# javascript :

# agir sur les éléments

# au programme...

# au programme...

# sélection d'éléments

Pour manipuler les éléments de la page il faut au préalable les sélectionner.

La sélection d'éléments peut se faire

- par son attribut `id`

# sélection d'éléments

Pour manipuler les éléments de la page il faut au préalable les sélectionner.

La sélection d'éléments peut se faire

- par son attribut `id`
- par son attribut `class`

# sélection d'éléments

Pour manipuler les éléments de la page il faut au préalable les sélectionner.

La sélection d'éléments peut se faire

- par son attribut `id`
- par son attribut `class`
- par sa balise

# sélection d'éléments

Pour manipuler les éléments de la page il faut au préalable les sélectionner.

La sélection d'éléments peut se faire

- par son attribut `id`
- par son attribut `class`
- par sa balise
- par un **sélecteur CSS**

# sélection par l'identité

## getElementById

la méthode **getElementById** de l'objet **document** sélectionne l'unique élément du document dont l'id est fourni en paramètre, ou **null** si aucun

- le résultat est un objet élément (de type HTMLElement)

```
var element = document.getElementById("joconde");
```

*exemple-selection.html* - *exemple-selection.js*

# sélection par l'attribut classe

## `getElementsByClassName`

la méthode `getElementsByClassName` sélectionne les éléments dont la classe est fournie en paramètre

# sélection par l'attribut classe

## `getElementsByClassName`

la méthode `getElementsByClassName` sélectionne les éléments dont la classe est fournie en paramètre

- peut s'invoquer sur l'objet `document` ou sur un objet élément, dans le second cas seuls les éléments descendants sont sélectionnés

# sélection par l'attribut classe

## `getElementsByClassName`

la méthode `getElementsByClassName` sélectionne les éléments dont la classe est fournie en paramètre

- peut s'invoquer sur l'objet `document` ou sur un objet élément, dans le second cas seuls les éléments descendants sont sélectionnés
- a pour résultat la liste des éléments sélectionnés
  $\longrightarrow$ se manipule comme un tableau non mutable
  cette liste est dynamique

```
                          // tous les éléments de classe 'remarque'
var classList = document.getElementsByClassName("remarque");
                         // tous les éléments de classe 'droite' et '
                            encadre'
var eltList1 = document.getElementsByClassName("droite encadre");
                         // les mêmes éléments
var eltList2 = document.getElementsByClassName("encadre droite");

                         // éléments descendants de 'unique' de classe
                            'droite' et 'encadre'
var elt = document.getElementById("unique");
var eltLis3 = elt.getElementsByClassName("droite encadre");
```

# sélection par balise

## `getElementsByTagName`

la méthode `getElementsByTagName` sélectionne les éléments dont la balise est fournie en paramètre

se comporte comme `getElementsByClassName`.

```
                          // tous les <div> du document
var divList = document.getElementsByTagName("div");
divList.length;

                          // toutes les <img> descendants de 'section1'
var sec1 = document.getElementById("section1");
var sec1ImgList = sec1.getElementsByTagName("img");
sec1ImgList[0];           // -> le premier élément sélectionné
```

# sélection par sélecteurs CSS

## querySelectorAll

la méthode `querySelectorAll` sélectionne les éléments retenus par le sélecteur CSS fourni en paramètre

# sélection par sélecteurs CSS

## querySelectorAll

la méthode `querySelectorAll` sélectionne les éléments retenus par le sélecteur CSS fourni en paramètre

- peut s'invoquer sur l'objet `document` ou sur un objet élément, dans le second cas seuls les éléments descendants sont sélectionnés

# sélection par sélecteurs CSS

## querySelectorAll

la méthode `querySelectorAll` sélectionne les éléments retenus par le sélecteur CSS fourni en paramètre

- peut s'invoquer sur l'objet `document` ou sur un objet élément, dans le second cas seuls les éléments descendants sont sélectionnés
- a pour résultat la liste des éléments sélectionnés, cette liste n'est pas dynamique

# sélection par sélecteurs CSS

## querySelectorAll

la méthode `querySelectorAll` sélectionne les éléments retenus par le sélecteur CSS fourni en paramètre

- peut s'invoquer sur l'objet `document` ou sur un objet élément, dans le second cas seuls les éléments descendants sont sélectionnés
- a pour résultat la liste des éléments sélectionnés, cette liste n'est pas dynamique

