Création des objets "natifs":

natif	L'instanciation de la majorité des objets est pris en charge par le navigateur lors de l'analyse du flux html. Principaux objets : window, document, noeuds éléments , noeuds texte. On peut aussi créer de manière dynamique de nouveaux éléments (voir paragraphe création d'éléments)
jquery	Vous devez vous-même i**nstanciez** un objet Jquery . Pour cela, vous devez utiliser **la méthode constructeur $** $(regle css) : crée UN objet Jquery relié à 0 à n objets du dom Ex : $("#div1") : objet jquery relié à l'objet du dom ayant pour id div1 $(".bleu") : objet jquery relié aux objets du dom ayant pour class bleu $("input:checked") : objet jquery relié aux objets du dom de tag input et dont l'état est coché. $("input[value=5]"): objet jquery relié aux objets du dom de type input et dont l'attribut value a pour valeur 5 On peut aussi construire un objet jquery à l'aide d'un objet du dom: $(document), $(window), $(document.body),.....

Attention: L'objet jquery est relié à des objets du DOM. Il faut donc que ceux-ci aient été instanciés avant que l'on puisse construire les objets souhaités. On ne construira donc ces objets que lorsque le DOM sera prêt

sélection des objets:

natif	document.getElementById("id") , document.getElementsByName("nom") , document.getElementsByTagName("tag"), document.getElementsByClassName("class"), document.querySelector("selecteur css") document.querySelectorAll("selecteur css") . Ces 4 dernières méthodes s'appliquent aussi à tout nœud élément du DOM + parcours du dom : childNodes, firstChild, lastChild, nextSibling, previousSibling. **Attention au noeud vide (voir traversée du DOM)**
jquery	La sélection des objets du DOM s'effectue lors de la construction de l'objet Jquery (voir paragraphe précédent) ou à l'aide des méthodes permettant de traverser le DOM . Les méthodes find, first, last, has, filter, eq,contents , .. permettent de réduire l'ensemble des éléments sélectionnés. Ex $("div").first() : permet de sélectionner la première div $("p").eq(3) permet de sélectionner le 4ème paragraphe

La gestion de l'évènement load

natif	Tant que le DOM n'est pas construit, on ne peut quasiment rien faire. Il faut donc écouter la diffusion de l'évènement load pour pouvoir commencer la mise en place de son code javascript window.addEventListener("load", demarrer,false);
jquery	Il y a une distinction entre l'évènement **ready** et l'évènement **load**. ready, le dom est construit mais tous les éléments externes (notamment les images) ne sont pas obligatoirement chargés. load le dom est construit et tous les éléments externes sont chargés. Généralement on écoute ready $(document).ready(demarrer);

modification d'attribut

natif	On utilise .attribut ou la méthode setAttribute("attribut",valeur);
	● Si on a sélectionné un et un seul élément : element.attribut=valeur
	Ex :
	var image1=document.getElementById("img1")
	image1.src="img1.jpg";
	image1.setAttribute("alt","une belle image");
	● Si on a sélectionné une collection d'éléments, il faut les traiter un par un à l'aide d'une boucle for
	Ex: var lesImages=document.getElementsByTagName("img");
	for (var i=0;i<lesImages.length;i++) {
	lesImages[i].src="img1.jpg"; lesImages[i].alt="une belle image";
	}

jquery	On utilise la méthode **attr({attribut1:valeur,attribut2:valeur,..})** Le paramètre de la méthode attr est un objet JSON Ex : $("#img1").attr({src:"img1.jpg",alt:"une belle image"}); $("img").attr({src:"img1.jpg",alt:"une belle image"});

modification des propriétés css

natif	• Si on a sélectionné un et un seul élément : element.style.proprieteCss=valeur
	Ex :
	var para1=document.getElementById("p1")
	para1.style.border="2px solid red";
	para1.style.fontSize="24px"
	• Si on a sélectionné une collection d'éléments, il faut les traiter un par un à l'aide d'une boucle for
	Ex: var lesParass=document.getElementsByTagName("p");
	for (var i=0;i<lesParas.length;i++) {
	lesParas[i].style.border="2px solid"; lesParas[i].style.fontSize="24px";
	}

jquery	On utilise la méthode **css({propCss1:valeur,propCss2:valeur,..}) (formatJSON)** Ex : $("#p1").css({border:"2px solid red",fontSize::24}); $("p").css({border:"2px solid red",fontSize::24});

