Javascript dans le navigateur

Matthieu Nicolas
Licence Pro CIASIE
Slides par Christophe Bouthier

Plan

- L’accès au DOM
- Les événements
- Les timers et le thread unique

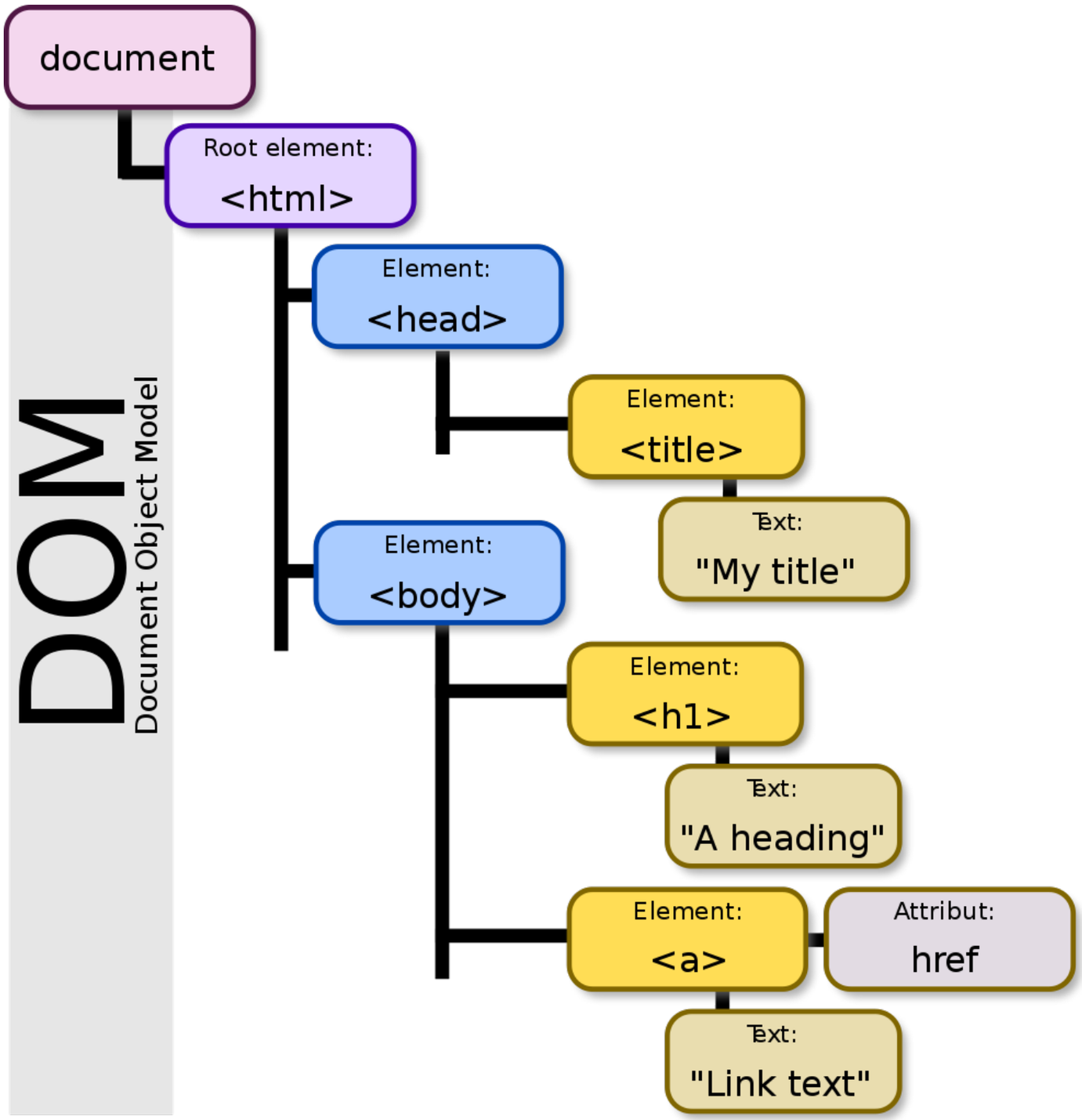
L’accès au DOM

Javascript dans le navigateur

Le DOM

- **D**ocument **O**bject **M**odel
- Hiérarchie de noeuds représentant les éléments HTML de la page web

Le DOM


DOM
Document Object Model
document
Root element:
<html>
Element:
<head>
Element:
<title>
Text:
"My title"
Element:
<body>
Element:
<h1>
Text:
"A heading"
Element:
<a>
Attribut:
href
Text:
"Link text"

Le DOM

- Modifiable :
  - Ajout de noeuds
  - Suppression de noeuds
  - Modification du contenu des noeuds

Accès au DOM

- window.document
  - racine du dom
  - possède les méthodes « globales »
- Différents types de noeuds
  - element : tag HTML
  - text : texte
- hiérarchie de noeuds

Accès aux elements

- document.getElementsByTagName(« tag »)
  - retourne une liste d'éléments
  - nom des éléments (a, img, div…)
- element.getElementsByTagName(« tag »)
  - idem, mais uniquement dans la hiérarchie sous element
- Retourne TOUS les éléments, pas juste les sous-fils

