Web Assembly
pour le
Web sémantique

Julian BRUYAT

M2 Intelligence Artifificielle – Université Lyon 1
LIRIS – Équipe TWEAK
07/09/2020

Contexte REPID

- Stage s'inscrivant dans le projet REPID
- « Un raisonneur *efficace*, *portable*, incrémental et distribué »
  - Inferray, raisonneur efficace en Java

    ↓ Stage de T. Bourg
  - Sophia, interface commune aux implémentations en Rust de bibliothèques RDF

    ↓ Ce travail
  - HyLaR, raisonneur distribué en Javascript

Quels sont les apports et inconvénients de l'utilisation de Rust pour la conception d'une bibliothèque RDF compilée en Web Assembly visant à être utilisée en Javascript ?

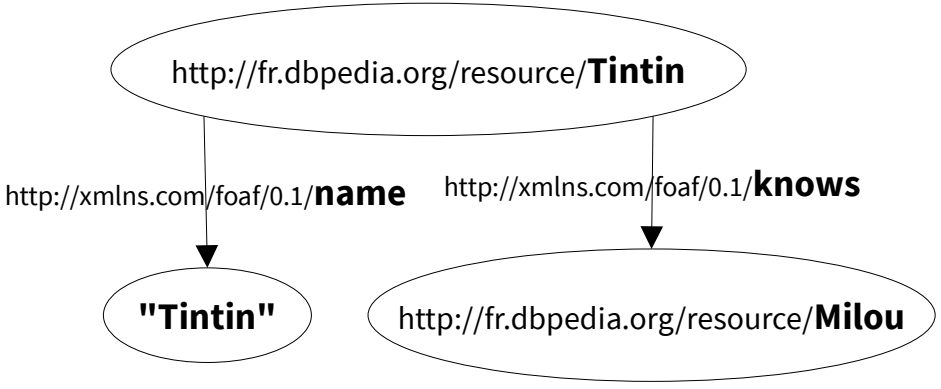
Plan

- **Contextualisation**
  - **Le Web sémantique**
  - **Rust et le Web Assembly**
- **Propositions et résultats**
  - **Dataset boisé**
  - **Adaptateur naïf de Sophia vers RDF.JS**
  - **Autres approches d'exportation**
  - **Évaluation dans le cadre de SPARQL**
- **Conclusion**

Contextualisation

Web sémantique – En théorie

- Web classique : partage de documents
- Web sémantique : partage de données
  - Graphe RDF : Utilisation d'URL pour décrire les ressources et leurs liens



Sujet	*Prédicat*	*Objet*
http://fr.dbpedia.org/resource/**Tintin**	http://xmlns.com/foaf/0.1/**name**	**"Tintin"**
http://fr.dbpedia.org/resource/**Tintin**	http://xmlns.com/foaf/0.1/**knows**	http://fr.dbpedia.org/resource/**Milou**

Un exemple de graphe RDF représenté sous forme de graphe et sous forme d'un tableau de triplés

  - Inférence de triplés : « Est-ce que Milou connaît Tintin ? »
- Dataset : regroupement de graphes, représenté par des quads (SPOG)

Web sémantique – En pratique

- Exemples de bibliothèques pour manipuler des graphes / datasets RDF :
  - Java – Quelques bibliothèques monolithiques : Apache Jena
  - Javascript – Diversités des bibliothèques : N3.js[RDF.JS], Graphy[RDF.JS], rdfstore-js, rdflib.js
    - Interface commune à certaines : RDF.JS (https://rdf.js.org)

```
interface DatasetCore {
  readonly attribute unsigned long  size;
  DatasetCore    add (Quad quad);
  DatasetCore    delete (Quad quad);
  boolean        has (Quad quad);
  DatasetCore    match (optional Term? subject, optional Term? predicate,
                        optional Term? object, optional Term? graph);

  iterable<Quad>;
};
```