## querySelector

`querySelector` ne fournit que le premier élément de la liste

```
    // tous les éléments <img> du document  emboités dans un
    //    élément <div> de classe 'exercice'
var listElement = document.querySelectorAll("div.exercice img");

    // le premier de ces éléments
var premier = document.querySelector("div.exercice img");

    // tous les liens ciblant un ".pdf" descendants
    //    de l'élément d'id 'exo1'
var elmt = document.getElementById("exo1");
var listElem = elemt.querySelectorAll('a[href$=".pdf"]');
   // ou encore
var listElmBis = document.querySelectorAll('#exo1 a[href$=".pdf"]');
```

NB :

- les sélecteurs avec les pseudo-classes `:link`, `:visited` et des pseudo-éléments ne sont pas acceptés
  la liste résultat est toujours vide

# équivalences des sélecteurs

```
document.getElementById("unId")
```

≡

```
document.querySelector("#unId")
```

# équivalences des sélecteurs

```
document.getElementById("unId")
```

≡

```
document.querySelector("#unId")
```

```
document.getElementsByClassName("uneClasse")
```

≡

```
document.querySelectorAll(".uneClasse")
```

# équivalences des sélecteurs

```
document.getElementById("unId")
≡
document.querySelector("#unId")
```

```
document.getElementsByClassName("uneClasse")
≡
document.querySelectorAll(".uneClasse")
```

```
document.getElementsByTagName("uneBalise")
≡
document.querySelectorAll("uneBalise")
```

# équivalences des sélecteurs

```
document.getElementById("unId")
```
≡
```
document.querySelector("#unId")
```

```
document.getElementsByClassName("uneClasse")
```
≡
```
document.querySelectorAll(".uneClasse")
```

```
document.getElementsByTagName("uneBalise")
```
≡
```
document.querySelectorAll("uneBalise")
```

mais les listes ne sont pas dynamiques avec `querySelectorAll`

# comparaison

- `querySelector(All)` permet de sélectionner plus simplement des éléments :

  *"la première image contenue dans l'élément d'id `section1`"*

# comparaison

- `querySelector(All)` permet de sélectionner plus simplement des éléments :

*"la première image contenue dans l'élément d'id `section1`"*

```
var sec1 = document.getElementById("section1");
var sec1ImgList = sec1.getElementsByTagName("img");
var img = sec1ImgList[0];
```

ou

```
img = document.getElementById("section1").getElementsByTagName("img")[0];
```

# comparaison

- `querySelector(All)` permet de sélectionner plus simplement des éléments :

*"la première image contenue dans l'élément d'id `section1`"*

```
var sec1 = document.getElementById("section1");
var sec1ImgList = sec1.getElementsByTagName("img");
var img = sec1ImgList[0];
```

ou

```
img = document.getElementById("section1").getElementsByTagName("img")[0];
```

devient

```
var img = document.querySelector("#section1 img");
```

# comparaison

- `querySelector(All)` permet de réaliser des sélections difficiles autrement :

*"tous les éléments de titre `h1` et `h2`"*

# comparaison

- `querySelector(All)` permet de réaliser des sélections difficiles autrement :

*"tous les éléments de titre `h1` et `h2`"*

```
var lesTitres = document.querySelectorAll("h1,h2");
```

# comparaison

- `querySelector(All)` permet de réaliser des sélections difficiles autrement :

*"tous les éléments de titre h1 et h2"*

```
var lesTitres = document.querySelectorAll("h1,h2");
```

plutôt que

```
var lesH1 = document.getElementsByTagName("h1");
var lesH2 = document.getElementsByTagName("h2");
var lesTitres = lesH1.concat(lesH2);
```

qui **ne conserve pas** l'ordre d'apparition dans le document

# comparaison

- mais `querySelectorAll`/`querySelector` **moins efficaces** que leurs équivalents
  voir *ce comparatif des performances*

# au programme...

# au programme...

# propriétés des éléments

les objets éléments obtenus par sélection possèdent des propriétés manipulables :

- attributs
- contenu
- style css

il est possible d'agir sur ces propriétés après sélection de l'élément

# manipuler les attributs

- les attributs html sont des propriétés

# manipuler les attributs

- les attributs html sont des propriétés
  - même nom, en minuscules, avec « conversion camelback »

