

# Javascript

Les petits conseils de Clara et Thomas

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

# Présentation du langage



# C'est quoi le JavaScript ?

- JavaScript utilisé pour contrôler le comportement d'élément
- langage non typé
- c'est un langage puissant mais peu apprécié dans le domaine du web (malgré son utilité)
- il existe une quantité incroyable de Frameworks que vous pourrez découvrir (NodeJS, Angular, ViewJS, React,...)

# Quand utiliser Javascript ?

- quand vous ne pouvez pas réaliser la tâche dans un autre langage
- à utiliser **avec parcimonie** (tourne côté client et est donc contournable, désactivable)
- pour faire de **l'événementielle** : code exécuté à la survenue d'un événement (réagir à un clic, ...)
- Dans certain cas, soulager la charge du serveur (vérification des éléments avant un envoi important)
- insertion de code HTML

# Intégration au code HTML

- pour lier un fichier Javascript à votre fichier HTML, une ligne suffit:

```
<script src="my_sctrript_location"></script>
```

- la **séparation des fichiers** est très importante en terme de maintenabilité
- on évitera de mettre tout le code Javascript dans un unique fichier

# Programmation événementiels

–

## Les fonctions



# La programmation événementielle

- réagir à différents événements
- Quelques événements :  
`blur/change/checking/click/dblclick/focus/keydown/keypress/.../mousedown/.../resize/scroll/submit`
- `target.addEventListener(event, handler, useCapture);` `useCapture` vaut `false` par défaut
- `target.removeEventListener(event, handler, useCapture)` exactement les même paramètre que lorsqu'on l'a ajouté

```
document.getElementById("myId").addEventListener("click", function () {  
    alert("Button clicked");  
});
```

# La programmation événementielle

- Certaines fonctions peuvent prendre des paramètres qui varient en fonction du contexte dans lequel elle sont utilisée

```
document.getElementById("submitButton").addEventListener("click", function (e) {  
    e.preventDefault();  
    alert("Button clicked but the info were not sent");  
});
```



# Quelques mots sur les fonctions

- Fonction classique :

```
function sayHello(name) {  
    console.log("Hello " + name);  
}  
  
sayHello("World");
```

- Fonction anonyme (très utilisée en Javascript):
  - souvent utilisé en argument de fonction

```
setTimeout(function () {  
    console.log('Execute later after 1 second');  
}, 1000);
```

# Quelques mots sur les fonctions

- Arrow function :

```
setTimeout(() => {  
    console.log("Goodbye World");  
}, 1500);
```

Attention, dans ce type de fonction, "this" n'est pas utilisable !

- Fonction stockée dans des variables :

```
const add = function (a, b) {  
    return a + b;  
};  
  
const add = (a, b) => a + b;
```

# Sélections d'éléments HTML



# Sélection d'éléments

- Les différents sélecteurs (s'applique sur `document` ou n'importe quel élément HTML) :
  - `getElementById("myID")` retourne 1 élément car un 'id' est censé être unique !
  - `getElementsByClassName("myClass")` retourne une liste !
  - `getElementsByTagName("myTag")` retourne une liste !
  - `querySelector(selector)` retourne le premier élément ou une liste avec `querySelectorAll(selector)`
    - les `selector` sont les mêmes qu'en css
      - sélecteur de type : `"h1"`
      - sélecteur de classe : `".myClass"`
      - sélecteur d'id : `"#myID"`
      - bien sûr on peut les enchaîner
- Ces fonctions s'utilisent sur des éléments. Si on veut récupérer un/des éléments depuis le HTML:
  - `document.getElemBy...("filter")`

On peut les chaîner:

- `document.getByTagName("myTag").getByClassName("myClass");`

# Sélection d'éléments

- Le résultat d'une sélection peut être stocker dans une variable, que l'on pourra utiliser pour modifier l'élément sélectionné:

```
var myElem = document.getElementById("myId");  
myElem.style.color = "blue";
```

# Modification du style d'éléments

- Changer le style :
  - `elem.style.property = "newProperty"`
    - `document.getElementById("myId").style.color = "blue"`
- Changer un attribut :
  - `elem.setAttribute(attribut, value)`
    - `document.setAttribute("class", "myClass")`
  - `elem.getAttribute(attributName)` retourne la valeur de l'attribut si l'élément `elem` le possède, null sinon

# Quelques conseils et bonnes pratiques



# Bon à savoir

- Test d'égalité :

- `==` : effectue une conversion de type et compare la valeur
- `===` : effectue une comparaison de valeur sans conversion de type

```
console.log(0 == '0'); //true
console.log(0 === '0'); //false
```

- déclaration de variables -> PAS TYPE

- `var` : portée globale (accessible dans tout les blocs après qu'il a été déclaré)
- `let` : portée de bloc (uniquement accessible dans le bloc ou il a été déclaré)
- `const` : constante (accessible dans tout les blocs après qu'il a été déclaré)

```
var x = 10;
// Here x is 10
{
  const x = 2;
  // Here x is 2
  {
    let x = 4;
    // Here x is 4
  }
  // Here x is 2
}
// Here x is 10
```

- conversion de type automatique

- évitez d'écrire du code qui se base sur des conversions implicites

- `forEach`

```
array1.forEach((element, index) => {
  console.log(element);
  console.log(index);
});
```



# Les bonnes pratiques

- en Javascript, on utilise beaucoup le Camel Case (ex: mySuperVariable) pour les variables et les fonctions
- toujours documenter son code !


```
/**
 * Calculate the area of a rectangle.
 * @param {Int} length the length of the rectangle
 * @param {Int} width  the length of the rectangle
 * @return {Int}       the area of the rectangle
 */
function calcArea(length, width) {
    return length * width;
}
```

# Le debugging

- la console du navigateur est votre meilleure amie !
- vous pouvez tester des bouts de code dans le terminal avec la commande: `js monScript.js`
- N'hésitez pas à abuser du `console.log(output)` pour avoir un aperçu de ce que vous faite

# Pour aller plus loin

Les classes, les objets et l'asynchrone



# Des requêtes asynchrones ?

- Javascript vous permet d'effectuer AJAX:
  - Récupérer des données d'un serveur après le chargement de la page
  - Actualiser un contenu sans recharger la page
  - Envoyer des données au serveur en arrière plan

```
function getInfo(){
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById("demo").innerHTML = this.responseText;
        }
    };
    xhttp.open("GET", "ajax_info.txt", true);
    xhttp.send();
}
```

# Les classes et les objets

- On peut créer des classes en Javascript (si on veut pouvoir créer plusieurs objets d'une même classe)
- on peut créer des objets sans classes s'ils ont vocation d'être unique.

```
const Rectangle = class {  
  constructor(height, width) {  
    this.height = height;  
    this.width = width;  
  }  
  
  area() {  
    return this.height * this.width;  
  }  
};
```

```
class Rectangle {  
  constructor(height, width) {  
    this.height = height;  
    this.width = width;  
  }  
  
  area() {  
    return this.height * this.width;  
  }  
};
```

```
var rectangle2 = {  
  height : 2,  
  width : 3,  
};
```

- Dans les deux cas, on peut toujours ajouter des attributs à notre objet plus tard (attention, la classe n'est pas modifiée)

```
rectangle2.area = rectangle.height * rectangle.width;  
rectangle.area = rectangle.area();
```

# Vos questions ?

Et après on rajoute quelques fonctionnalités au site de poti-animaux