Lessons from Robots about JSON in the JavaScript and C# programming languages.  
Example JSON strings for human controller software to communicate with robots.  
Note that these examples are oversimplified in that we focus on the JSON by including literals where in real life we need variable values.  
eg the code below says:  
signalObject.commanderid = 14;  
In the real life version, the commanderid can be any number between 10 and 19 and the code reads it from a screen input field:  
signalObject.commanderid = parseInt(document.getElementById("CommanderId").value);

Example JSON robot remote control signal that the human sends from a laptop and the robot receives:   
{"categoryid":1,"commanderid":14,"robotid":114,"accesskey":"demo-access-key", "message": "turn right", xdata":{"vx1x1":20,"vx1x2":50,"vx2x1":-10,"vx2x2":-25}}  
Note that we use the keywords "signal", "request" or "response" for the complete JSON string, depending on context,   
and "message" as one of the child elements, usually for descriptive text.

|  |  |
| --- | --- |
| JavaScript | C# |
| //Serialising = working with an object  //then converting that to a JSON string //usually for sending to another system  var signalObject = new Object();  signalObject.categoryid = 1; //commander sends to robot  signalObject.commanderid = 14;  signalObject.robotid = 114;  signalObject.accesskey = "demo-access-key";  signalObject.message = "turn right";  //child object inside a parent object  //create the child first  var xdataObject = new Object();  xdataObject.vx1x1 = 20;  xdataObject.vx1x2 = 50;  xdataObject.vx2x1 = -10;  xdataObject.vx2x2 = -25;  //add the child into the parent  signalObject.xdata = xdataObject;  //serialise the object into a string which is useful for sending  signal = JSON.stringify(signalObject);  //send the signal over the Internet using a "websocket"  connection.invoke("XSignal", signal); | //Serialising = working with an object  //then converting that to a JSON string //usually for sending to another system  Signal signalObject = new Signal();  signalObject.categoryid = 1; //commander sends to robot  signalObject.commanderid = 14;  signalObject.robotid = 114;  signalObject.accesskey = "demo-access-key";  signalObject.message = "turn right";  //child object inside a parent object  //create the child first  XData xdataObject = new XData();  xdataObject.vx1x1 = 20;  xdataObject.vx1x2 = 50;  xdataObject.vx2x1 = -10;  xdataObject.vx2x2 = -25;  //add the child into the parent  signalObject.xdata = xdataObject;  //serialise the object into a string which is useful for sending  string signal = JsonConvert.SerializeObject(signalObject);  //send the signal over the Internet as a data web page  return signal;  //NOTE that C# needs support code to do this, ref the next page |

|  |  |
| --- | --- |
| //JavaScript - more details  //This code would almost always appear in a function (method) | //C# - more details  //This code needs to be structured inside a method (function)  //in a class  //At the top of the class we need this line of code to bring in  //JSON processing created separately by James Newton-King.  //"using" is like the "import" statement  //which we see in some other programming languages  using Newtonsoft.Json;  //C# is "strongly typed", also called "strictly typed"  //which means that we need to define "Signal" and "XData".  //These would often appear as 2 separate "class files"  //in the same project.  public class Signal  {  public int categoryid;  public int commanderid;  public int robotid;  public string accesskey;  public string message;  public XData xdata;  }  public class XData  {  public int vx1x1;  public int vx1x2;  public int vx2x1;  public int vx2x2;  } |

|  |  |
| --- | --- |
| //Deserialising = receive data as a JSON string  //then convert that to an object for data processing  var signalObject = JSON.parse(signal);  var categoryId = signalObject.categoryid;  var commanderId = signalObject.commanderid;  var robotId = signalObject.robotId;  var accessKey = signalObject.accesskey;  var message = signalObject.message;  var vx1x1 = signalObject.xdata.vx1x1;  var vx1x2 = signalObject.xdata.vx1x2;  var vx2x1 = signalObject.xdata.vx2x1;  var vx2x2 = signalObject.xdata.vx2x2; | //Deserialising = receive data as a JSON string  //then convert that to an object for data processing  //"best practice" includes use of classes like the ones above. //The following code needs class Signal and class XData  //to be available in this project  Signal signalObject  = JsonConvert.DeserializeObject<Signal>(signal);  int categoryId = signalObject.categoryid;  int commanderId = signalObject.commanderid;  int robotId = signalObject.robotid;  string accessKey = signalObject.accesskey;  string message = signalObject.message;  int vx1x1 = signalObject.xdata.vx1x1;  //----------------------------------------------------  //"quick and dirty" method  //without needing a class Signal and class XData  //Good to use this if the signal content can vary so we cannot  //predict it closely enough to write classes that always work  dynamic signalObject = JsonConvert.DeserializeObject(signal);  int categoryId = (int)signalObject.SelectToken("categoryid");  int commanderId = (int)signalObject.SelectToken("commanderid");  int robotId = (int)signalObject.SelectToken("robotid");  string accessKey = (string)signalObject.SelectToken("accesskey"); string message = (string)signalObject.SelectToken("message");  int vx1x1  = (int)signalObject.SelectToken("xdata").SelectToken("vx1x1");  int vx1x2  = (int)signalObject.SelectToken("xdata").SelectToken("vx1x2");  int vx2x1  = (int)signalObject.SelectToken("xdata").SelectToken("vx2x1");  int vx2x2  = (int)signalObject.SelectToken("xdata").SelectToken("vx2x2"); |