# manipuler les attributs

- les attributs html sont des propriétés
  - même nom, en minuscules, avec « conversion camelback »
  - l'attribut `class` devient `className`

# manipuler les attributs

- les attributs html sont des propriétés
  - même nom, en minuscules, avec « conversion camelback »
  - l'attribut `class` devient `className`
  - la valeur peut être `string`, `number` ou `boolean` selon attribut

# manipuler les attributs

- les attributs html sont des propriétés
  - même nom, en minuscules, avec « conversion camelback »
  - l'attribut `class` devient `className`
  - la valeur peut être `string`, `number` ou `boolean` selon attribut
- on peut également utiliser `getAttribute` et `setAttribute`

# manipuler les attributs

- les attributs html sont des propriétés
  - même nom, en minuscules, avec « conversion camelback »
  - l'attribut `class` devient `className`
  - la valeur peut être `string`, `number` ou `boolean` selon attribut
- on peut également utiliser `getAttribute` et `setAttribute`
  - dans ce cas la valeur est **toujours** une chaîne de caractères

```
    // récupération de l'élément d'id 'celebre'
var monImage = document.getElementById("celebre");
    // accès à son attribut 'alt'
var texteAlt = monImage.alt;
    // modification d'attributs
monImage.src = "img/joconde.jpg";
monImage.alt = "la Joconde";
    // ajout d'une classe à celles existantes
monImage.className = monImage.className + " flottantdroite";
```

*exemple-propriete.html* – *exemple-propriete.js*

# manipuler le contenu

## innerHTML

la propriété **innerHTML** représente le contenu HTML d'un élément

# manipuler le contenu

## innerHTML

la propriété **innerHTML** représente le contenu HTML d'un élément

- en lecture, sa valeur **contient** les balises
- en modification, son contenu **est interprété** par le navigateur

# manipuler le contenu

## innerHTML

la propriété **innerHTML** représente le contenu HTML d'un élément

- en lecture, sa valeur **contient** les balises
- en modification, son contenu **est interprété** par le navigateur

## textContent

la propriété **textContent** représente le contenu textuel d'un élément

# manipuler le contenu

## innerHTML

la propriété **innerHTML** représente le contenu HTML d'un élément

- en lecture, sa valeur **contient** les balises
- en modification, son contenu **est interprété** par le navigateur

## textContent

la propriété **textContent** représente le contenu textuel d'un élément

- en lecture, sa valeur **ne contient pas** les balises HTML
- en modification, son contenu **n'est pas interprété** par le navigateur

```
var element = document.getElementById("exemple");
var htmlText = element.innerHTML;
alert(htmlText);  // -> "<p> Ceci est <strong>
                  //  mon</strong> contenu.</p>"

var txt = element.textContent;
alert(txt);       // -> "Ceci est mon contenu"
```

```
...
<div id="exemple">
   <p>Ceci est
        <strong>mon
        </strong>
      contenu.
   </p>
</div>
...
```

```
  // modifie le contenu HTML de l'élément et donc son 'affichage'.
  //   La balise <em> est interprétée.
elemt.innerHTML = "un <em>autre</em> contenu";

  // modifie le contenu texte de l'élément et donc son 'affichage.'
  // Le texte <strong> N'est PAS interprété.
elemt.textContent = "un contenu <strong>texte</strong>";
```

*exemple-text-inner.html* - *exemple-text-inner.js*

# agir sur les propriétés CSS

- la propriété `style` d'un élément permet de **modifier** les propriétés CSS pour cet élément

# agir sur les propriétés CSS

- la propriété **`style`** d'un élément permet de **modifier** les propriétés CSS pour cet élément
- on utilise directement le nom de la propriété CSS après « conversion camelback » si nécessaire
  - `font-size` ⇝ **`fontSize`**
  - `border-right-style` ⇝ **`borderRightStyle`**, etc.

