Лабораторная работа № 12. Коллекции Map и Set, метод Object.create, параметры по умолчанию, замыкание

Цель: изучить коллекции Map и Set, создание объекта с помощью метода Object.create, параметры по умолчанию, замыкание.

Теория

Коллекция Map

Map – это коллекция для хранения пар «*ключ-значение*». Ключ может быть абсолютно любым. Ключи сохраняются как есть, без преобразования типов (напоминание: в объекте ключами могут быть только строки и symbol).

Проверка ключей на равенство выполняется при помощи **Object.is()** (в частности, NaN считается равным NaN). Примеры активирующих действий JavaScript:

- size – возвращает текущее количество элементов
- clear() – очищает коллекцию от всех элементов
- delete(key) – пытается удалить пару по ключу *key*. Возвращает *true*, если пара была в *Map*
- entries() – итератор из массивов [*ключ, значение*] в порядке вставки пар
- forEach() – выполняет аргумент-функцию для каждой пары
- get(key) – получает элемент по ключу *key* (нет ключа – возвращает *undefined*)
- has(key) – проверяет, есть ли ключ *key* в коллекции (нет ключа – возвращает *false*)
- keys() – итератор ключей в порядке вставки пар
- set(key, value) – записывает по ключу *key* значение *value*
- values() – итератор значений в порядке вставки пар

Метод **Object.keys** возвращает массив строковых элементов, соответствующих именам перечисляемых свойств, найденных непосредственно в самом объекте. Порядок свойств такой же, как и при ручном перечислении свойств в объекте через цикл.

```
let map = new Map();
map.set('1', 'str');    // ключ - строка
map.set(1, 'num');      // ключ - число
map.set(true, 'bool');  // ключ - булевое значение
alert(map.size);        // 3 - количество пар
```

```
// в обычном объекте это было бы одно и то же
 alert(map.get(1));      // num
 alert(map.get('1'));    // str
 map.delete('1');
 alert(map.has('1') ? 'Yes' : 'No');    // No
```

Коллекция Set

Set – множество неповторяющихся значений. Коллекция Set имеет схожесть с Map: ключом может быть абсолютно произвольное значение, которое сохраняется без преобразования типов. Проверка ключей на равенство в Set осуществляется аналогично проверке в Map.

- size – возвращает текущее количество значений в множестве
- add(value) – добавляет значение *value* (если его там нет), возвращает экземпляр *Set*
- clear() – очищает коллекцию от всех значений
- delete(value) – пытается удалить значение *value*. Возвращает *true*, если значение было в множестве на момент вызова
- entries() – итератор из массивов [*ключ, значение*] в порядке вставки значений
- forEach() – выполняет аргумент-функцию для каждого значения
- has(value) – проверяет, есть ли значение *value* в множестве (нет значения – возвращает *false*)
- keys() – итератор значений в порядке вставки (то же, что и values())
- values() – итератор значений в порядке вставки пар

При повторных вызовах *set.add()* с одним и тем же значением ничего не происходит. С помощью этой особенности и получается, что каждое значение встречается лишь единожды.

```
// конструктору Set можно передать итерируемый объект
  let set = new Set([1, '2', false]);
  set.add(NaN).add(1);
// 1, 2, false, NaN
  set.forEach((value, key, set) => alert(key));
  set.delete(false);
// 1, 2, NaN
  for (let value of set.values()) alert(value);
```

Object.create

Объекты также можно создавать с помощью метода **Object.create**. Этот метод очень удобен, так как позволяет вам указывать объект прототип для нового вашего объекта без определения функции конструктора.

```
var car = {
  type: 'coupe',
  displayType:function(){
    document.write(this.type + '<br>');
  }
};
var audi = Object.create(car);
audi.displayType();
var Toyota = Object.create(car);
Toyota.type='sedan';
Toyota.displayType();
```

Параметры по умолчанию

В JavaScript стандартное значение параметров функций *undefined*. Однако, в некоторых случаях может быть полезно задать иное значение по умолчанию. Для таких ситуаций предназначены параметры по умолчанию.

Ранее для проверки и задания стандартных значений использовалось тело функции, где параметры сравнивались с *undefined*. В приведённом ниже примере, для параметра b не передано значение при вызове функции, в результате он будет иметь значение *undefined* и результатом вычисления *a+b* в функции *add()* может быть значение *NaN*. Однако, этот случай отслеживается на второй строке примера:

```
function add(a,b){
  b = typeof b! == 'undefined' ? b : 1;
  document.write(a+b);
}
add(2);
```

С параметрами по умолчанию проверка их значений в теле функции более не требуется. Вам достаточно указать *1* в качестве значения по умолчанию для параметра *b* в заголовке функции:

```
function add(a, b = 1){
  document.write(a+b);
}
add(2);
```

Замыкание

Если функция *F()* возвращает свою вложенную функцию *g()*, то все переменные и аргументы функции из *scope F()* (и их значения) будут доступны в *g()*. Это явление называется замыканием (closure).

```
function func(param) {
  var clos = param;
  function res(x) { return x + clos; }
```

```
  return res;
}
let f1 = func(100);
alert(f1(10));    // '110'
let f2 = func(200);
alert(f2(10));    // '210'
```

Задания к лабораторной работе № 12

Задание 1. Телефонная книжка. Создайте коллекцию *Map* с ключами ”1”, ”2”, ”3”, ”4”, ”5”. Каждому ключу должен соответствовать номер телефона. Сделайте функцию, которая при вводе в окно *confirm* ключа выводит соответствующий номер телефона.

Задание 2. Создайте объект *Set*. Добавьте в него элементы 1, 3, 5 с помощью метода *add()*. Вычислите его размер с помощью метода *size()*. Используйте метод *has()* для проверки элементов 1, 3, 5 в объекте *Set*. Какой тип данных возвращает метод *has()*?

Задание 3. Создайте объект с именем proto. Задайте свойства: имя, фамилия, отчество. Добавьте метод, выводящий содержимое этих свойств на экран. Используйте объект proto как прототип и создайте новый объект с именем proto2 (использовать *Object.create()*). Измените свойство имя и вызовите метод вывода содержимого свойств на экран.

Задание 4. Сделайте функцию, которая получает имя пользователя и выводит на экран 'Привет, Имя' (вместо Имя - полученное параметром имя). По умолчанию (то есть если не передать параметр) функция должна выводить 'Привет, Аноним'.

Задание 5. Даны кнопки. Привяжите к каждой кнопке событие по клику, которое будет считать количество нажатий по кнопке и выводить его в текст кнопки. Количество нажатий для каждой кнопки должно хранится в замыкании.