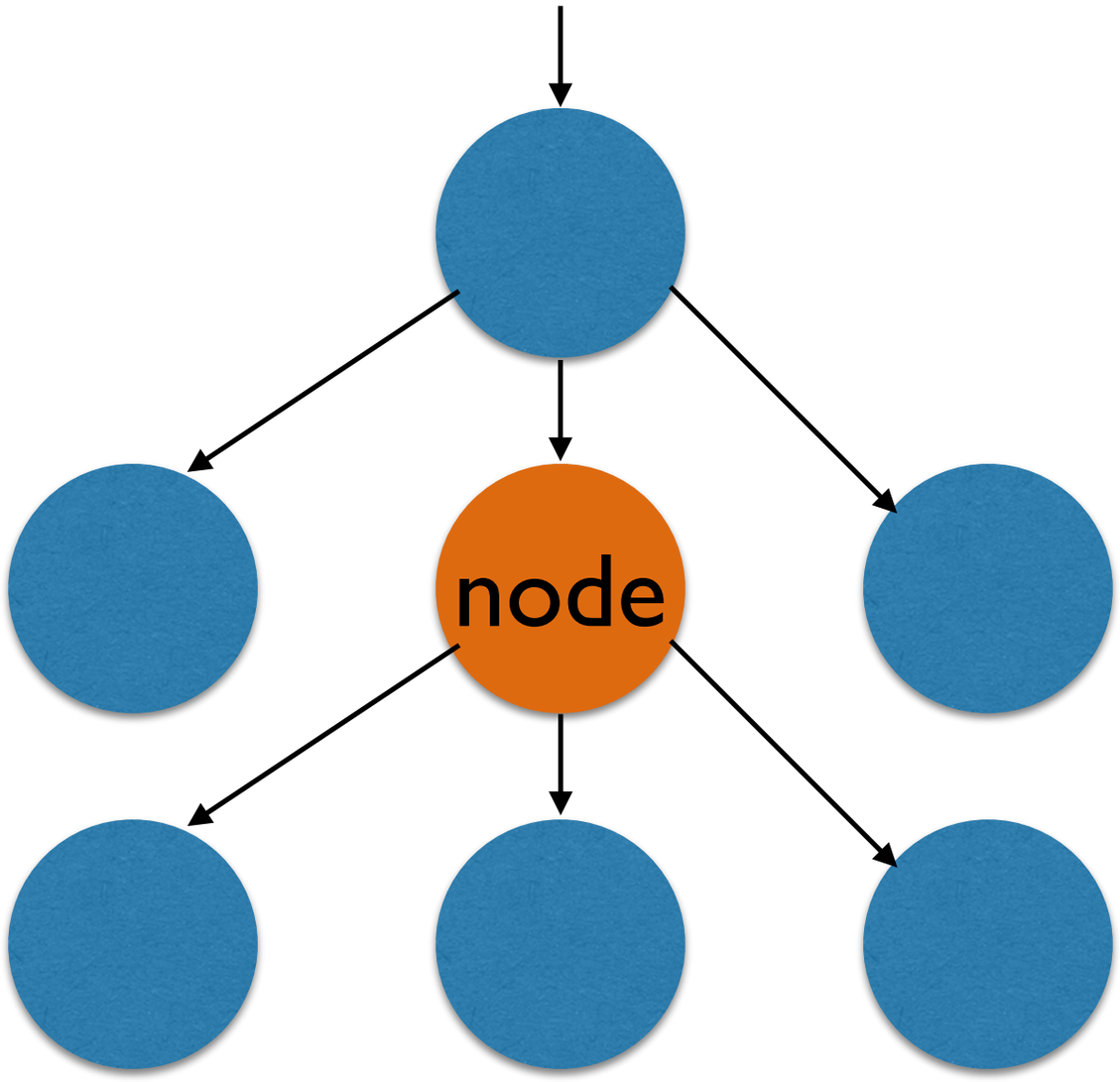
Accès par CSS

- document.getElementByID(« id »)
  - retourne un seul élément
  - ID CSS
- document.getElementsByClassName(« cl »)
  - retourne une liste d’éléments
  - classe CSS

