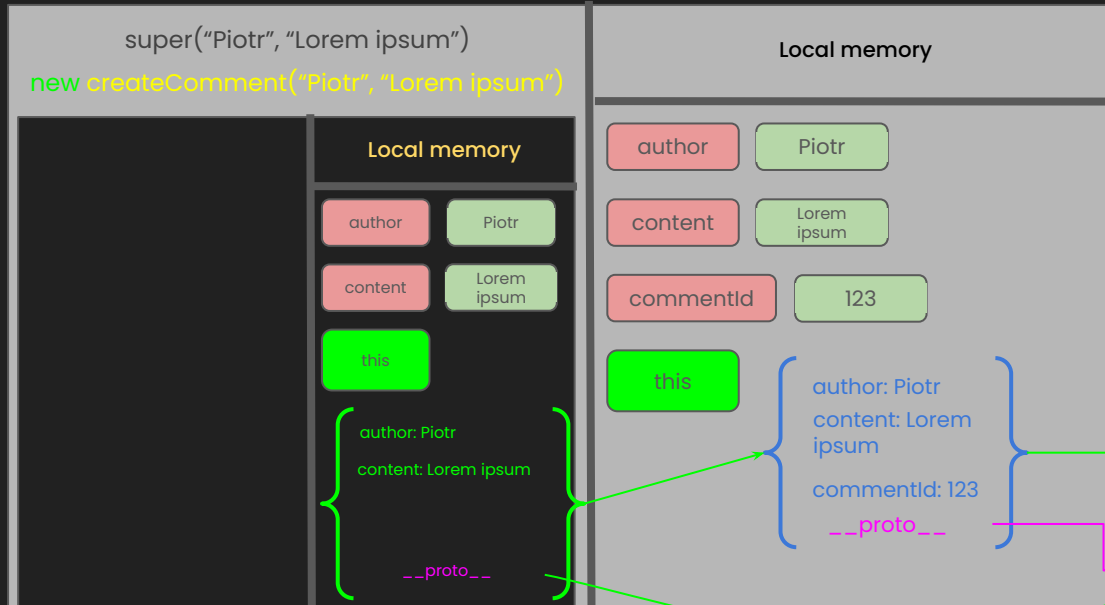
comment = new createComment("Piotr", "Lorem ipsum")



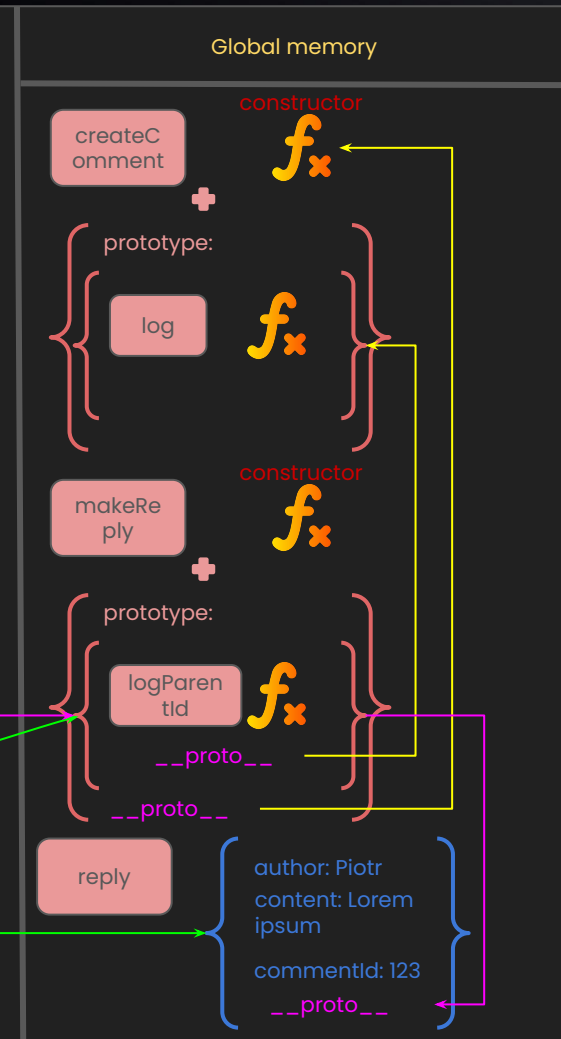
Global memory



reply = new makeReply("Piotr", "Lorem ipsum", "123")



```
12 class makeReply extends createComment {
13   constructor(author, content, commentId) {
14     super(author, content);
15     this.commentId = commentId;
16   }
17
18   logParentId() {
19     console.log(`Parent comment ID: ${this.commentId}`);
20   }
21 }
```



HOMEWORK

USE CLASSES

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