# agir sur les propriétés CSS

- la propriété **`style`** d'un élément permet de **modifier** les propriétés CSS pour cet élément
- on utilise directement le nom de la propriété CSS après « conversion camelback » si nécessaire
  - `font-size` ⇝ **`fontSize`**
  - `border-right-style` ⇝ **`borderRightStyle`**, etc.
- les valeurs sont toujours des chaînes de caractères

# agir sur les propriétés CSS

- la propriété **`style`** d'un élément permet de **modifier** les propriétés CSS pour cet élément
- on utilise directement le nom de la propriété CSS après « conversion camelback » si nécessaire
  - `font-size` ⇝ **`fontSize`**
  - `border-right-style` ⇝ **`borderRightStyle`**, etc.
- les valeurs sont toujours des chaînes de caractères
- les unités doivent être précisées

```
    // sélection de l'élément voulu
var element = document.getElementById("exemple");

    // modification de certaine propriétés CSS
    //   'l'affichage' est immédiatement impacté
element.style.fontWeight = "bold";
element.style.fontSize = "20px";      // ne pas oublier l'unité
element.style.marginLeft = "50px";
element.style.marginTop = "2%";
element.style.backgroundColor = "rgba(128,0,0,0.5)";

var l = element.style.marginLeft;    // /!\ 'string' avec les unités
var nouveauL = l + 100;                  // -> "50px100" !!!
var valNouvL = parseInt(l)+100;          // -> 110
element.style.marginLeft = valNouvL+"px"; //ne pas oublier l'unité !
```

*exemple-modification-style.html*

# accès aux valeurs du style

**Attention**

`style` **ne permet pas** d'accéder aux valeurs des propriétés définies dans une feuille de style,
accès seulement aux propriétés définies dans le document HTML ou via `style`

il faut utiliser `getComputedStyle`

# style « calculé »

## getComputedStyle

la méthode **getComputedStyle** de l'objet **window** permet d'obtenir un objet regroupant l'ensemble des valeurs des propriétés CSS appliquées par le navigateur pour un élément

# style « calculé »

## getComputedStyle

la méthode **getComputedStyle** de l'objet **window** permet d'obtenir un objet regroupant l'ensemble des valeurs des propriétés CSS appliquées par le navigateur pour un élément

les propriétés CSS

# style « calculé »

## `getComputedStyle`

la méthode `getComputedStyle` de l'objet `window` permet d'obtenir un objet regroupant l'ensemble des valeurs des propriétés CSS appliquées par le navigateur pour un élément

les propriétés CSS

- ont le même nom que précédemment

  pas de « raccourci » autorisé : `margin` interdit, utiliser `marginLeft`, ...

# style « calculé »

## `getComputedStyle`

la méthode **`getComputedStyle`** de l'objet **`window`** permet d'obtenir un objet regroupant l'ensemble des valeurs des propriétés CSS appliquées par le navigateur pour un élément

les propriétés CSS

- ont le même nom que précédemment

  pas de « raccourci » autorisé : `margin` interdit, utiliser `marginLeft`, ...
- sont en **lecture seule**

# style « calculé »

## `getComputedStyle`

la méthode `getComputedStyle` de l'objet `window` permet d'obtenir un objet regroupant l'ensemble des valeurs des propriétés CSS appliquées par le navigateur pour un élément

les propriétés CSS

- ont le même nom que précédemment

  pas de « raccourci » autorisé : `margin` interdit, utiliser `marginLeft`, ...
- sont en **lecture seule**
- s'expriment en unité **absolue** (*px*,...)

```
    // sélection de l'élément voulu
var element = document.getElementById("exemple");

    // récupération du style calculé
var computed = window.getComputedStyle(element);
var marge = computed.marginLeft;          // avec les unités, en px
var couleur = computed.backgroundColor;   // rgb(...,...,...)

   // appliquer un facteur d'échelle de 10% à la largeur :
var largeur = parseInt(computed.width);  // récupère valeur calculée
var nvLargeur = Math.floor(largeur*1.10);// ajoute 10%
element.style.width = nvLargeur + "px";    // modifie style appliqué
```

*exemple-modification-style.html* - *exemple-computed-style.js*

# manipuler les propriétés CSS

pour manipuler les valeurs des propriétés CSS d'un élément, on utilise

# manipuler les propriétés CSS

pour manipuler les valeurs des propriétés CSS d'un élément, on utilise

- `getComputedStyle` pour **accéder** aux valeurs

# manipuler les propriétés CSS

pour manipuler les valeurs des propriétés CSS d'un élément, on utilise

- `getComputedStyle` pour **accéder** aux valeurs
- `style` pour **modifier** les valeurs

## à suivre...

# javascript : événements