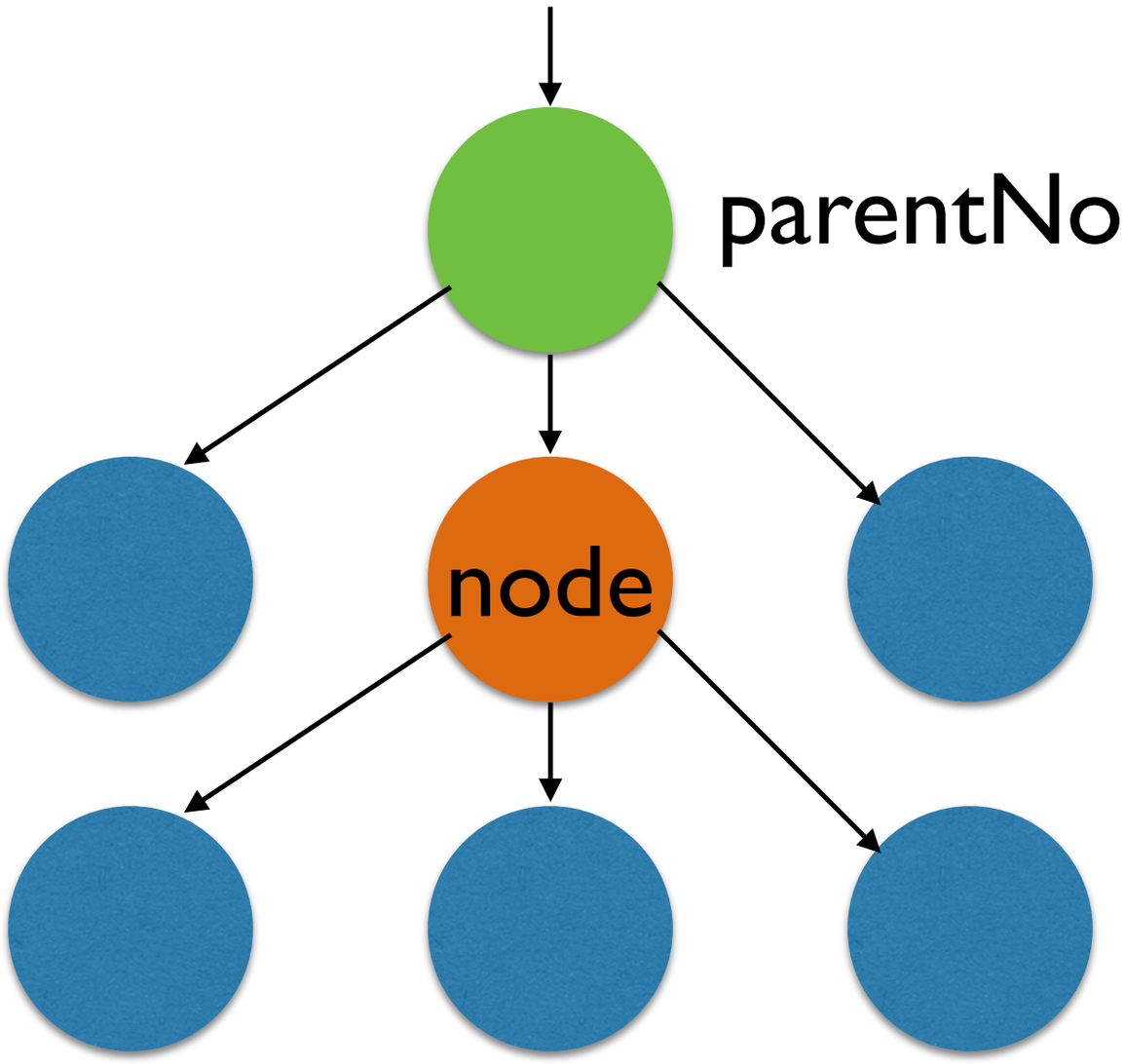
Selector API

- querySelector(« cssSel »)
  - document ou element
  - prend un sélecteur css
  - retourne le premier élément correspondant
- querySelectorAll(« cssSel »)
  - idem, mais retourne tous les éléments correspondants

