JSON Canonicalization Scheme

Internet-Draft:  

On-line application for testing/evaluation:  
https://mobilepki.org/jws-jcs
Motivation/Rationale

eyJhbGciOiJFUzI1NiJ9.eyJvdGhlclBhIn0.1FtcRzGP7lUzkzBYe9ko5O14T0jZWFWX8eIrYi

versus

{
   "statement": "Hello signed world!",
   "otherProperties": [2e+3, true],
   "signature": "eyJhbGciOiJFUzI1NiJ9..1FtcRzGP7lUzkzXfyy3Gw"
}

- Keeps JSON format for documentation, logging, debug, embedding, countersigning
- Signed messages rather than signature containers embedding messages

Problem Statement: Is this really doable in a reasonable way?
Canonicalization of Native JSON Elements

```json
{
    "numbers": [333333333.3333333, 1E30, 4.50, 2e-3, 0.000000000000000000001],
    "string": "\u20ac$\u000F\u000a\u0042\u0022\u005c",
    "literals": [null, true, false]
}
```

Compatible implementations exist for JavaScript, Java, C#, Python, Go and Ruby

**Constraint:** Limits JSON **Number** to IEEE-754 double precision (I-JSON)
Canonical Hashable JSON

Fully canonical JSON would require normalizing data types that are stuffed into JSON Strings like date which does not have a native JSON counterpart.

"timeStamp": "2019-03-23T07:30:00Z"

Since there are almost no real-world applications for canonical JSON outside of cryptography as well as the difficulty creating exact representations for a huge number of data types both from an implementation and standards point of view, the team settled on “hashable” JSON which only deals with native JSON types.

Constraint: Parsed JSON String data MUST be kept “as is” when serialized