XEBIA NATIS

Les bases en JavaScript

Michael
XEBIA NATIS

XNATIS

Compétence visée :

Créer des classes avec leurs constructeurs et leurs propriétés

XEBIA NATIS

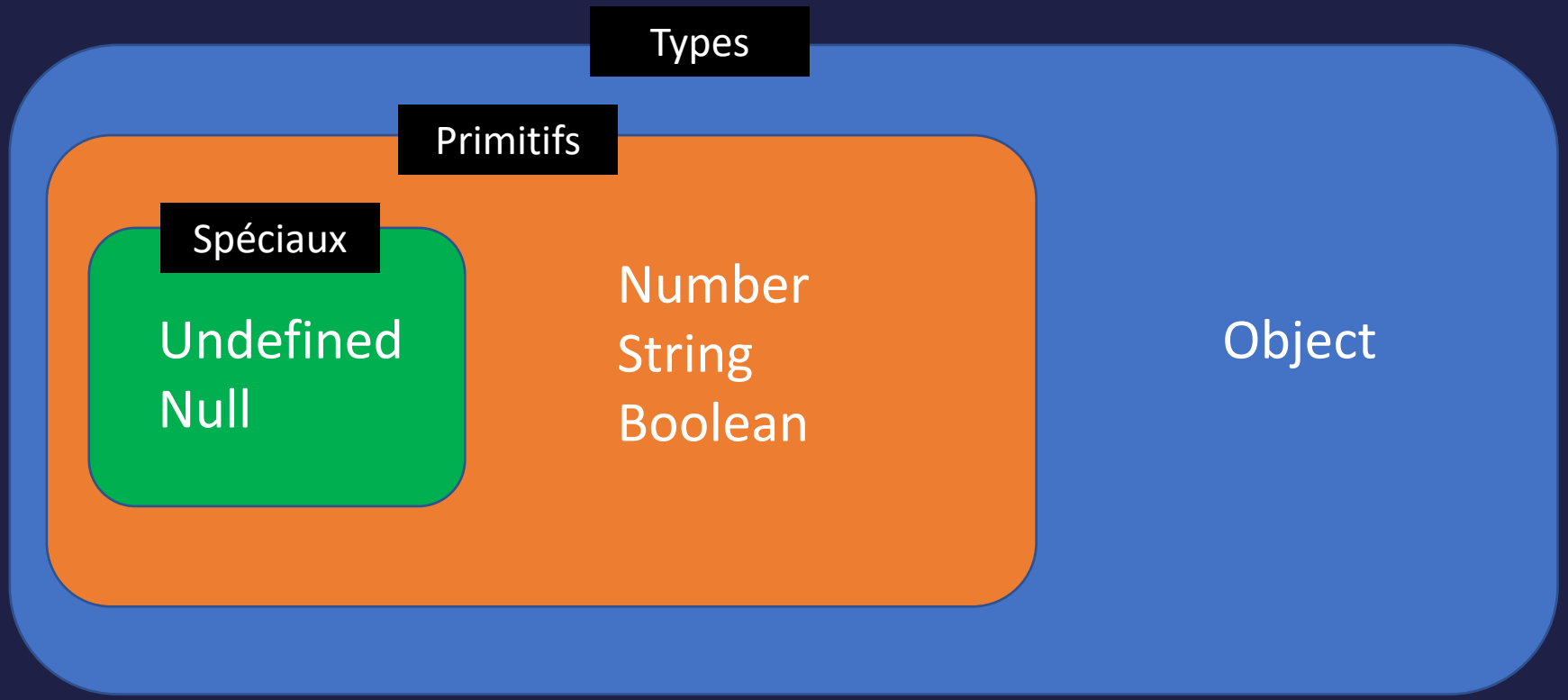
1. Les types
2. La portée statique
3. Le polymorphisme par shadowing
4. Le contexte objet this
5. Les designs pattern en JS
6. L'Instanciation
7. Le DOM

XWIKI SAS

XWIKI NATIS

Les types

XEBNATIS


Types
Primitifs
Spéciaux
Undefined
Null
Number
String
Boolean
Object

XEBIA NATIS

L'opérateur typeof renvoie une chaîne qui indique le type de son opérande.