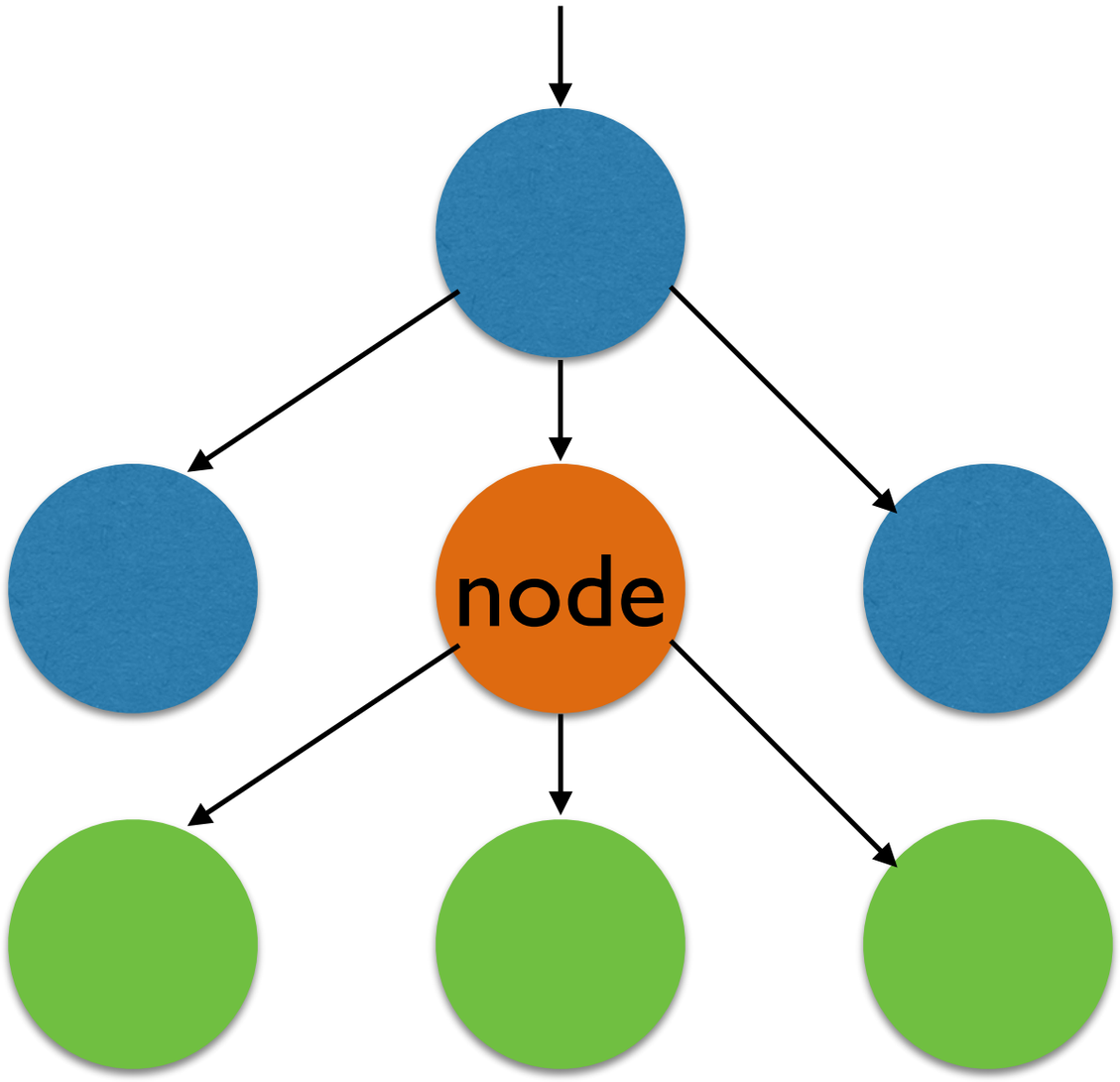
Navigation



Navigation

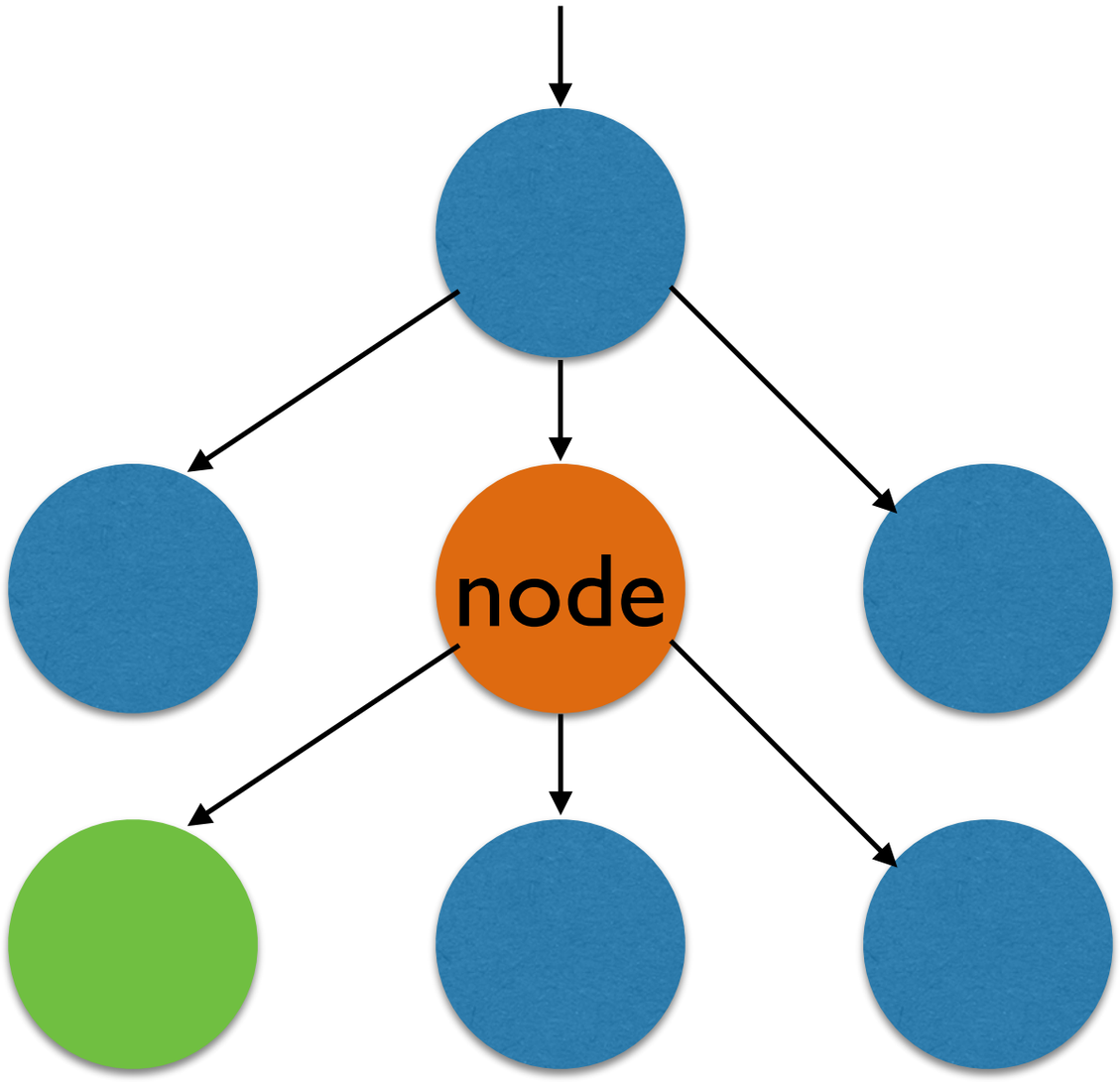


Navigation



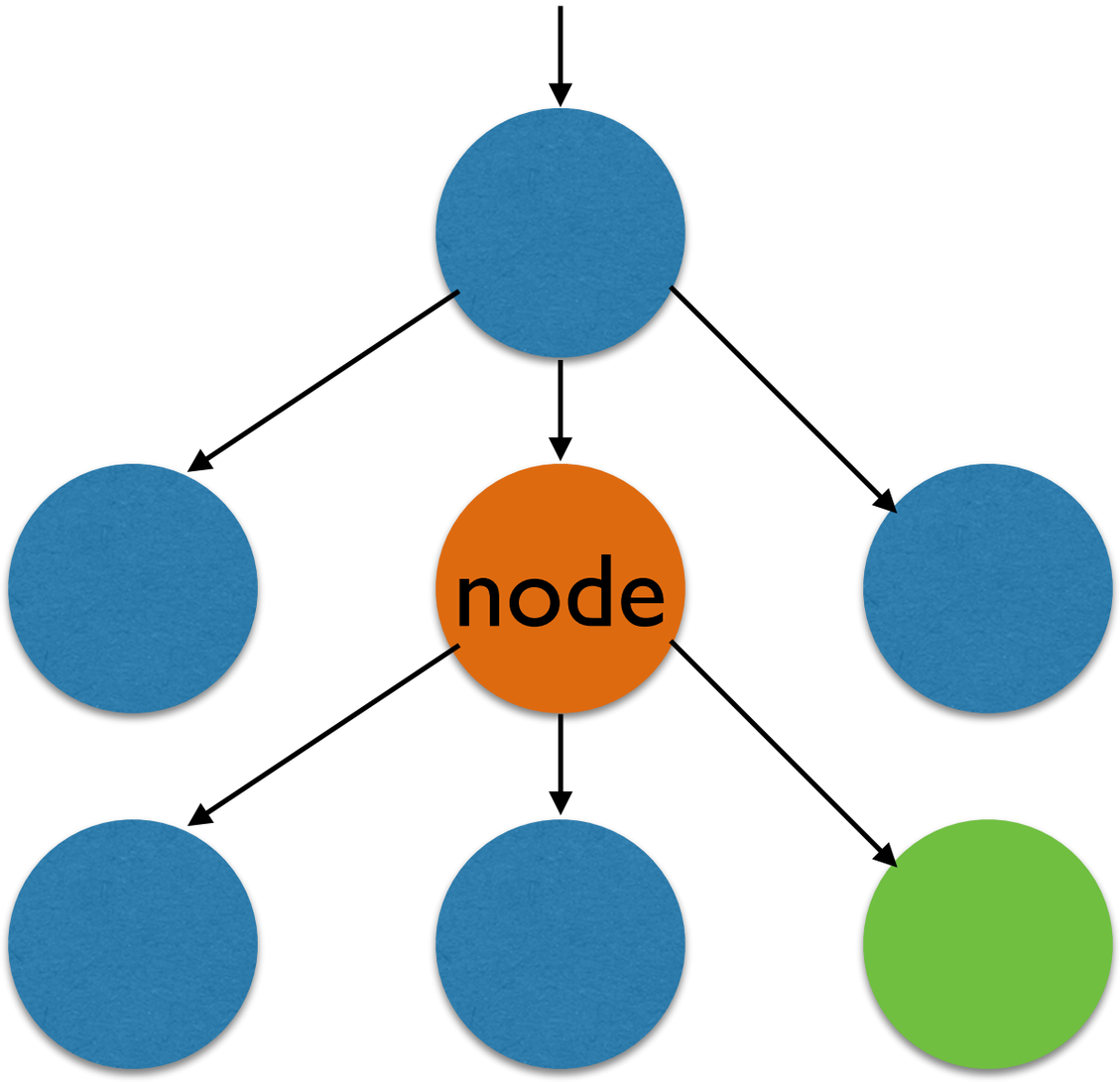
childNodes

Navigation



firstChild

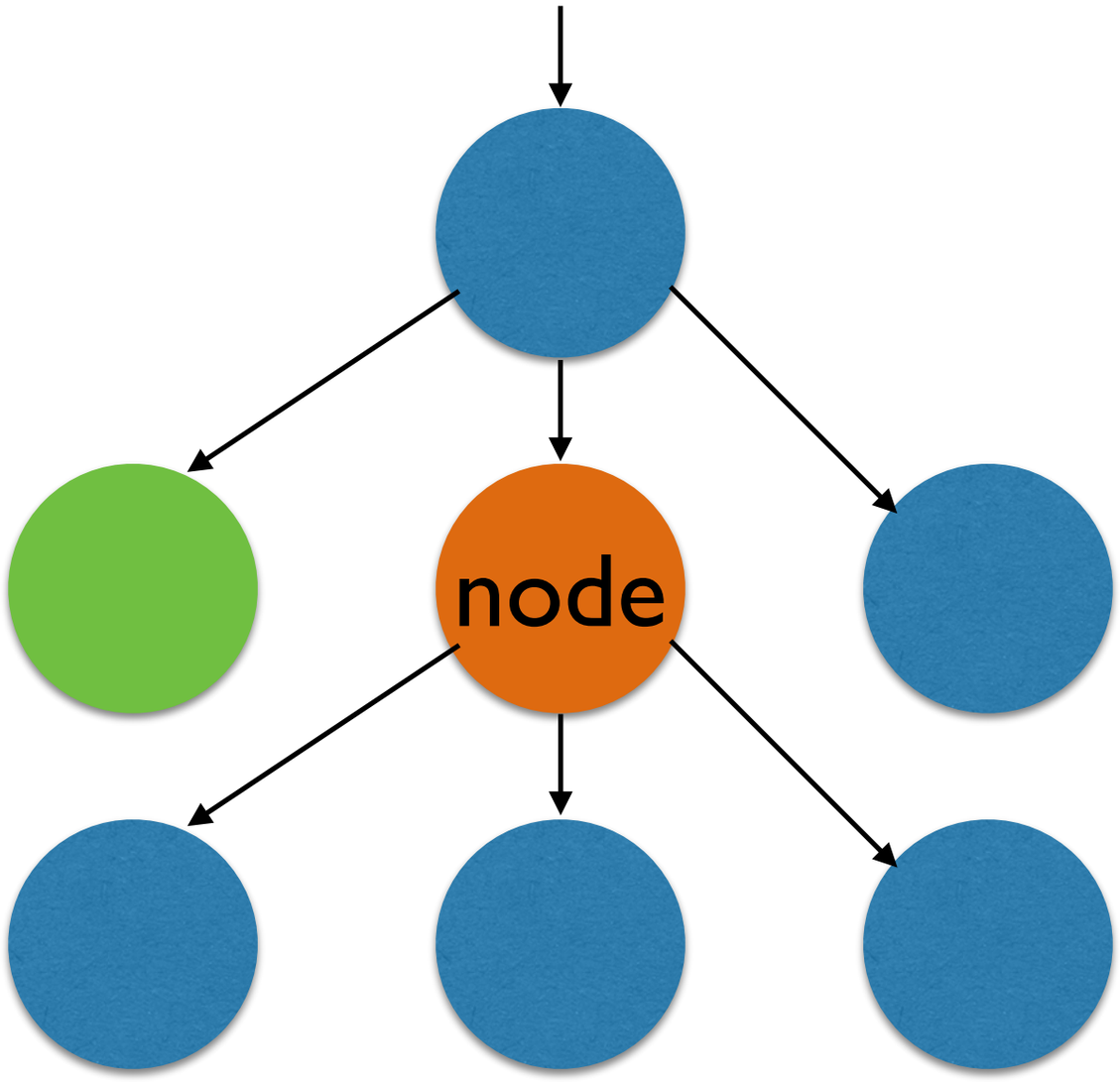
Navigation



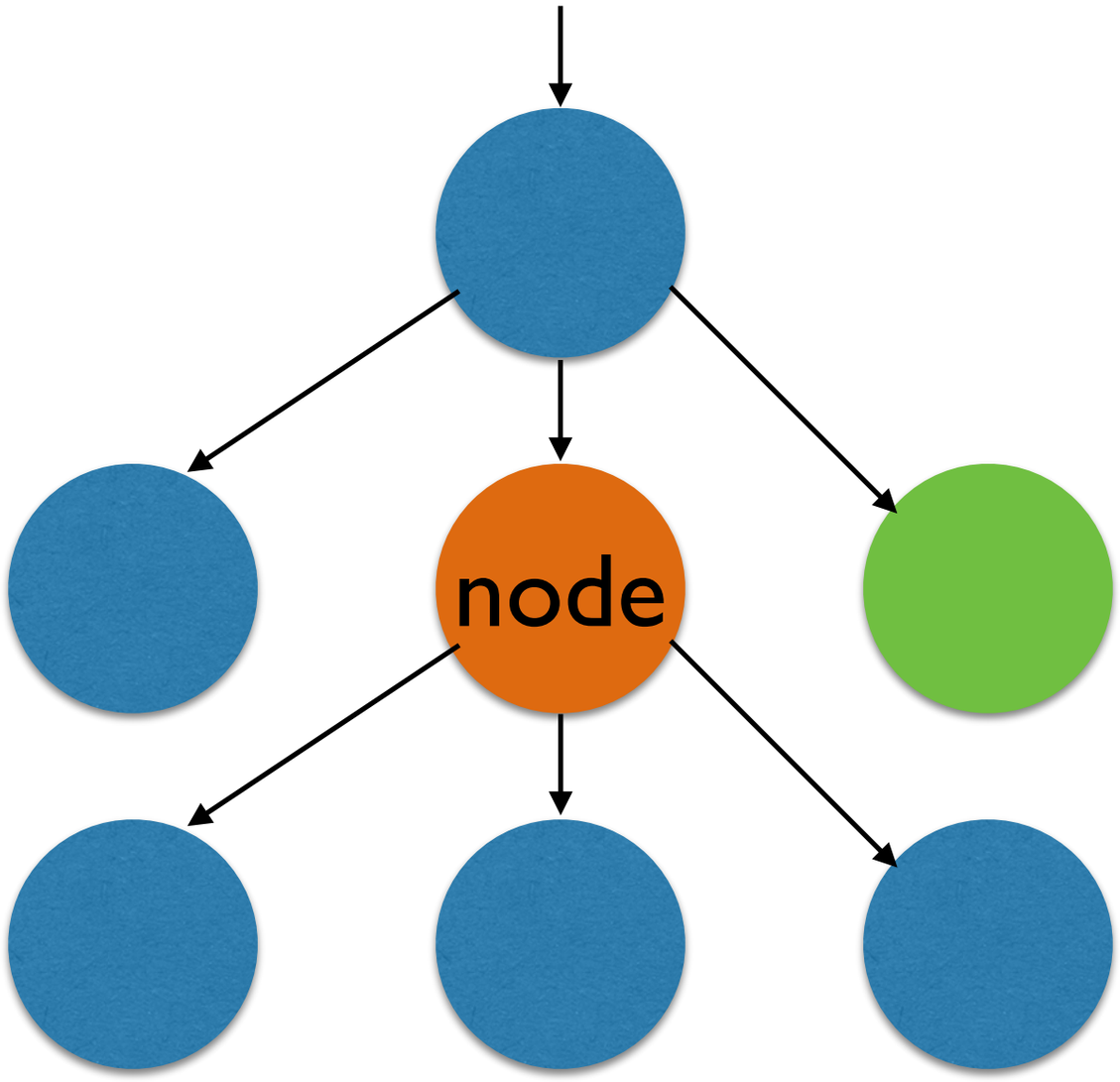
lastChild

Navigation

previousSibling



Navigation



nextSibling

Noeud element

- tagname
  - nom de l'élément (div, img, …)
- innerHTML
  - contenu brut sous forme de String
- className, id
  - classe CSS, id CSS
- Instanceof HTMLElement
  - modif du prototype -> modif de tous les elts

Noeud Texte

- nodeName
  - «#text »
- nodeValue
  - la valeur du texte
- Chaque zone d’espace est un noeud texte !
  - y compris les retours chariots

Création d’un noeud

- e = document.createElement(« div »)
- c = document.getElementById(« id »)
- c.appendChild(e)

Suppression d’un noeud

- e = document.getElementById(« e »)
- c = document.getElementById(« c »)
- c.removeChild(e)

