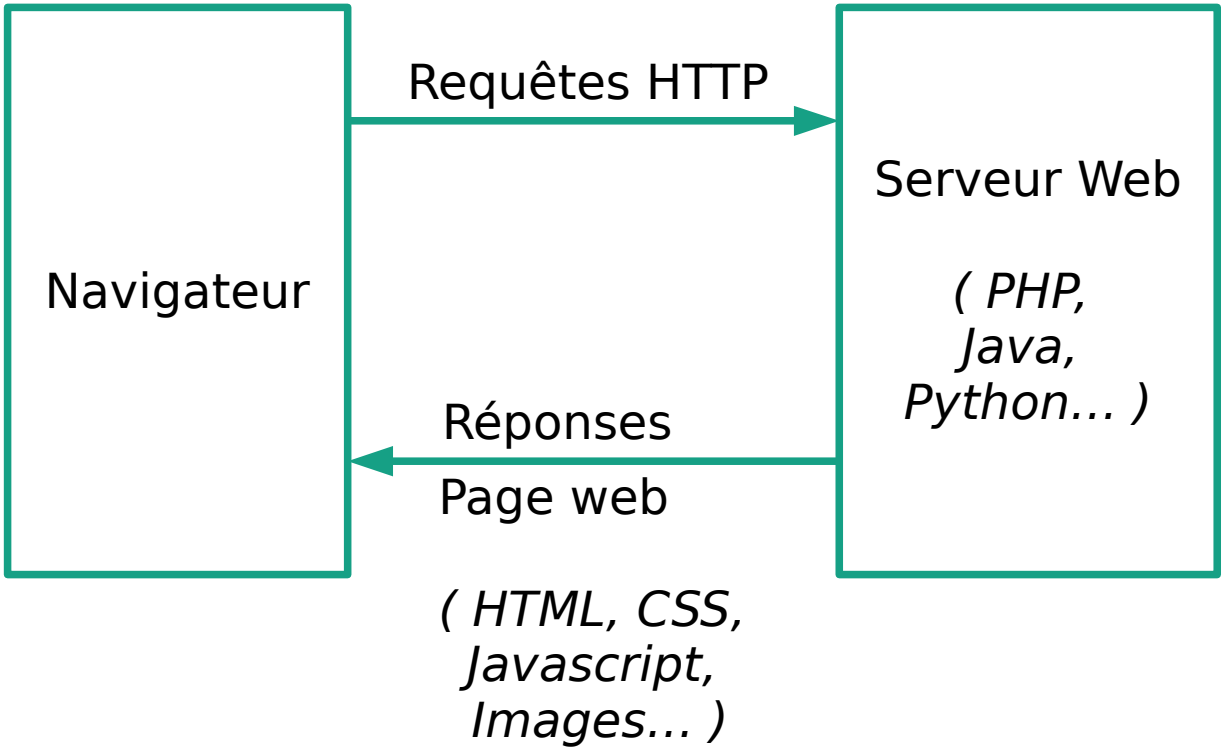
Javascript

Javascript
Nathanaël Martel

Rappel : architecture n-tiers



Rappel : HTML / CSS / javascript

- **Ce sont des langage normé**
- **Ils sont interprété (exécuté) par le navigateur**
- **Les normes évoluent et tous les navigateurs ne respectent pas complètement les normes**
  - Un outil comme caniuse.com vous permet de vérifier le support de tel ou tel fonctionnalité

Rappel : HTML / CSS / Javascript

- **HTML :**
  - Le fond
  - Le sens, et donc l'utilisation de balise qui ont du sens
- **CSS :**
  - La forme
  - L'apparence
  - Le comportement
- **Javascript**
  - Le comportement
  - Les interactions
  - La programmation côté navigateur

Client / Serveur

- **Javascript est le seul langage côté client, le seul qui soit disponible dans les navigateurs :**
  - Nous n'avons pas le choix !
- **Il est possible d'utiliser ce langage également côté serveur (pour générer du HTML et s'interfacer avec une base de données)**
  - Node.js, react.js, view.js…

HTML / DOM

- **HTML**
  - Code source qui est envoyé par le serveur
  - Clic droit / afficher la source
- **DOM**
  - Structure généré par le navigateur en fonction du code source
  - Le navigateur y attache le CSS
  - Clic droit / examiner (inspecter)
  - Peut être modifié en javascript

Debugage

Examiner l'élément (Inspecter) / Onglet « console ».