ajout ou suppression d'une classe

natif	element setAttribute("class",nomClasse) ou element.className=nomClasse. Inconvénient : on remplace toutes les classes associées à l'objet par une seule. Avec HTML5 est apparue la propriété **classList** qui dispose de méthodes **add, remove,toggle et contains** pour manipuler les classes
jquery	On utilise la méthode **addClass(nomClasse)** pour ajouter une classe et **removeClass(nomClasse)** pour l'enlever. La méthode **hasClass(nomClasse)** permet de savoir si la classe est associée ou non à l'objet

gestion des événements

natif	• on utilise la méthode addEventListener
	• Si on a sélectionné un et un seul élément : element.addEventListener("evenement", fonction, phase)
	Ex :
	var para1=document.getElementById("p1")
	para1.addEventListener("click",agrandir,false);
	
	• Si on a sélectionné une collection d'éléments, il faut les traiter un par un à l'aide d'une boucle for
	Ex: var lesParas=document.getElementsByTagName("p");
	for (var i=0;i<lesParas.length;i++) {
	lesParas[i].addEventListener("click",agrandir,false);
	}
	
	Pour supprimer un écouteur, on fait appel à la méthode **removeEventListener**
	Ex: para1..removeEventListener("click",agrandir,false);

jquery	chaque évènement correspond à une méthode, : **click, change, mouseover, mouseout**,....Certaines méthodes permettent de gérer plusieurs événements : ex **hover** regroupe mouseover et mouseout Ex : $("#p1").click(agrandir); $("#p1").mouseover(rollOver) ; $("#p1").mouseout(rollOut) $("p").click(agrandir); $("p").hover(rollOver,rollOut); On peut aussi utiliser la méthode **on** (notamment pour les événement n'ayant pas de méthode associée) Ex : $("audio").on("ended",terminer) ; **Pour supprimer un écouteur, on utilise la méthode off:** $("p").off() supprime les fonctions écouteurs de tous les évènements associés aux paragraphes $("p").off("click") supprime les fonctions écouteurs de l'évènement click pour les paragraphes $("p").off("click",agrandir) : supprime la fonction écouteur agrandir associée au click sur un paragraphe

Le mot clé this

natif	Dans une fonction écouteur, this represente l'objet qui a écouté l'évènement (c'est à dire l'objet associé à addEvent ou addEventListener). **Dans les fonctions qui ne sont pas liées à l'écoute d'un évènement this représente l'objet window**
jquery	Dans une fonction écouteur, this représente l'objet qui a écouté l'évènement . **Attention , this est un objet du DOM et non pas un objet JQuery.** Vous devez écrire $(this) pour obtenir un objet jquery correspondant à l'objet du dom sélectionné. Dans les fonctions qui ne sont pas liées à l'écoute d'un évènement this représente l'objet window

La traversée du DOM

natif	On peut, à partir d'un noeud élément , accéder à d'autres noeuds ayant un lien hiérarchique avec lui : parentNode: noeud parent, firstChild ou firstElementChild: premier enfant, lastChild ou lastElementChild: dernier enfant, childNodes ou children: enfants, nextSibling ou nextElementSibling:frère suivant, previousSibling ou previousElementSibling: frère précédent.
jquery	On peut s'adresser à une collection et pas seulement à un seul noeud comme en natif. La traversée se fait sans être perturbée par les nœuds vides grâce aux méthodes children() , next() , nextAll(), prev(), prevAll(), siblings(),parent(), parents(),...Attention first() et last() ne ramène pas respectivement le premier et le dernier enfant des objets interrogés mais ne retourne que le premier objet de la collection(respectivement le dernier) Ex: $("p").nextAll() : sélectionnera tous les frères qui se situent après un élément p $("p").nextAll("a") :sélectionnera tous les noeuds a qui se situe après un élément p $("p").first() : sélectionnera le premier élément de la collection $("p:first-child"): sélectionnera les éléments p qui sont premiers enfants $("p").children(":first-child"): sélectionnera les premiers enfants des éléments p

La sélection/modification du texte d'un noeud

natif	on accède à l'élément dont on souhaite le texte puis on obtient le noeud texte à l'aide de firstChild(ou getFirst()) et le contenu grâce à l'attribut nodeValue.
	Ex var eltP1= document.getElementById("p1");
	var texteP1= eltP1.firstChild.nodeValue; // récupération
	eltP1.firstChild.nodeValue="nouveau texte"; // mise à jour
	ou pour les navigateurs au norme : **textContent** (ou innerText pour les autres) ou bien encore innerHTML
	eltP1.innerHTML="nouveau texte" ;

jquery	Il existe une méthode text() qui permet directement de récupérer ou mettre à jour le texte Ex : var texteP1= $("#p1").text(); $("#p1").text("nouveau texte");