XWIKI NATIS

Un objet est un pointeur (référence) alors qu'un contenu d'un type primitif est une valeur

XEBIA NATIS

```
const a = 'hello';
let b = a;
b = 'ca va';
console.log(a === b); // prints false
```

XPNATIS

```
const obj = {
    id: 4
};
let copie = obj;
copie.id = 3;
console.log(obj === copie); // prints true
```

XWIKI SAS

XDNATIS

La portée statique

XWIKI NATIS

La portée (scope) d'une variable x est la région du programme dans laquelle les utilisations de x font référence à sa déclaration.

L'une des raisons fondamentales du scoping est de garder les variables dans différentes parties du programme distinctes les unes des autres.

XWIKI NATIS

La portée (ou scope) concerne la visibilité des variables. Elle détermine la portion de code a partir du moment où la variable naît jusqu’au moment où elle meurt.

XWIKI NATIS

Puisqu'il n'y a qu'un petit nombre de noms de variables courts et que les programmeurs partagent des habitudes concernant la dénomination des variables (par exemple, i pour un index de tableau), dans tout programme de taille modérée, les conflits seront évités !

XWIKI NATIS

Avec une portée dynamique (dynamic scoping), un identifiant global fait référence à l'identifiant associé à l'environnement le plus récent, et est rare dans les langues modernes.

En termes plus simples, dans la portée dynamique, l’interpréteur recherche d'abord le bloc courant puis successivement toutes les fonctions appelantes.

XP NATIS

```
function hello() {
    var x = 1;
}

function cava() {
    console.log(x);
}

hello();
cava();
```

?

XPNATIS

```
function hello() {
    var x = 1;
}

function cava() {
    console.log(x);
}

hello();
cava();
```

XWIKI NATIS

La portée statique est également appelée portée lexicale. Dans cette portée, une variable fait toujours référence à son environnement de niveau supérieur.

XWIKI NATIS

Il s'agit d'une propriété du texte du programme et sans rapport avec la pile d'appels d'exécution. La portée statique facilite également la création d'un code modulaire, car un programmeur peut déterminer la portée simplement en regardant le code.

XWIKI NATIS

La portée dynamique oblige le programmeur à anticiper tous les contextes dynamiques possibles !

XEBIA NATIS

Les variables ont 3 portées statiques :

- Global scope
- Function scope (ou aussi local scope)
- Block scope (pour let et const)

XDNATIS

Nous pouvons ne pas travailler dans le contexte global en mettant notre code dans une fonction et l'exécuter directement : c'est une IIFE (Immediately Invoked Function Expression)

XEBIA NATIS

Pour retrouver une variable, l'interpréteur de JS va regarder dans la portée statique courante, et s'il ne la trouve pas, il regardera dans la portée statique externe … puis ainsi de suite jusqu'au Global scope.

C'est ce qu'on appelle le Scope chain !

XWIKI NATIS

Une closure (fermeture) est la paire formée d'une fonction et des références à son état environnant (l'environnement lexical).

En d'autres termes, une closure donne accès à la portée d'une fonction externe à partir d'une fonction interne (on dit aussi que la fonction « capture son environnement »)

XEBIA NATIS

En JavaScript, une closure (fermeture) est créée chaque fois qu'une fonction est créée.

XWIKI SAS

XWIKI NATIS

Le polymorphisme par shadowing

XEBIA LABS

```
function factory()
{
    let x = 'hello';
    return function sayHello() {
        console.log(x);
    }
}

let x = 'bye';
const fun = factory();
fun();
```

XEBIA LABS

```
function factory()
{
    return function sayHello() {
        console.log(x);
    }
}

let x = 'bye';
const fun = factory();
fun();
```

XNATIS

Lorsqu'une variable est déclarée dans une certaine portée ayant le même nom défini sur sa portée externe et lorsque nous appelons la variable depuis la portée interne, la valeur attribuée à la variable dans la portée interne est la valeur qui sera stockée dans la variable dans l'espace mémoire.