Cela permet :

- **De voir les erreurs d'exécution**
- **De tester du code en direct**
- **De tester du code avec les variables déclarer dans la page**
- **Javascript permet d'y afficher du code :**
  - `console.log("information de debugage");`

Lien HTML / Javascript

- **Indiquer le fichier de script**
  - `<script src="/assets/js/validator.min.js"></script>`
  - Il est possible d'en mettre plusieurs
- **Utiliser la balise `<script>`**
  - À éviter un maximum
- ~~**Utiliser les attributs…**~~
  - **Jamais**, cela pose de problème de découplage et peut bloquer l'exécution du javascript

Syntaxe

- **Proche de c/c++**
- **Chaque ligne finit par un ;**
- **Sensible à la casse**

Variable

- **Déclaration de variable**
  - `var i=0;`
  - ```
    var i, j;
    i = 2;
    j = 3;
    ```
- **Les variables ont une portée locale**
- **La lecture d'une variable non déclaré déclenche une erreur**
- **Tant qu'aucune valeur n'est affecté, la variable est « `undefined` »**
    ```

Type de variable

- Il n’y a pas de déclaration de type
- Principaux type :
  - String
  - Number
  - Boolean
  - Null
  - Undefined
- La fonction `typeof()` permet de connaître le type d’une variable

Syntaxe

- **Opérations**
  - `+ - * / %`
- **concaténation**
  - `"Hello " + "World"`

Syntaxe

- **Opérateurs de comparaison**
  - `==` `!=`
  - `===` `!==` **(en valeur et en type)**
  - `<` `>` `<=` `>=`
- **Opérateurs logique**
  - `&&`
  - `||`
  - `!`

Syntaxe

- **Structure if**
  - ```
    if ((test <= 19) && (test2 == 15)) {
       console.log("Vrai !");
    } else {
       console.log("Faux !");
    };
    ```
    ```

Syntaxe

- **Structure switch**

```
switch (test) {
  case 1:
    console.log("1");
    break;
  case 12:
    console.log("1");
    break;
  default:
    alert("Autre chose");
};
```

Syntaxe

- **Boucle while**

```
while (x < 10) {
  console.log("x = " + x);
  x++;
};
```

Syntaxe

- **Boucle for**

```
for (x = 0 ; x < 10 ; x++) {
  console.log("x = " + x);
};
```

Syntaxe

- **Fonction**

```
function multiplication(x, y) {
    alert(x * y);
    return x*y;
};
var resultat = multiplication(2, 3);
```

Objets

- Les objets de JavaScript, sont prédéfinis dans le langage, ou créés par le programmeur.
  - le navigateur est un objet qui s'appelle "navigator".
  - La fenêtre du navigateur se nomme "window"
  - La page HTML est un autre objet, que l'on appelle "document".
  - Un formulaire à l'intérieur d'un "document", est aussi un objet.
  - Un lien hypertexte dans une page HTML, est encore un autre objet. Il s'appelle "link"

Objets

```
var personne = {
    prenom: "Jean",
    nom: "Dupont",
    age: 20,
    identite: function() {
        return "Prénom: " + this.prenom + " , nom: " + this.nom;
    };
};
// Accès aux propriétés
console.log(personne.prenom);
// Utilisation des méthodes
console.log(personne.identite);
```

Objet string

- **Propriété :**
  - **length : retourne la longueur de la chaîne de caractères;**
- **Méthodes :**
  - **indexOf( ) : permet de trouver l'indice d'occurrence d'un caractère dans une chaîne;**
  - **lastIndexOf( ) : permet de trouver le dernier indice d'occurrence d'un caractère;**
  - **slice( ) : retourne une portion de la chaîne;**
  - **substr( ) : retourne une portion de la chaîne;**
  - **substring( ) : retourne une portion de la chaîne;**
  - **toLowerCase( ) : permet de passer toute la chaîne en minuscule;**
  - **toUpperCase( ) : permet de passer toute la chaîne en majuscules;**

Objet array

- **Propriété :**
  - **length : retourne le nombre d'éléments du tableau;**
- **Méthodes :**
  - **concat( ) : permet de concaténer 2 tableaux;**
  - **join( ) : converti un tableau en chaîne de caractères;**
  - **reverse( ) : inverse le classement des éléménts du tableau;**
  - **slice( ) : retourne une section du tableau;**
  - **sort( ) : permet le classement des éléments du tableau;**

Manipulation du DOM

- Il faut utiliser l'objet `document`
- Méthodes :
  - `getElementById()`
  - `getElementsByTagName()`
  - `getElementsByClassName()`
  - `querySelector()`
  - `QuerySelectorAll()`
- Exemples :

```
document.getElementById("titre")
document.getElementsByTagName("h2")
document.querySelector("p a")
document.querySelectorAll("h2").length
```

Manipulation du DOM

- **Contenu d'un élément :**
- **Propriétés :**
  - `InnerHTML` → **Le contenu HTML**
  - `TextContent` → **Uniquement le texte**
- **Ces propriétés peuvent être modifié**
- **Exemples :**
  - ```
    document.querySelector("p a").innerHTML;
    document.querySelector("p").innerHTML;
    document.querySelector("p").textContent;
    ```
  - ```
    document.querySelector("p a").textContent = "nouveau lien";
    ```

