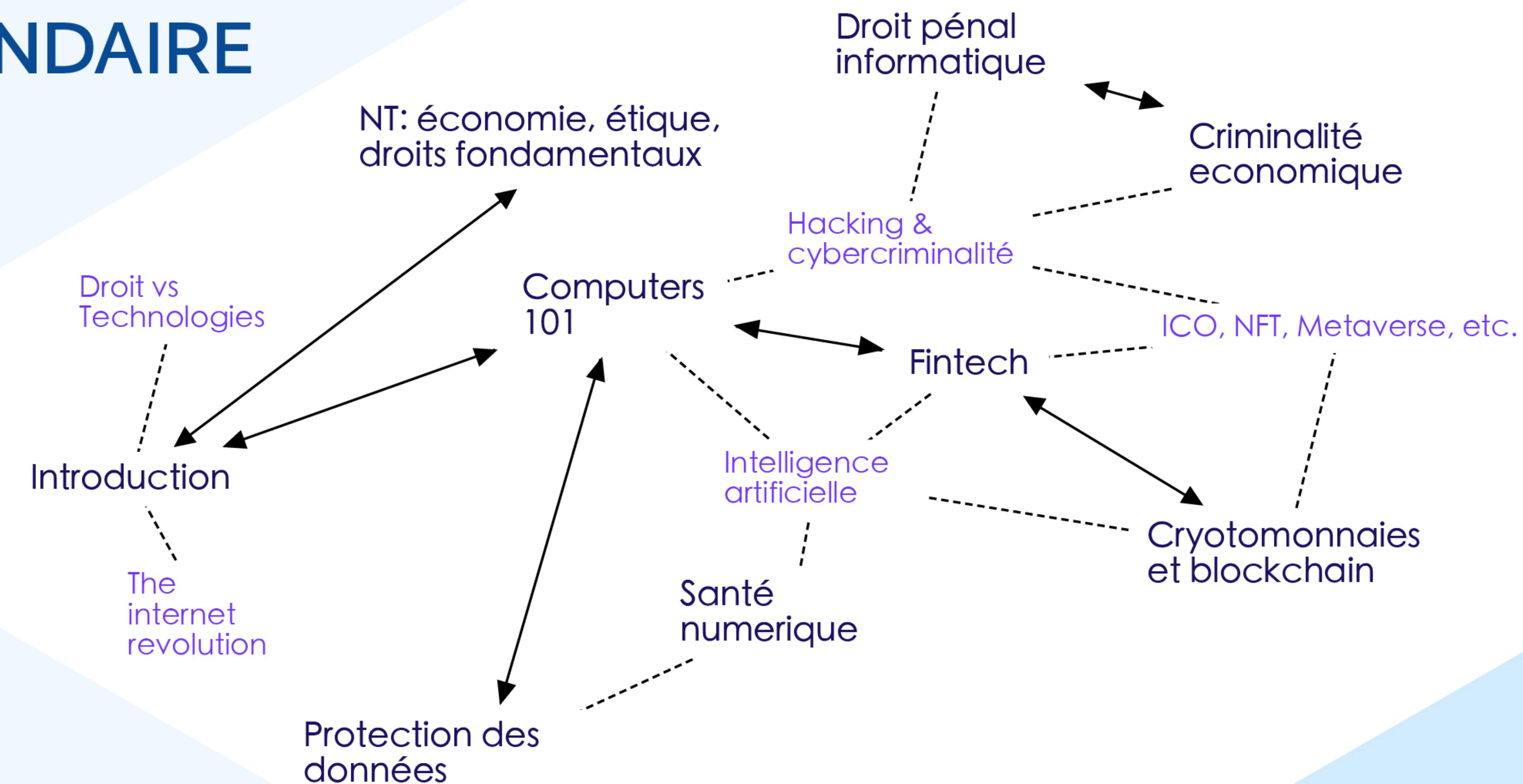


Semestre d'automne 2025-26 / Jeudi / 13h30-16h45

DROIT ET NOUVELLES TECHNOLOGIES

OPTION SECONDAIRE

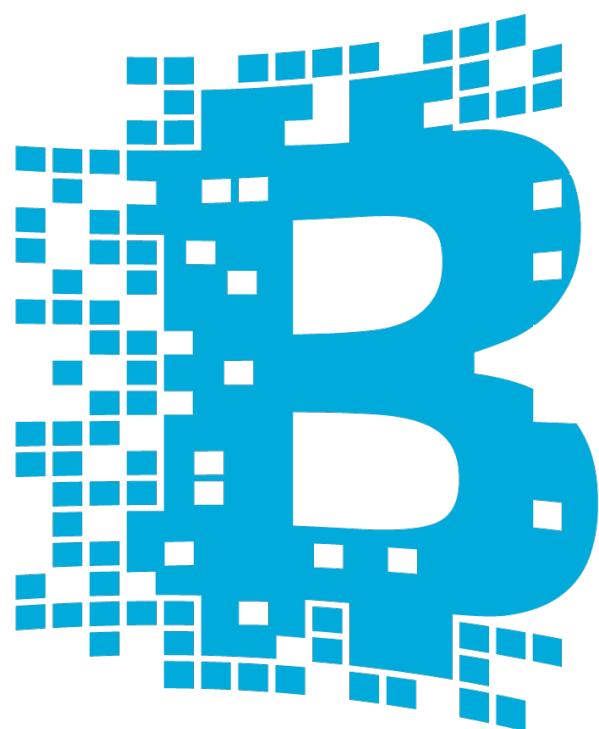
GET STARTED





“Le droit, confronté à ces changements rapides et radicaux, se doit de s’adapter et de se