Spécification de l'interface DatasetCore de RDF.JS

  - Rust : Sophia, Oxigraph

Web Assembly – Motivations

- Javascript : langage interprété sur navigateur
  - Programmation orientée prototype (~ orientée objet)
  - Également sur serveur avec Node.js (2009)
  - Langage « lent » : Code au format textuel, mise en place de la compilation à la volée, ramasse-miettes, …
- Web Assembly (2015)
  - Exécuté dans une machine virtuelle proche des processeurs actuels
  - Code compilé appelé par Javascript
  - Communication grâce à des appels de fonction modifiant la mémoire linéaire attribuée à Web Assembly et simulant la RAM

Web Assembly – De Rust à Javascript

- Rust : langage bas niveau développé par Mozilla depuis 2010

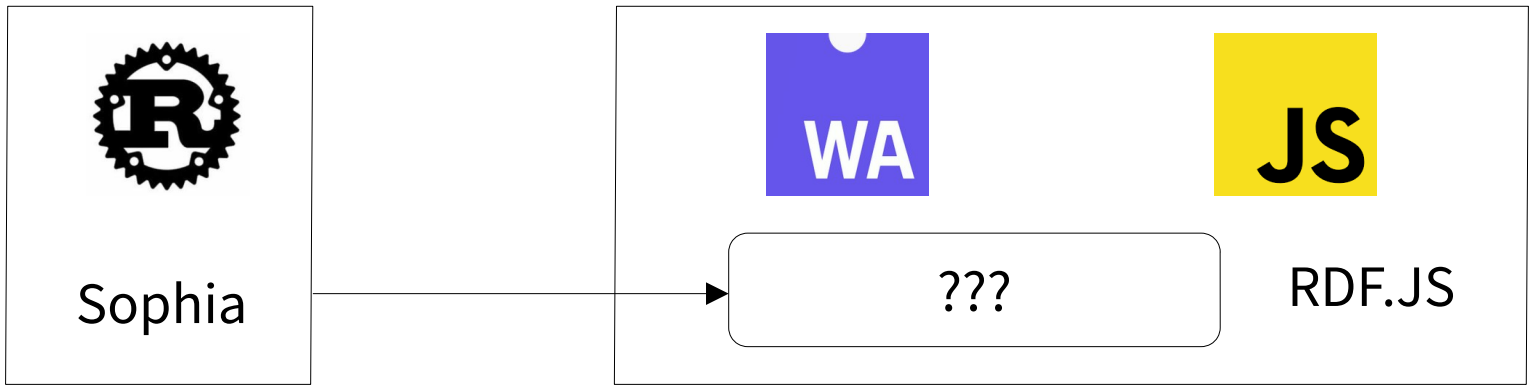
```rust
#[wasm_bindgen]
pub struct Cat { name: String }

#[wasm_bindgen]
impl Cat {
    #[wasm_bindgen (constructor)]
    pub fn new(name: String) -> Cat {
        Cat { name: name }
    }
    pub fn get_cry(&self) -> String {
        format!("{}: meow", self.name)
    }
}
```

```
» let felix = new wasm_bindgen.Cat("Felix");
» console.log(felix.get_cry());
Felix: meow
```

Une bibliothèque écrite en Rust et utilisée en Javascript grâce à wasm-bindgen

Rappel de la problématique



Schématisation de la problématique traitée lors du stage

Quels sont les apports et inconvénients de l'utilisation de Rust pour la conception d'une bibliothèque RDF compilée en Web Assembly visant à être utilisée en Javascript ?