Modification d’un noeud

- e = document.getElementById(« e »)
- e.innerHTML = « test »
- e.innerHTML = « <h1>coucou</h1> »

Accès aux attributs

- Deux types d’accès
  - méthode getAttribute(« name »)
  - .attribut
- exemple
  - img.getAttribute(« src »)
  - img.src

Problèmes

- Noms incompatibles js
  - a.getAttribut(« class »)
- Pas forcément les mêmes valeurs
  - normalisation, notamment URL

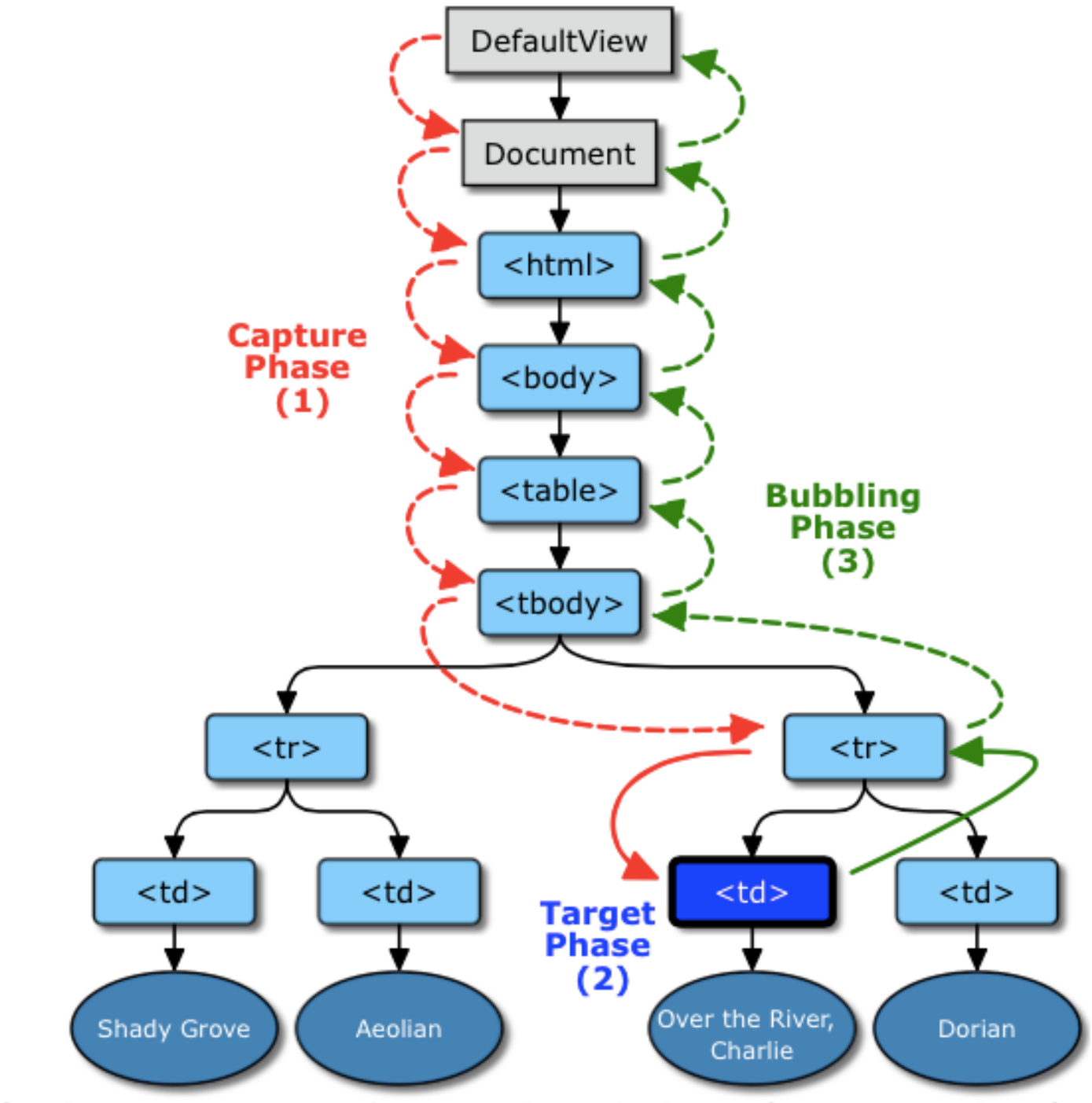
CSS

- attribut « style »
- différences entre inline et feuille CSS
- différences entre déclaré et calculé
- retourne un objet avec tout dedans
- problème avec les unités
- problème de compatibilités cross-browser

Les événements

Javascript dans le navigateur

Event propagation



http://www.w3.org/TR/DOM-Level-3-Events/#event-flow

DOM Level 2

- elt.addEventListener(<evt-name>, <callback>, <capture>);
- elt.removeEventListener(<evt-name>, <callback>, <capture>);

DOM Level 2

- Evt-name
  - click, mousedown, transitionend
- Callback
  - premier param = event
  - this = let
- Capture
  - true -> capture
  - sinon bubbling

Event Properties

- Plusieurs propriétés
  - target -> DOM element où l'event a eu lieu
  - currentTarget -> DOM element qui a appelé le callback
- Mais aussi propriétés spécifiques
  - x, y pour mouseevent, …

Event Methods

- preventDefault()
  - -> supprime le comportement par défaut du navigateur (a -> ouvre une nouvelle page)
- stopPropagation()
  - -> supprime le reste de la propagation, en capture et/ou en bubbling
- Si callback retourne false -> preventDefault et stopPropagation

Delegation d’event

- Bubbling -> event passe au niveau supérieur
- Très utile si on a beaucoup d’elements (liste)
  - on met le callback sur l’élément parent
  - on filtre en fonction de event.target
- Du coup, fonctionne même pour les éléments ajoutés APRES à la liste

Les timers et le thread unique

Javascript dans le navigateur

Contexte

- Browser Event Loop
  - boucle d’event
  - géré par le navigateur
  - c’est elle qui appelle le script
- Mécanisme de callback
  - events
- langage fonctionnel tout à fait pertinent

Thread unique

- Browser Event Loop
- En charge de tous les events
  - Browser events (page load, …)
  - User events (click, …)
  - Network events (ajax)
  - Time events (timers)
- Une seule boucle = un seul thread !