XWIKI NATIS

Ceci est connu sous le nom de Shadowing ou Variable Shadowing.

XEBIA LABS

```
let number = 10;

{
    function double()
    {
        return number * 2;
    }
}
{
    function double()
    {
        return number * 3;
    }
}

console.log(double());
```

XEBIA NATIS

```
let number = 10;
function double()
{
    let number = 20;
    number = number * 2;
}
double();
console.log(number);
```

XWIKI SAS

XWIKI NATIS

Le contexte objet ‘this’

XWIKI NATIS

La portée (ou scope) concerne la visibilité des variables alors que le contexte fait référence à l'objet auquel appartient une variable ou une fonction.

XEBIANATIS

GEC / Global Execution Context est également appelé exécution de base / par défaut.

Tout code JavaScript qui ne réside dans aucune fonction sera présent dans le contexte d'exécution global.

XWIKI NATIS

La raison derrière son nom « contexte d'exécution par défaut » où le code commence son exécution lorsque le fichier se charge pour la première fois.

XEBIA NATIS

Le GEC découle du flow d'exécution suivant :

1. l'interpréteur crée un objet global pour Node.js et un objet Window pour les navigateurs
2. Il référence l'objet ci-dessus au mot-clé 'this'
3. Il crée une memory heap afin de stocker des variables et des références de fonction

XWIKI NATIS

La moyen standard d’obtenir l’objet global est globalThis !

XWIKI NATIS

On peut créer un nouveau contexte (et donc un objet) en utilisant l'opérateur new.

Cet object remplacera le contexte courant (this).

XWIKI NATIS

Le contexte courant est en effet dénoté par le mot clé ‘this’.

Il référence majoritairement l’objet courant sur lequel les opérations s’appliquent.

XEBNATIS

La méthode .apply() appelle une fonction en lui passant une valeur this et des arguments sous forme d'un tableau (ou d'un objet semblable à un tableau).

XEBIANATIS

```
function Person(name) {
    this.name = name;
}

const obj = {
    age: 14
};
Person.apply(obj, ['Sophie']);
console.log(obj);
console.log(obj instanceof Person);
```

XDNATIS

La méthode .call() réalise un appel à une fonction avec une valeur this donnée et des arguments fournis individuellement.

XEBIA LABS

```
function Person(name) {
    this.name = name;
}

const obj = {
    age: 14
};
Person.call(obj, 'Sophie');
console.log(obj);
console.log(obj instanceof Person);
```

XEBIA NATIS

Les objets sont constitués de membres (propriétés ou méthodes) qui permettent de les décrire.

Construire un objet, c'est le faire physiquement exister en mémoire pour contenir des données.

XWIKI NATIS

Les valeurs des membres d'un objet peuvent être construites à partir de types de données primitifs ou à partir d'autres objets.

Comme des poupées russes ! ☺

XEBNATIS

```
const obj = {
    id: 3,
    firstname: 'Annie',
    lastname: 'Versaire',
    email: 'annie.versaire@gmail.com'
};
```

XEBNATIS

```
const obj = {
    id: 3,
    firstname: 'Annie',
    lastname: 'Versaire',
    email: 'annie.versaire@gmail.com',
    children: [
        {
            id: 4,
            age: 12,
            firstname: 'Jean',
            lastname: 'Aimarre'
        },
        {
            id: 6,
            age: 24,
            firstname: 'Guy',
            lastname: 'Tarre'
        },
    ]
};
```

XDNATIS

Les designs pattern en JS

XEBIA NATIS

De part la nature de JS, il y a des designs pattern propres à ce langage ou dont l'implémentation est propre à ce langage

XEBIA NATIS

Designs pattern (patrons de conception) :

- Factory
- Singleton
- Module Pattern
- Revealing Module Pattern

XWNATIS

La fonction d'usine (factory) est similaire aux fonctions de constructeur, mais au lieu d'utiliser new pour créer un objet, les fonctions factory créent simplement un objet et le renvoie.