Propositions et résultats

Dataset boisé – Motivations

```
interface DatasetCore {
  // Opérations d'ajouts, supression, parcours
  DatasetCore    match (optional Term? subject, optional Term? predicate,
                        optional Term? object, optional Term? graph);
};
```

Rappel de la définition d'un DatasetCore dans RDF.JS

- Besoins de la méthode match :
  - Trouver tous les quads correspondant à un motif donné
  - Les utiliser pour construire un nouveau dataset
  - De manière efficace
- Dans le contexte de Rust et de Sophia

Dataset boisé – Implémentation

1/ Création d'un dictionnaire de correspondance entre termes et index [HDT]

```
dbpedia:A rdf:type dbpedia:Vowel .
dbpedia:A rdf:type dbpedia:Letter .
dbpedia:B rdf:type dbpedia:Letter .
dbpedia:C rdf:type dbpedia:Letter .
dbpedia:D rdf:type dbpedia:Letter .
```

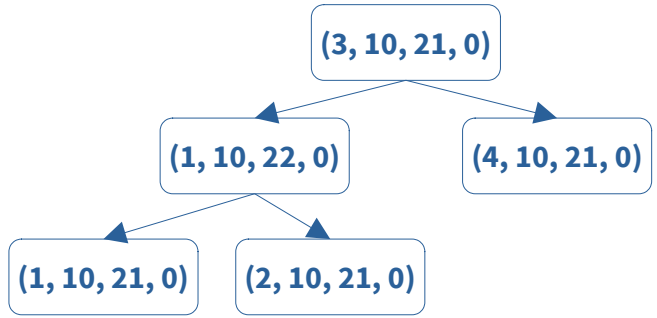
→

```
(1, 10, 21, 0)
(1, 10, 22, 0)
(2, 10, 21, 0)
(3, 10, 21, 0)
(4, 10, 21, 0)
```

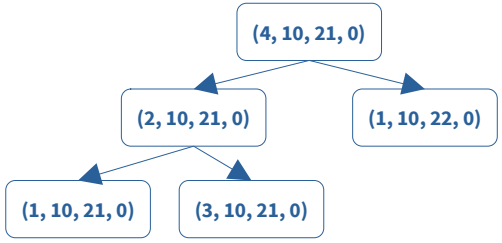
```
1 = dbpedia:A
2 = dbpedia:B
...
10 = rdf:type
21 = dbpedia:Letter
22 = dbpedia:Vowel
0 = Graphe par défaut
```

2/ Création d'un arbre trié

Par défaut : SPOG



3/ A la demande : Création d'arbres OGPS, GPSO, POGS, GSPO, OSGP
Permet de répondre à n'importe quel motif de requete [RDF-4X]

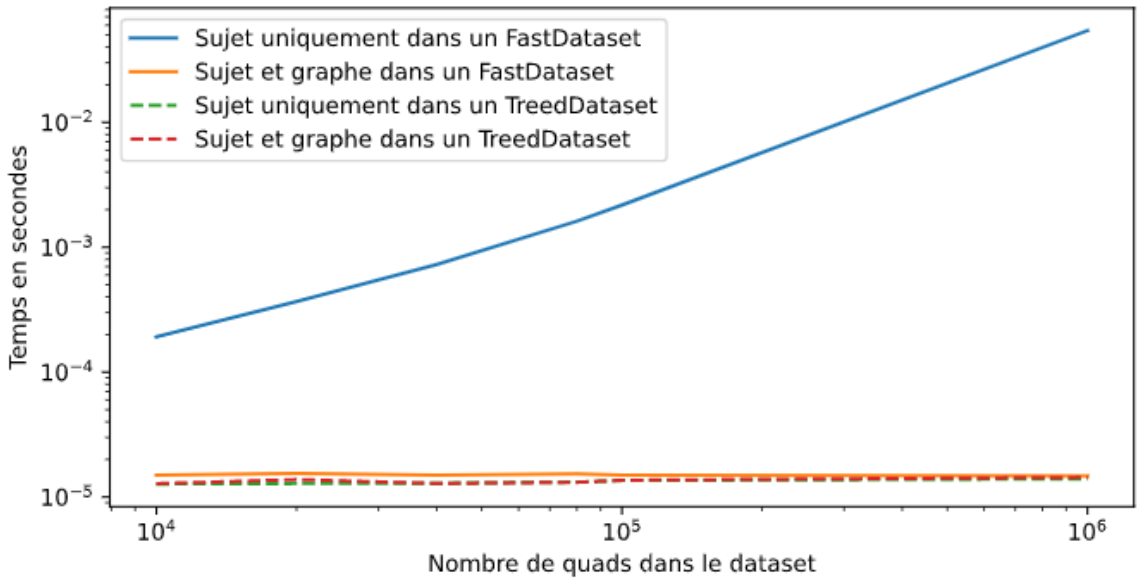


...

Conversion d'un document au format N-Triples / N-Quads en un arbre binaire SPOG (ou plus)

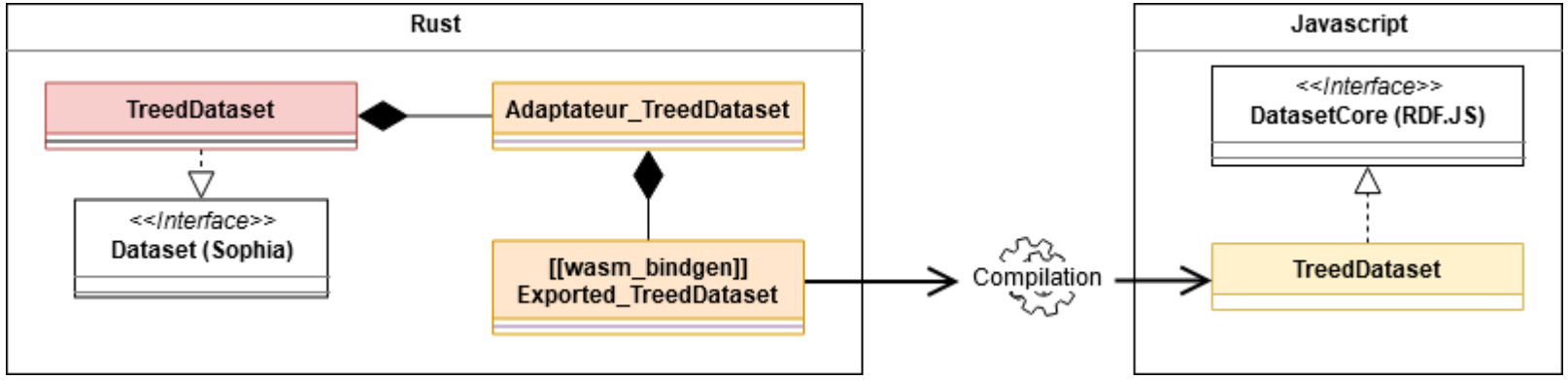
Dataset boisé – Performances en Rust

- Jeu de données : Des extraits de la base « *Person* » de Dbpedia
- Recherche de tous (7) les quads ayant pour sujet Vincent Descombes Sevoie



Temps pour trouver les sept quads correspondant à Vincent Descombes Sevoie en Rust natif avec un FastDataset (dataset par défaut de Sophia) et un TreedDataset (notre implémentation)

Exportation naïve avec un adapteur – Procédé



Procédé pour exporter un TreedDataset de Rust vers Javascript en adhérant à l'API RDF.JS

Exportation naïve avec un adapteur – Résultats

	Temps de chargement (s)	Temps pour trouver toutes les personnes (s)	Mémoire utilisée par le dataset (kB)
Dataset boisé	11,26	0,582	30‘204
Graphy	5,61	0,216	602‘652

Ressources utilisées pour charger 1 million de quads issus du jeu de données Person et pour trouver toutes les personnes (création d’un nouveau Dataset avec seulement les personnes puis itération)

Autres approches – N-Quads

- Récupération des quads du dataset en Javascript (itération)
  - Approche naïve : un échange par quad entre Web Assembly et Javascript
  - Approche N-Quads : on fait un seul échange avec la liste de tous les quads sérialisés puis désérialisés avec le format N-Quads

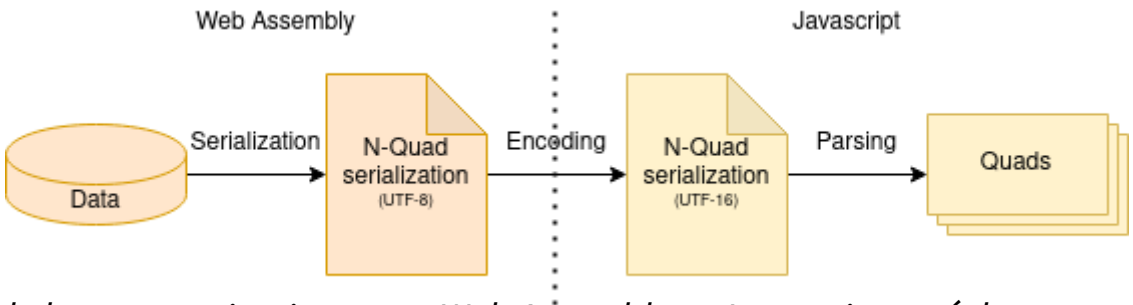
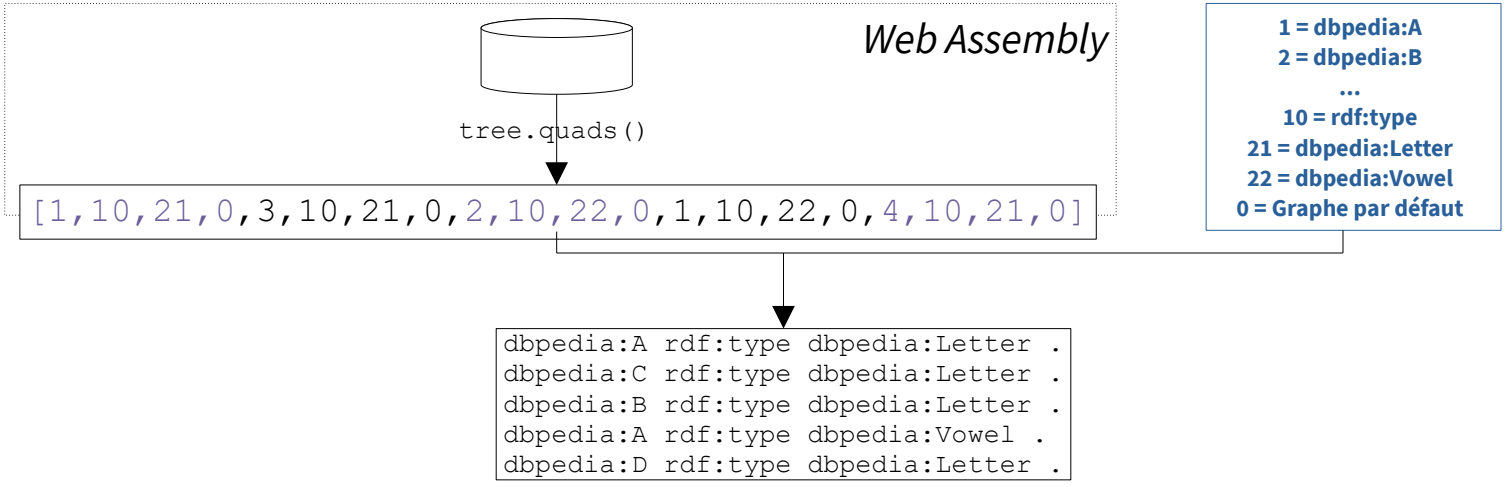


Schéma de la communication entre Web Assembly et Javascript en échangeant la liste des quads au format N-Quads (format textuel)

