**Тема:** Порождающие паттерны: «Строитель», «Синглтон», «Пул».

**«Пул и Синглтон»**

Описание:

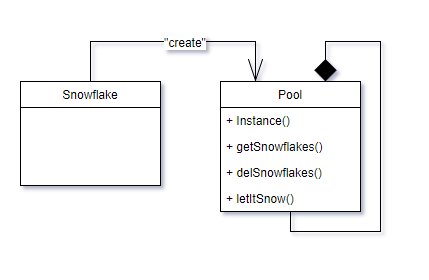
Желательно, чтобы все многократно используемые объекты, свободные в некоторый момент времени, хранились в одном и том же пуле объектов. Тогда ими можно управлять на основе единой политики. Для этого класс Object Pool проектируется с помощью паттерна Singleton.

Если у нас много однотипных объектов, создание и уничтожение которых занимает много ресурсов, удобно использовать этот паттерн. Пулы объектов (известны также как пулы ресурсов) используются для управления кэшированием объектов. Клиент, имеющий доступ к пулу объектов, может избежать создания новых объектов, просто запрашивая в пуле уже созданный экземпляр. Пул объектов может быть растущим, когда при отсутствии свободных создаются новые объекты или c ограничением количества создаваемых объектов. Уничтожение объекта заменяется на его возврат в пул для дальнейшего использования.

Пример реализации:

Создадим пул снежинок. Для этого нам понадобиться, собственно, класс самого пула и класс снежинки. Класс пул реализуем с использованием паттерна синглтон, т.к. нам нужен один пул снежинок, не более.

Диаграмма классов:



Исходный код:

Pool.js

**class** Pool {  
 constructor(count = 50) {  
 **if** (Pool.*instance*) {  
 // console.log('Pool already exists!');  
 //Вместо обычного лога используем наш логгер  
 Log.*log*("Pool already exists!", 'Error');  
 **throw** (**new** Error('Pool already exists!'));  
 }  
 **this**.Instance(count);  
 **this**.getSnowflake = **this**.getSnowflake.bind(**this**);  
 **this**.delSnowflakes = **this**.delSnowflakes.bind(**this**);  
 **this**.letItSnow = **this**.letItSnow.bind(**this**);  
 }  
  
 **static set** *instance*(val) {  
 **this**.\_instance = val;  
 }  
  
 **static get** *instance*() {  
 **return this**.\_instance;  
 }  
  
 Instance(count = 50) {  
 **this**.used = [];  
 **this**.unused = [];  
 // for (let i = 0; i < count; i++) {  
 // this.getSnowflake();  
 // }  
 **let** i = 0;  
 **const** getOneSnowflake = () => {  
 **this**.getSnowflake();  
 **if** (i++ < count) {  
 timer = setTimeout(getOneSnowflake, 500);  
 } **else** {  
 clearTimeout(timer);  
 }  
 };  
 **let** timer = setTimeout(getOneSnowflake, 500);  
 Pool.*instance* = **this**;  
 }  
  
 getSnowflake(params = {}) {  
 **let** ret = **this**.unused.pop() || (**new** Snowflake()).htmlNode;  
 **this**.used.push(ret);  
 document.body.appendChild(ret);  
 **return** ret;  
 }  
  
 delSnowflakes(snowflakes) {  
 **let** idx = **this**.used.indexOf(snowflakes),  
 ret = **this**.used.splice(idx, 1)[0];  
 ret.style.top = '0px';  
 ret.style.left = rand(0, document.documentElement.clientWidth - 30) + 'px';  
 **this**.unused.push(ret);  
 document.body.removeChild(ret);  
 **return** ret;  
 }  
  
 letItSnow() {  
 **const** tick = () => {  
 **try** {  
 **this**.used.forEach((curr) => {  
 curr.style.top = (parseInt(curr.style.top, 10) + rand(-1, 10)) + 'px';  
 curr.style.left = ((parseInt(curr.style.left, 10) || 0) + rand(-10, 10)) + 'px';  
 **if** (parseInt(curr.style.top, 10) > document.documentElement.clientHeight - 30 || parseInt(curr.style.left, 10) > document.documentElement.clientWidth - 30 || parseInt(curr.style.left, 10) < 0) {  
 **this**.delSnowflakes(curr);  
 **this**.getSnowflake();  
 }  
 });  
 } **catch** (e) {  
 console.log(e);  
 **debugger**;  
 // clearTimeout(timer);  
 }  
 timer = setTimeout(tick, 100);  
 };  
  
 //start  
 // eslint-disable-next-line  
 **let** timer = setTimeout(tick, 100);  
 }  
}

Snowflake.js

**class** Snowflake {  
 constructor(params = {}) {  
 **this**.htmlNode = document.createElement('div');  
 **this**.htmlNode.style.backgroundImage = 'url("/snowflake.png")';  
 **this**.htmlNode.style.backgroundSize = 'contain';  
 **this**.htmlNode.style.width = params.width || '30px';  
 **this**.htmlNode.style.height = params.height || '30px';  
 **this**.htmlNode.style.display = 'block';  
 **this**.htmlNode.style.position = 'absolute';  
 **this**.htmlNode.style.top = '0px';  
 **this**.htmlNode.style.left = rand(0, document.documentElement.clientWidth - (params.width || 30)) + 'px';  
 console.log('Snowflake did created');  
 }  
}

Снимок экрана:

