Numba's JSON
Шорткаты

Необходимые пространства имен:

```
using Numba.Data.Json.Engine;
using Numba.Data.Json.Engine.DataTypes;
using Numba.Data.Json.Engine.Extentions;
using Numba.Data.Json.Extensions;
```

Создать примитивные данные:

```
JsonBool jBool = true;
JsonInt jInt = 1234;
JsonChar jChar = '#';
JsonString jString = "Hello Numba's JSON!";
JsonDecimal jDecimal = decimal.MaxValue;
```

Создать JsonObject используя синтаксис инициализации объектов:

```
JsonObject jObject = new JsonObject() { { "name", "Zaur" }, { "age", 25 } };
```

Добавление, удаление и вставка полей в JsonObject:

```
jObject.Add("married", true);
jObject.RemoveField("married");
jObject.Insert(1, "growth", 178.7f);
```

Чтение и запись значений в поля:

```
int age = jObject.GetInt("age").Value;
jObject.SetInt("age", age + 1);
```

Тест на JsonNull тип:

```
jObject.Add("isnotnullfield", new JsonNull());
bool isJsonNullField = jObject.IsNullField("isnotnullfield");
```

Работа с JsonNumber:

```
jObject.Add("number", new JsonNumber("1"));
byte number = jObject.GetNumber("number").ToByte();
```

Получение строковых данных объекта:

```
string stringData = jObject.ToString();
```

Создать JsonArray используя синтаксис инициализации объекта:

```
JsonArray jArray = new JsonArray() { 1, true, '#', 3.14f, 6.18d, decimal.MaxValue, "Hi" };
```

Цикл по элементам JsonArray:

```
foreach (IJsonValue value in jArray) { }
```

Добавление, удаление и вставка элемента:

```
jArray.Add(long.MinValue);
jArray.Remove(true);
jArray.Insert(1, false);
```

Чтение и запись элемента:

```
IJsonValue jValue = jArray[3];
jArray[3] = (JsonFloat)(jValue.AsFloat().Value / 2f);
```

Проверка на содержание:

```
bool containSharp = jArray.Contains(new JsonChar('#'));
```

Сериализация JsonArray:

```
string stringData = jArray.ToString();
```

Составные объекты:

```
jObject.Add("jArray", jArray);
```