Autres approches – WasmTree

- Ne plus manipuler de chaînes en Rust / Web Assembly pour éviter les coûts liés à l'encodage
- Un seul échange sous la forme d'un tableau d'index
- Approche n'utilisant plus Sophia pour se mettre au service de Javascript



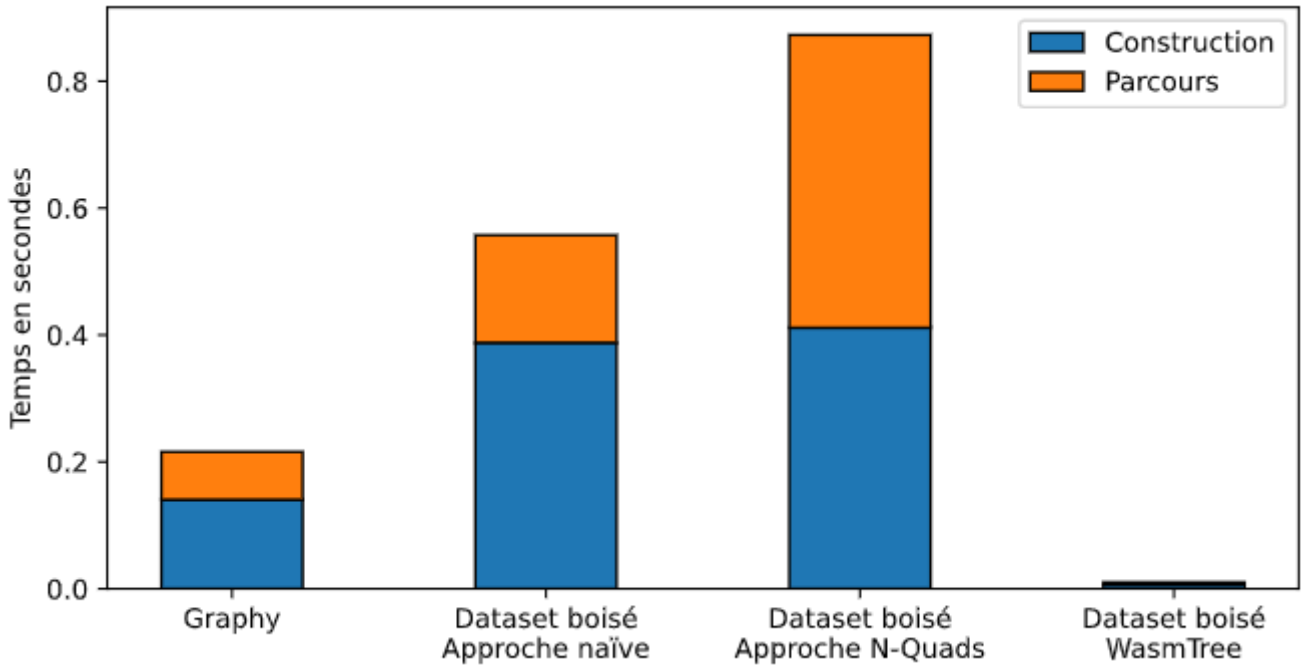
Processus de reconstruction des quads sous forme d'index en Javascript dans WasmTree

Autres approches – Évaluation générale

	Temps de chargement (s)	Temps pour trouver toutes les personnes (s)	Mémoire utilisée par le dataset (kB)
Dataset boisé naïf avec Sophia	11,26	0,582	30‘204
Dataset boisé avec Sophia et échanges N-Quads	11,28	0,87	30‘226
Dataset boisé sans Sophia (WasmTree)	5,41	0,011	429‘244
Graphy	5,61	0,216	602‘652

Ressources utilisées pour charger 1 million de quads issus du jeu de données Person et pour trouver toutes les personnes

Autres approches – Évaluation sur la recherche



Répartition de la charge entre parcours et itération dans le temps de recherche de toutes les personnes d'un extrait de 1'000'000 de quads du dataset Persondata

