Core Algorithms

- Expand – Apply a context to create a document with fully expanded IRIs and values.
- Compact – Use a context to replace IRIs with terms and represent values as strings, where possible.
- Flatten – Remove internal embedding by introducing blank nodes to relate node objects with each other.
- To RDF – Transform JSON-LD to the RDF Abstract Syntax
- From RDF – Transform from the RDF Abstract Syntax into expanded JSON-LD.
- Frame – Apply a *frame document* to structure (and compact) a flattened JSON-LD document using object embedding and filtering.

```
[{
  "@type": ["http://schema.org/Person"],
  "http://schema.org/colleagues": [
    {"@id": "http://www.xyz.edu/students/alicejones.html"},
    {"@id": "http://www.xyz.edu/students/bobsmith.html"}
  ],
  "http://schema.org/image": [{
    "@id": "http://localhost:9393/examples/schema.org/janedoe.jpg"
  }],
  "http://schema.org/name": [{"@value": "Jane Doe"}],
  "http://schema.org/url": [{"@id": "http://www.janedoe.com"}]
 }]
```

```
@prefix schema: <http://schema.org/> .

[
  a schema:Person;
  schema:colleagues <http://www.xyz.edu/students/alicejones.html>,
    <http://www.xyz.edu/students/bobsmith.html>;
  schema:image <http://localhost:9393/examples/schema.org/janedoe.jpg>;
  schema:name "Jane Doe";
  schema:url <http://www.janedoe.com>
] .
```

JSON-LD 1.1 Motivation

- While JSON-LD 1.0 was widely used, there remained frustration points:
  - Multiple values require indexing arrays to find – introduce improved maps
  - In-the-wild JSON often separates the properties of an entity from the entity itself – introduce property nesting.
  - The list structure was only one level, specs such as GeoJSON require multiple levels – introduce recursive lists.
  - The need to embed contexts within nested objects inhibits easy reuse, and is not supported in framing – introduce scoped contexts.