réinventer. Les juristes de demain devront être familiarisés avec ces problématiques et ceux qui pourront faire valoir des compétences spécifiques seront avantagés sur le marché du travail.”

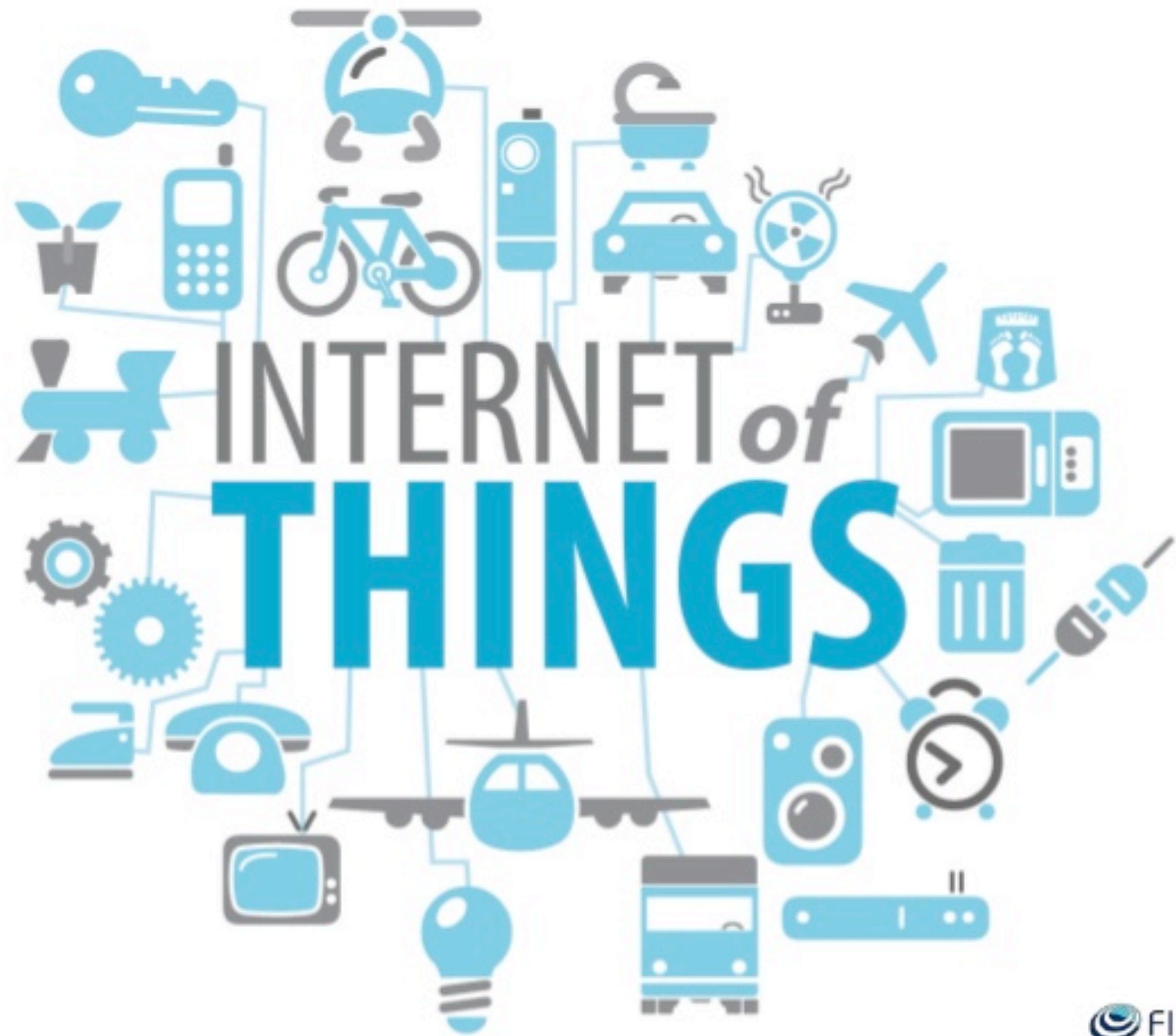
“Une révolution comparable au lancement d'internet” - William Mougayar



