SPARKQL Queries and responses for js ontology

1.
SELECT ?class ?superClass WHERE { ?class rdf:type owl:Class . OPTIONAL { ?class rdfs:subClassOf ?superClass . } }

Result:

```
class            | superClass
-----------------|---------------
:LetVariable     | :Variable
:ConstVariable   | :Variable
:ForLoop         | :Loop
:Loop            | :ControlStructure
:Promise         | :ObjectType
```

Fact obtained: A ForLoop is a type of Loop, which is a ControlStructure.

2.
SELECT ?property ?domain ?range WHERE { ?property rdf:type owl:ObjectProperty . ?property rdfs:domain ?domain . ?property rdfs:range ?range . }

Result:

```
property        | domain                  | range
----------------|-------------------------|---------------
:hasParameter   | :Function               | :Variable
:returns        | :Function               | :DataType
:contains       | :Object, :Class, :Module | :Property, :Method, :Function, :Variable
:extends        | :Class                  | :Class
```

Fact obtained: The hasParameter property connects a Function to a Variable.

3.
SELECT ?property ?domain ?range WHERE { ?property rdf:type owl:DatatypeProperty . ?property rdfs:domain ?domain . ?property rdfs:range ?range . }

Result:

property	domain	range
:hasName	:Variable, :Function, :Class, :Module	xsd:string
:hasType	:Variable, :Property	:DataType
:hasValue	:Variable, :Property	xsd:anySimpleType
:hasScope	:Variable	xsd:string
:hasEventType	:Event	xsd:string

Fact obtained: The hasType property assigns a DataType to a Variable or Property.

4.

SELECT ?class ?property ?constraint ?value WHERE { ?class rdfs:subClassOf ?restriction . ?restriction owl:onProperty ?property . { ?restriction owl:cardinality ?value } UNION { ?restriction owl:minCardinality ?value } UNION { ?restriction owl:hasValue ?value } UNION { ?restriction owl:someValuesFrom ?value } BIND(IF(EXISTS { ?restriction owl:cardinality ?v }, "cardinality", IF(EXISTS { ?restriction owl:minCardinality ?v }, "minCardinality", IF(EXISTS { ?restriction owl:hasValue ?v }, "hasValue", "someValuesFrom"))) AS ?constraint) }

Result:

class	property	constraint	value
:Variable	:hasType	cardinality	1
:Function	:hasParameter	minCardinality	0
:AsyncFunction	:returns	hasValue	:Promise
:ConstVariable	:hasValue	cardinality	1
:Closure	:uses	someValuesFrom	:Variable

Fact obtained: Every AsyncFunction must return a Promise.

5.

SELECT ?subClass WHERE { ?subClass rdfs:subClassOf* :ControlStructure . }

Result:

```
subClass
---------------
:ControlStructure
:Loop
:ForLoop
:ForInLoop
:ForOfLoop
:WhileLoop
:DoWhileLoop
:Conditional
:IfStatement
:SwitchStatement
:TernaryOperator
:ExceptionHandling
:TryCatch
:Throw
```

Fact obtained: ForLoop, WhileLoop, and IfStatement are all types of ControlStructure.

6.

SELECT ?property ?range WHERE { ?property rdfs:domain ?domain . ?property rdfs:range ?range . :Function rdfs:subClassOf* ?domain . }

Result:

```
property        | range
----------------|---------------
:hasParameter   | :Variable
:returns        | :DataType
:uses           | :Variable, :DataType
:contains       | :Property, :Method, :Function, :Variable
:hasName        | xsd:string
:hasPrototype   | :Prototype
```

Fact obtained: A Function can have parameters (hasParameter) that are Variables.

7.
SELECT ?individual ?type WHERE { ?individual rdf:type ?type . FILTER(?type != owl:NamedIndividual) }

Result:

individual	type
:myFunction	:Function
:x	:LetVariable
:myAsyncFunction	:AsyncFunction
:myObject	:CustomObject

Fact obtained: myFunction is an instance of Function.

8.
SELECT ?class ?property WHERE { ?class rdfs:subClassOf ?restriction . ?restriction owl:onProperty ?property . ?restriction owl:hasValue :Promise . }

Result:

class	property
:AsyncFunction	:returns

Fact obtained: An AsyncFunction always returns a Promise.