Evaluation avec SPARQL – Contexte

- SPARQL (SPARQL Protocol and RDF Query Language)
  - Langage de requête sur des graphes et des datasets RDF
- Comunica : Infrastructure logicielle qui peut construire un SPARQL end point à partir d'un Store RDF.JS
- BSBM (Berlin SPARQL Benchmark)
  - *Benchmark* évaluant les performances d'un *SPARQL end point* utilisant un jeu de données sur des produits fictifs
- Oxigraph : Moteur SPARQL écrit en Rust

Evaluation avec SPARQL – Résultats

Moteur SPARQL	Mix de requêtes par heure	Accélération par rapport à Comunica natif
Comunica (natif utilisant N3.js)	65,64	
Comunica (utilisant WasmTree)	446,22	x6,8
Oxigraph	8877,66	x135,25

Mesure des performances avec BSBM (2000 produits) sur différents SPARQL end point en Javascript

Conclusion

- Bonnes performances avec une répartition judicieuse des tâches pour augmener le rapport entre les gains de performances et les coûts des échanges et conversions entre Web Assembly et Javascript
- Table de correspondance en Javascript : perte de la sémantique des termes et donc de la possibilité de faire de l'inférence en Rust
- Jeunesse des technologies utilisées : limitation de la mémoire à 2Go, fuites de mémoires, pas de possibilité d'exporter directement les méthodes implémentant un trait, encodage … (non abordés dans cette présentation)

Références

- [BSBM] Bizer, C., & Schultz, A. (2009). The berlin sparql benchmark. International Journal on Semantic Web and Information Systems (IJSWIS), 5(2), 1-24.
- [HDT] Fernández, J. D., Martínez-Prieto, M. A., Gutiérrez, C., Polleres, A., & Arias, M. (2013). Binary RDF representation for publication and exchange (HDT). Journal of Web Semantics, 19, 22-41.
- [RDF-4X] Abbassi, S., & Faiz, R. (2016, November). RDF-4X: a scalable solution for RDF quads store in the cloud. In Proceedings of the 8th International Conference on Management of Digital EcoSystems (pp. 231-236).
- [Sophia] Champin, P. A. (2020, April). Sophia: a Linked Data and Semantic Web toolkit for Rust. In The Web Conference 2020: Developers Track.

Merci de votre attention.
Avez-vous des questions ?

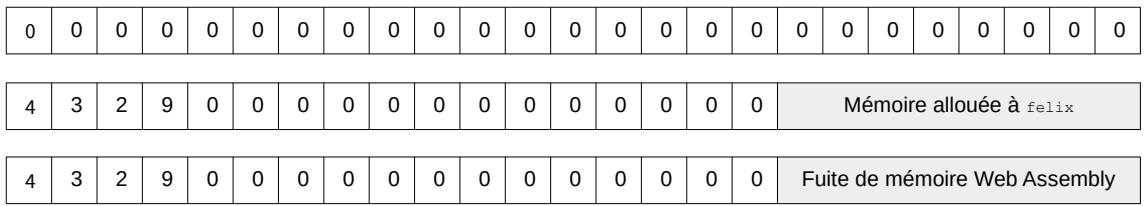
Annexe – Problèmes de mémoire

- Problème de la libération de la mémoire Web Assembly

Code Javascript

```
// Code Javascript
{
  let felix = new wasm_bindgen.Cat();
}
// Suite du code Javacript
```

Mémoire linéaire



Exemple de programme provoquant une fuite de mémoire Web Assembly

  - Pas de garbage collector pour récupérer la mémoire
  - Besoin de libérer explicitement la mémoire (appel à la méthode `.free()`)
  - Depuis Juillet 2020 : les *WeakRefs* permettent de simuler un destructeur
- Problème d'adressage en Web Assembly : limite de 32 bits (2 à 4Go)