BLOCKCHAIN

The word "bitcoin" in a lowercase, dark gray sans-serif font. To the left of the text is the white Bitcoin logo icon, which consists of a circular arrangement of eight segments forming a stylized 'B' or hash symbol, set against an orange circular background.

bitcoin



“The Internet of Things is not just a technology, but will change the models of business” - iotlaw.net



Est-on prêt pour l'intelligence artificielle?



eHealth et mobile health: quelle protection pour nos données sensibles?

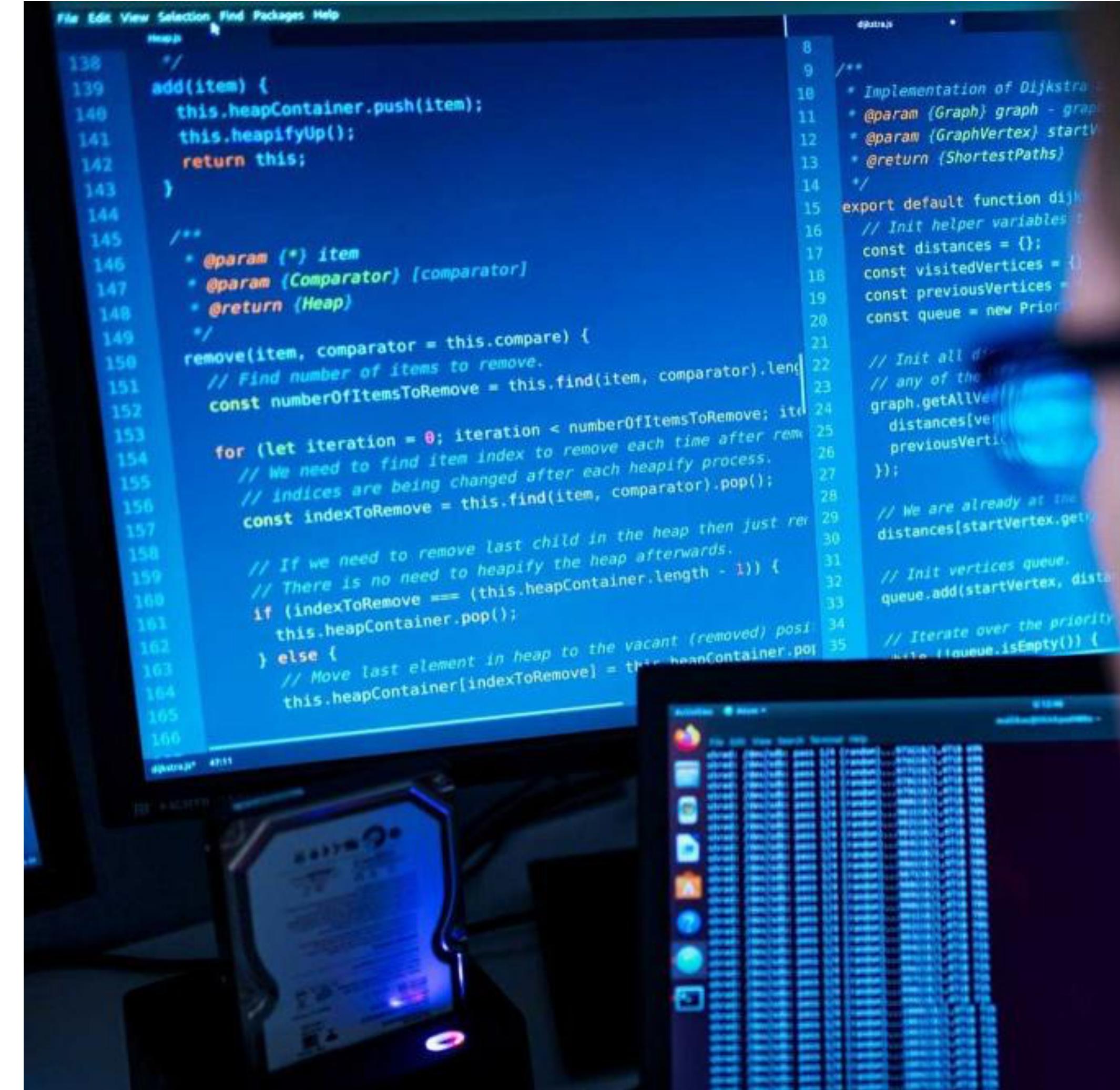
“Fintech: meet the revolutionary new area of a traditional industry” - business.com