La création dynamiques d'éléments du DOM

natif	Il faut d'avoir créer le nouvel objet :
	document.createElement(tag") pour un noeud élément
	document.createTextNode("texte") pour un noeud texte
	Puis il faut définir ses attributs, ses propriétés css et éventuellement son intractivité
	Enfin il faut le ranger dans le dom
	noeudParent.appendChild (nouveau noeud) permet de l'insérer en tant que que dernier enfant du noeud parent choisi
	noeudParent.insertBefore (nouveau noeud, noeud frere) permet de l'insérer en tant que qu'enfant du noeud parent choisi et juste devant le noeud frère passé en deuxième paramètre
	Ex
	var nouvelleImg=document.createElement("img");
	nouvelleImg.src="img1.jpg";
	nouvelleImg.style.border="2px solid red";
	nouvelleImg.style.width="100px";
	document.getElementById("div1").appendChild(nouvelleImg);

jquery	Pour créer le nouvel objet, on fournit le code html en paramètre du constructeur $
	puis on définit ses propriétés css (methode css) et éventuellement son interactivité
	puis on le range dans le dom :
	● on peut soit utiliser une méthode du nouvel objet : appendTo(parent) (s'ajoute comme dernier enfant) , prependTo(parent) s'ajoute comme premier enfant, insertAfter(noeudFrere) (s'ajoute après le noeud frère choisi) ou insertBefore(noeud frère) s'ajoute avant le noeud frère choisi)
	● ou utiliser une méthode qui s'adresse au parent ou au frère : append(nouvel enfant) , prepend(nouvel enfant) , after(nouveau frere) , before(nouveau frère)
	Ex :
	var nouvelleImg=$('');
	nouvelleImg.css({border:"2px solid red",width:100});
	nouvelleImg.appendTo("#div1");
	ou $("#div1").append(nouvelleImg);

La suppression d'éléments

natif	On utilise la méthode removeChild dont dispose tout nœud parent. On ne peut supprimer les enfants que un par un, et évidemment lorsqu'on supprime un nœud, cela a une incidence sur le rang des frères suivants.
	● Pour supprimer un noeud: **noeudASupprimer.parentNode.removeChild(noeudASupprimer)**
	ex : var eltP1=document.getElementById("p1");
	eltP1.parentNode.removeChild(eltP1);
	● Pour supprimer tous les enfants d'un noeud
	while (noeud.hasChildNodes()) {
	noeud.removeChild(noeud.firstChild);
	}
	ex : var eltDiv1=document.getElementById("div1");
	while(eltDiv1.hasChildNodes()) {
	eltDiv1.removeChild(eltDiv1.firstChild);
	}

jquery	Pour supprimer un noeud : **noeud.detach() ou noeud.remove()**
	ex: $("#p1").detach();
	
	Pour supprimer tous les enfants d'un noeud: **noeud.empty();**
	ex: $("#div1").empty();

La mise en place d'effets

natif	Pour qu'il y ait un effet, il faut qu'il s'écoule du temps entre 2 modifications successives d'une propriété. On ne peut donc pas donner cette impression à l'aide d'une boucle, il faut mettre en place un timer qui se chargera à intervalle régulier de modifier la propriété de sa valeur initiale jusqu'à sa valeur finale. Concrètement, on utilise un framework lorsqu'on veut mettre en place des effets
jquery	**hide()** : permet de cacher immédiatement un objet, hide(temps en ms ou slow ou fast) permet de le cacher avec un effet (joue à la fois sur la largeur , la hauteur et sur l'opacité). Même principe pour show **show()** : montre un objet caché en jouant sur la largeur, la hauteur et l'opacité , **fadeOut**(tps ou mot clé) : cache en objet en jouant sur l'opacité, **fadeIn**(tps ou mot clé) : montre un objet en jouant sur l'opacité, **slideUp(tps ou mot clé**) : cache un objet en jouant sur sa largeur,hauteur, **slideDown(tps ou mot clé**) : montre un objet en jouant sur sa largeur , hauteur, Enfin **animate()** permet d'animer une ou plusieurs propriétés sur un temps donné $('#div1').animate({opacity: 0.25, left:'+=50', height: 500},5000); permet de mettre en place un effet sur l'opacité, la position et la hauteur. Les valeurs indiquées sont celles que l'on souhaite avoir à la fin de l'effet. '+=50' indique que l'on veut augmenter de 50 la valeur actuelle de left.