XEBIA LABS

```
function factory()
{
    this.x = 'hello';
    return function sayHello() {
        console.log(this.x);
    }
}
this.x = 'bye';
const fun = new factory();
fun();
```

XWIKI SAS

XWIKI NATIS

Instanciation

XEBNATIS

Consultons les membres de nos objets ! ☺

Avec Object.getOwnPropertyNames

XWIKI NATIS

Un constructeur (ou initialisateur) est une fonction qui vous permet de fournir toute initialisation personnalisée qui doit être effectuée avant qu'une autre méthode puisse être appelée sur un objet instancié.

XDNATIS

Lorsqu’une fonction constructeur crée un objet, elle le « marque » :

- Par la propriété : constructor
- Par la propriété : __proto__

XEBIA NATIS

La plupart des constructeurs intégrés, tels que Object, Regex et Array, sont scope-safe.

Si new n'est pas utilisé, ils renvoient une instance appropriée de l'objet en appelant à nouveau le constructeur avec new.

XEBIANATIS

```
function Voiture(price) {
    if (!(this instanceof Voiture)) {
        return new Voiture(price);
    }
    this.price = price;
}

const obj = Voiture(234);
console.log(obj instanceof Voiture); // prints true
```

XWIKI NATIS

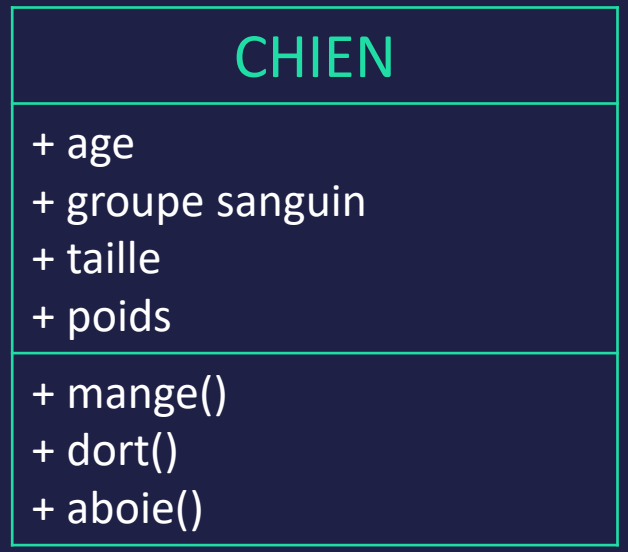
Quel est le type de class ?

XDNATIS

Les classes JavaScript ont été introduites avec ECMAScript 2015. Elles sont un « sucre syntaxique » par rapport à l'héritage prototypique.

En effet, cette syntaxe n'introduit pas de nouveau modèle d'héritage en JavaScript !

XEBIA NATIS



XWIKI SAS

XWIKI NATIS

Le Document Object Model

XDNATIS

Qu‘est-ce que le DOM ?

XDNATIS

Navigateur

Requete HTTP (avec optionnelement un payload)

Réponse HTTP avec de l‘ HTML

Serveur Web (serveur PHP)