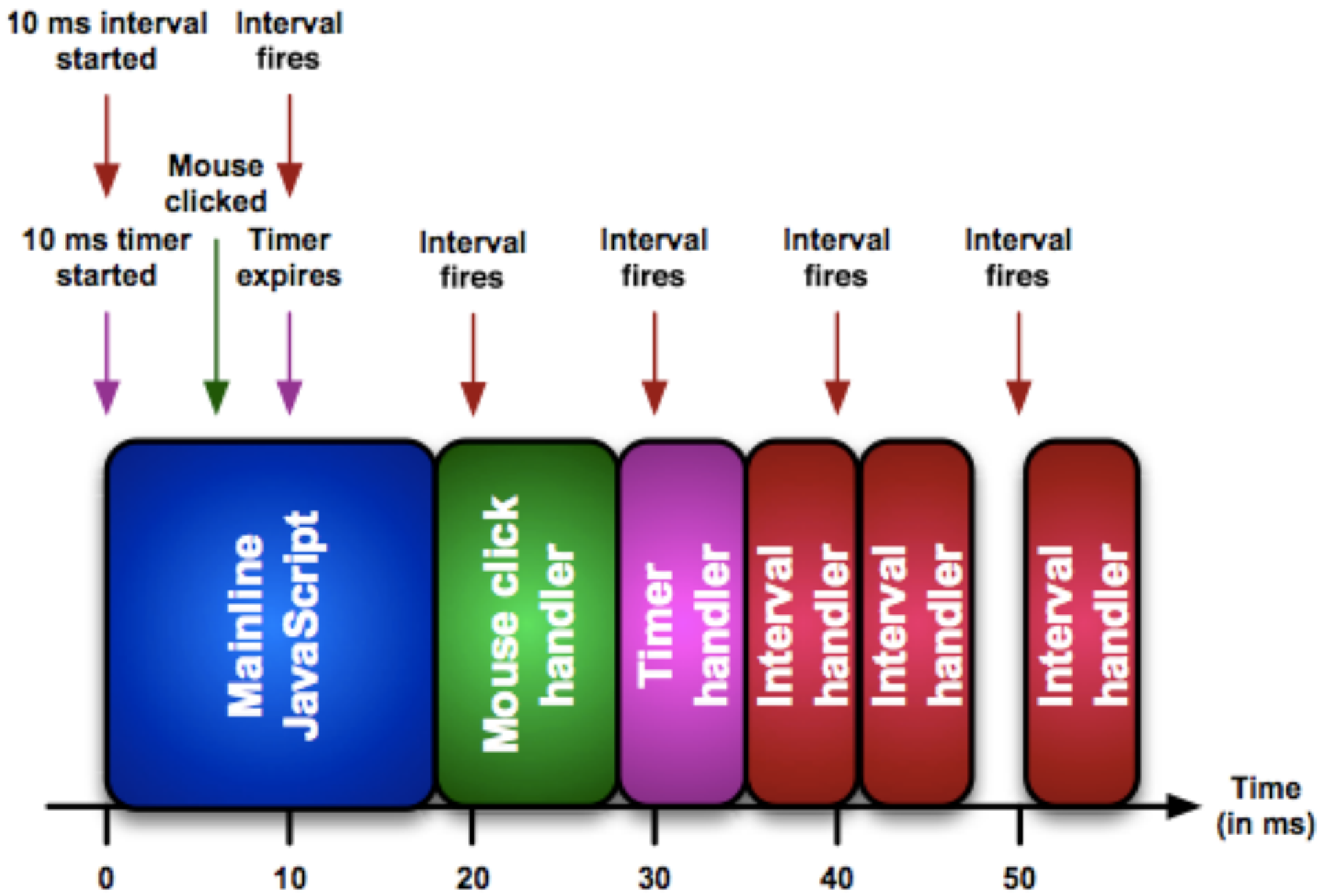
Timers

- setTimeout(callback, delay)
  - met en place un timer unique
  - callback est appelé une seule fois
- setInterval(callback, delay)
  - met en place un timer répétitif
  - callback est appelé tant que le timer n’est pas annulé (clearInterval())

Thread unique !

- Pas de garantie concernant les délais
- Les timers sont enregistrés dans la boucle d’event, et traité les uns après les autres
- Un setInterval n’est mis dans la queue que s’il n’est pas DEJA présent queue

Exemple



source : Secrets of the JavaScript Ninja

Problèmes

- Pas de garantie sur les temps
- Slow Script Warning
  - Si un script prend trop de temps sans laisser la main à la boucle d’event
  - message du navigateur demandant à l’utilisateur s’il veut killer le script
- Plus il y a de timer, plus le navigateur ralenti

Résumé

- Accès au DOM
  - getElementByID, getElementsByClassName
  - querySelector, querySelectorAll
- Event Handlers
  - addEventListener, removeEventListener
- Timers
  - setTimeout, setInterval