C Y B E R
C R I M E

*De nouvelles opportunités, mais aussi
pour les criminels...*

... et des nouveaux défis pour les autorités

A photograph showing a person from the side, wearing a virtual reality headset. They are looking at a computer monitor which displays two code editor windows. The left window shows code for a 'Heap' class with methods like 'add(item)' and 'remove(item, comparator)'. The right window shows code for a 'dijkstra' function, which implements Dijkstra's algorithm for finding shortest paths in a graph. Both windows have line numbers on the left and syntax highlighting for different programming constructs.

```
File Edit View Selection Find Packages Help
Heap
138  */
139  add(item) {
140   this.heapContainer.push(item);
141   this.heapifyUp();
142   return this;
143 }
144 /**
145  * @param {*} item
146  * @param {Comparator} [comparator]
147  * @return {Heap}
148 */
149 remove(item, comparator = this.compare) {
150   // Find number of items to remove.
151   const numberOfItemsToRemove = this.find(item, comparator).length;
152
153   for (let iteration = 0; iteration < numberOfItemsToRemove; iteration++) {
154     // We need to find item index to remove each time after removal.
155     // indices are being changed after each heapify process.
156     const indexToRemove = this.find(item, comparator).pop();
157
158     // If we need to remove last child in the heap then just remove it.
159     // There is no need to heapify the heap afterwards.
160     if (indexToRemove === (this.heapContainer.length - 1)) {
161       this.heapContainer.pop();
162     } else {
163       // Move last element in heap to the vacant (removed) position.
164       this.heapContainer[indexToRemove] = this.heapContainer.pop();
165     }
166   }
167 }

dijkstra
8
9 /**
10  * Implementation of Dijkstra's algorithm.
11  * @param {Graph} graph - graph to search.
12  * @param {GraphVertex} startVertex - starting vertex.
13  * @return {ShortestPaths}
14 */
15 export default function dijkstra(graph, startVertex) {
16   // Init helper variables.
17   const distances = {};
18   const visitedVertices = {};
19   const previousVertices = {};
20   const queue = new PriorityQueue();
21
22   // Init all distances to infinity.
23   // Any of the vertices can be reached from the start vertex.
24   graph.getAllVertices().forEach(vertex => {
25     distances[vertex] = Infinity;
26     previousVertices[vertex] = null;
27   });
28
29   // We are already at the start vertex.
30   distances[startVertex] = 0;
31
32   // Init vertices queue.
33   queue.add(startVertex, distances[startVertex]);
34
35   // Iterate over the priority queue until it is empty.
36   while (!queue.isEmpty()) {
37     const currentVertex = queue.remove();
38
39     graph.getNeighbors(currentVertex).forEach(neighb
```

et bien plus encore!



Inscrivez-vous maintenant!