Manipulation du DOM

- **Attributs d'un élément :**
- **Ce sont des propriétés qui peuvent être modifié :**
  - `href` → **pour l'URL d'un lien**
  - `src` → **pour la source d'une image**
  - **…**
- **Exemples :**
  - ```
    document.querySelector("p a").href;
    document.querySelector("p a").href =
    "https://google.com";
    ```
    ```

Manipulation du DOM

- **Style d'un élément :**
  - `style.color` → pour l'URL d'un lien
  - `style.fontSize` → pour la source d'une image
  - …
- **Exemples :**
  - ```
    document.querySelector("p a").style.color;
    document.querySelector("p a").style.color =
    "red";
    ```
    ```

Manipulation du DOM

- **Classes d'un élément :**
  - `element.classList` → **tableau contenant les noms des classes**
  - `element.classList.add("omega")` → **ajoute une classe**
  - `element.classList.remove("omega")`
  - `element.classList.toggle("omega")`
  - `element.classList.contains("omega")`
  - `element.classList.replace("old-class", "new-class")`
  - `…`
- **Exemples :**
  - ```
    document.querySelector("h1").classList
    document.querySelector("h1").add("omega")
    ```
    ```

Manipulation du DOM

Création d'un élément

Créer un nouvel élément

```
var element = document.createElement("p");
```

Valoriser des attributs

```
element.id = "nouveau";
```

Créer du contenu

```
var texte = document.createTextNode("Texte du paragraphe");
```

Ajouter le texte au paragraphe

```
element.appendChild(texte);
```

Manipulation du DOM

Ajouter l'élément dans le document

À la fin

```
document.body.appendChild(element);
```

Avant un autre élément

```
document.querySelector("h2").insertBefore(element)
```

Après un autre élément

```
document.querySelector("h2").insertAfter(element)
```

Manipulation du DOM

Propriétés d'un élément

```
document.querySelector("h2").parentNode
document.querySelector("body").childNodes
document.querySelector("body").children
document.querySelector("body").firstChild
document.querySelector("body").lastChild
document.querySelector("h2").nextSibling
document.querySelector("h2").previousSibling
```

Événements

- Il est possible d'ajouter des évènement à un nœud. C'est le « gestionnaire d'évènement » (event handler)
- Le code est exécuté dès le déclenchement de l'évènement.
- La méthode `element.addEventListener(event, function)` permet de lier à un nœud, une action.

Événements

```
function fonctionClick() {
    // this est un objet qui désigne
l'élément qui a généré l'événement
    console.debug(this);
    // On peut changer son texte
    this.textContent = "Nouveau titre...";
}
var h1 = document.querySelector("h1");
h1.addEventListener("click", fonctionClick);
```

Événements

```
var h1 = document.querySelector("h1");
h1.addEventListener("click", function() {
    // this est un objet qui désigne l'élément qui a généré l'événement
    console.debug(this);
    // On peut changer son texte
    this.textContent = "Nouveau titre...";
});
```

Événements

- **Quelques types d'événements**
  - `blur` → Un élément perd le focus
  - `change` → Un élément perd le focus et sa valeur a changé
  - `keyup` → Une touche est relâchée
  - `click` → Un bouton d'un dispositif de pointage a été appuyé ou relaché
  - `DOMContentLoaded` → Le document a fini de charger
  - `load` → Le document et ses assets ont fini de charger
  - `mouseover`/`mouseout` → la sourie rentre/sort sur la zone de l'élément
  - `resize` → le viewport a changé de taille (redimensionnement fenêtre)
  - `scroll` → la vue a été scrollé
  - `submit` → un formulaire a été soumis
  - `…`

Événements

- Suppression d'un évènement à un nœud :

```
element.removeEventListener(event, function)
```

- Bloquer les autres événements (suivre un lien)

```
event.preventDefault();
```

Ajax

- Javascript permet de faire des requêtes HTTP
  - Pour envoyer des données
  - Récupérer une nouvelle information
  - Avoir des détails sur une information
  - …
- C'est ce que l'on appel de l'ajax :
  - Asynchronous Javascript And Xml

Ajax

Requête HTTP

- **GET**
  - C'est la méthode la plus courante pour demander une ressource. Une requête GET est sans effet sur la ressource, il doit être possible de répéter la requête sans effet.
- **POST**
  - Cette méthode est utilisée pour transmettre des données en vue d'un traitement à une ressource (le plus souvent depuis un formulaire HTML). Le résultat peut être la création de nouvelles ressources ou la modification de ressources existantes.

Ajax

- **Requête GET**

```
var request = new XMLHttpRequest();
request.open('GET', '/my/url/12', true);
request.send();
```

- **Requête `POST`**

```
var request = new XMLHttpRequest();
request.open('POST', '/my/url', true);
request.setRequestHeader('Content-Type',
'application/x-www-form-urlencoded; charset=UTF-8');
request.send("foo=bar&lorem=ipsum");
```

Ajax

- Réponse

```
var request = new XMLHttpRequest();
request.open('GET', '/my/url/12', true);
request.onload = function() {
  if (this.status >= 200 && this.status < 400) {
    // Success!
    var resp = this.response;
  } else {
    // We reached our target server, but it returned an error

  }
};
request.onerror = function() {
  // There was a connection error of some sort
};
request.send();
```