XDNATIS

Navigateur

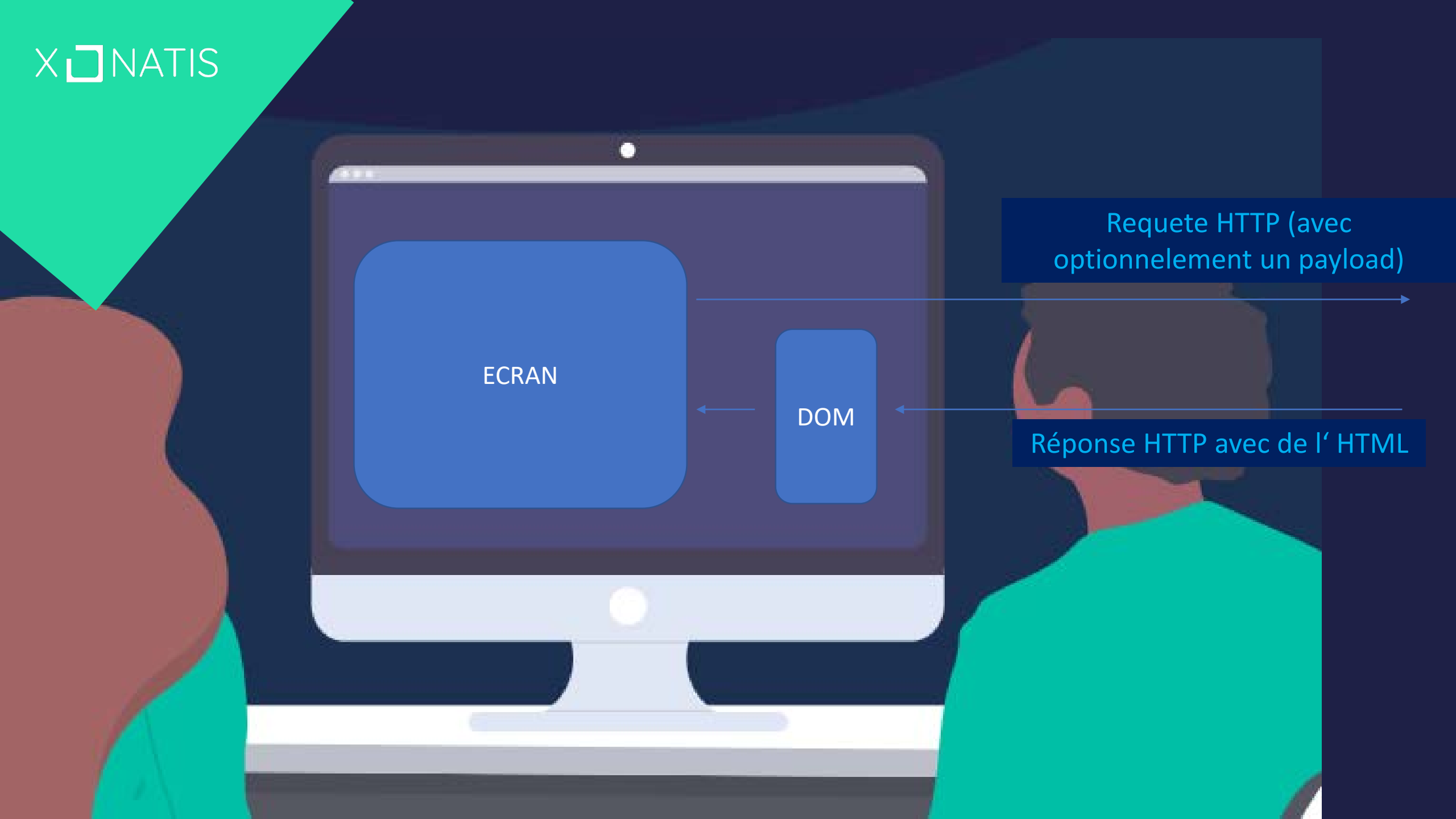
Requete HTTP (avec optionnelement un payload)

Réponse HTTP avec de l‘ HTML

Serveur Web (serveur PHP)

ZOOMONS DEDANS !

XDNATIS


ECRAN
DOM
Requete HTTP (avec optionnelement un payload)
Réponse HTTP avec de l‘ HTML

XEBIA NATIS

Le DOM est l'interprétation de l'HTML par le navigateur et c'est le DOM qui est affiché à l'écran.

XEBIA NATIS

Cela peut être donc faux de dire qu'à l'écran, on voit le code HTML, car plus précisément, on voit le DOM sur l'écran (bien que le DOM vient de l'HTML)

XWIKI NATIS

Pour aller plus loin

https://developer.mozilla.org/fr/docs/Learn/JavaScript/Objects/Basics