the shortcut

Fullstack Web Development Tutorial Lesson 19

Today’s lesson will cover

- Prototypes, inheritance

the shortcut

JavaScript fundamentals

Prototypal inheritance

- In JavaScript, all objects have a hidden `[[Prototype]]` property that's either another object or `null`.
- We can use `obj.__proto__` to access it (a historical getter/setter, there are other ways, to be covered soon).
- The object referenced by `[[Prototype]]` is called a "prototype".
- If we want to read a property of `obj` or call a method, and it doesn't exist, then JavaScript tries to find it in the prototype.
- Write/delete operations act directly on the object, they don't use the prototype (assuming it's a data property, not a setter).
- If we call `obj.method()`, and the `method` is taken from the prototype, `this` still references `obj`. So methods always work with the current object even if they are inherited.
- The `for..in` loop iterates over both its own and its inherited properties. All other key/value-getting methods only operate on the object itself.

Exercise

- Use `__proto__` to assign prototypes in a way that any property lookup will follow the path: `pockets` → `bed` → `table` → `head`. For instance, `pockets.pen` should be `3` (found in `table`), and `bed.glasses` should be `1` (found in `head`).

```
let head = {
  glasses: 1
};

let table = {
  pen: 3
};

let bed = {
  sheet: 1,
  pillow: 2
};

let pockets = {
  money: 2000
};
```

F.prototype

- The `F.prototype` property (don’t mistake it for `[[Prototype]]`) sets `[[Prototype]]` of new objects when `new F()` is called.
- The value of `F.prototype` should be either an object or `null`: other values won’t work.
- The `"prototype"` property only has such a special effect when set on a constructor function, and invoked with `new`.
- By default all functions have `F.prototype = { constructor: F }`, so we can get the constructor of an object by accessing its `"constructor"` property.

Native prototypes

- All built-in objects follow the same pattern:
  - The methods are stored in the prototype (`Array.prototype`, `Object.prototype`, `Date.prototype`, etc.)
  - The object itself stores only the data (array items, object properties, the date)
- Primitives also store methods in prototypes of wrapper objects: `Number.prototype`, `String.prototype` and `Boolean.prototype`. Only `undefined` and `null` do not have wrapper objects
- Built-in prototypes can be modified or populated with new methods. But it’s not recommended to change them. The only allowable case is probably when we add-in a new standard, but it’s not yet supported by the JavaScript engine

Prototype methods

Modern methods to set up and directly access the prototype are:

- Object.create(proto[, descriptors]) – creates an empty object with a given `proto` as `[[Prototype]]` (can be `null`) and optional property descriptors.
- Object.getPrototypeOf(obj) – returns the `[[Prototype]]` of `obj` (same as `__proto__` getter).
- Object.setPrototypeOf(obj, proto) – sets the `[[Prototype]]` of `obj` to `proto` (same as `__proto__` setter).

Other methods:

- Object.keys(obj) / Object.values(obj) / Object.entries(obj) – returns an array of enumerable own string property names/values/key-value pairs.
- Object.getOwnPropertySymbols(obj) – returns an array of all own symbolic keys.
- Object.getOwnPropertyNames(obj) – returns an array of all own string keys.
- Reflect.ownKeys(obj) – returns an array of all own keys.
- obj.hasOwnProperty(key): returns `true` if `obj` has its own (not inherited) key named `key`.
- All methods that return object properties (like `Object.keys` and others) – return “own” properties. If we want inherited ones, we can use `for..in`.

Exercise

- There's an object `dictionary`, created as `Object.create(null)`, to store any `key/value` pairs.
- Add method `dictionary.toString()` into it, that should return a comma-delimited list of keys. Your `toString` should not show up in `for..in` over the object.

```
let dictionary = Object.create(null);

// your code to add dictionary.toString method

// add some data
dictionary.apple = "Apple";
dictionary.__proto__ = "test"; // __proto__ is a regular property key here

// only apple and __proto__ are in the loop
for(let key in dictionary) {
  console.log(key); // "apple", then "__proto__"
}

// your toString in action
console.log(dictionary); // "apple,__proto__"
```

the shortcut

Self Study Assignments

To Dos

- Try to use Github pages, Netlify or Heroku to showcase live projects
- Create a game of Rock, Paper and Scissors using JS which works on console, or with interactive UI using HTML, CSS and JS however you prefer ***(If you are working on your own project where you are using JS already, feel free to ignore this task but please share the project update with Lena.)***
- Continue freecodecamp (FCC) Javascript. Ideally finish before we resume after summer.
- Continue with FCC HTML, CSS lessons. Ideally finish all the lessons by end of this month.
- If you believe FCC exercises aren't the best for you as in if you are quite advanced already, please start working on your own project and reach out to mentors for help if needed.