PAU CASESNOVES
Centre Integrat de Formació Professional

Desenvolupament d'aplicacions web

Desenvolupament web a l'entorn client

UT 3.1: Generar HTML.

Joan Pons Tugores

Índex de continguts

En la primera part del curs hem utilitzat la consola per mostrar les dades a l'usuari. Evidentment aquesta no és la manera normal de funcionar. El més habitual és que si generam informació des de Javascript la mostrem dins la pàgina, generant HTML o modificant l'existent.

document.write (desaconsellada)

Per entendre el que implica aquesta instrucció hem de parlar una mica sobre el funcionament dels navegadors. Quan el navegador comença a carregar una pàgina HTML genera una representació interna en forma d'arbre on cada element HTML és un node de l'arbre. Així com interpreta el codi, genera l'arbre. El que fa el *document.write* és introduir codi HTML durant aquest procés, i el navegador l'inclou a la seva representació.

Per això quan executam un *document.write* durant la càrrega de la pàgina, el contingut s'afegeix a l'arbre i el veim al navegador.

Quan el navegador acaba de interpretar la pàgina el procés de generació d'aquesta representació interna acaba. A partir d'aquest moment amb el *document.write* no es pot afegir més contingut.

Si executam un *document.write*, per exemple, en pitjar un botó, el navegador tornarà a iniciar el procés de generació de la representació interna de la pàgina amb el que li passam com a paràmetre al document.write. El que farem és substituir el contingut complet de la pàgina per el que escrivim.

document.getElementById

Una manera millor d'alterar el document és modificar el contingut de la representació interna que es fa al navegador. D'aquesta manera ho podem modificar en qualsevol moment, encara que el document ja s'hagi acabat de carregar. És aconsellable, però, esperar a que la interpretació del document hagi acabat. Pròximament veurem en detall com treballar amb el DOM, el Document Object Model. Ara ens basta tenir-ne una petita noció.

Podem seleccionar elements de la pàgina HTML de moltes maneres: per la classe CSS per l'etiqueta, per la posició... El més eficient, però, és assignar a cada element HTML que haguem de manipular des de Javascript un identificador amb l'atribut id. Podem utilitzar aquest atribut per a recuperar-lo i modificar les seves propietats:

```
document.getElementById("identificador de l'element")
```

Això ens torna una referència a l'element identificat per l'identificador. Podem treballar amb aquest "objecte" bàsicament canviant el valor de les seves "propietats". Pràcticament tots els atributs i esdeveniments dels elements HTML es mapegen a propietats:

```
document.getElementById("nom").readonly=true;
```

Converteix l'element "nom" en només lectura.

Si es tracta d'un **control d'un formulari** podem canviar o recuperar el seu valor:

```
<input type="text" id="prova"></input>
document.getElementById("prova").value="Hola";
```

```
console.log(document.getElementById("prova").value);
```

Si es tracta d'**un altre element HTML** podem canviar o recuperar el contingut d'aquest element:

```
<p id="paragraf">Hola</p>

document.getElementById("paragraf").innerHTML="Adeu!"
```

Per poder utilitzar aquesta tècnica l'element que volem modificar ha d'estar creat. Ens podem trobar en dues situacions:

- getElementById s'executa en resposta a una acció de l'usuari. Cap problema. L'usuari només interactua amb la pàgina quan aquesta ja està completament carregada (normalment). Per tant qualsevol element que volguem modificar estarà creat.
- getElementById s'executa durant la càrrega de la pàgina per inicialitzar l'element. Pot ser que l'element encara no s'hagi creat. Solució: Assegurar-nos que l'element s'ha creat. Com? La pàgina dispara l'esdeveniment onload quan s'ha acabat de carregar.

```
window.onload=function(){

}; //dins un script al head o a un fitxer extern enllaçat al head..
```

Per poder fer assignar codi a un esdeveniment, aprofitam que podem assignar funcions a variables, en aquest cas a l'esdeveniment

```
function saluda(){
```

```
        Alert("Hola");
}
function saludaPersonalitzat(nom){
        Alert("Hola"+nom);
}
window.onload=function(){
        document.getElementById("boto").onclick=saluda;
        document.getElementById("boto2").onclick=function(){
                saludaPersonalitzat("Joan");
        };
        document.getElementBuId("boto3").onclick=function(){
                alert("Joan");
        };
}; //dins un script al head.
```

Si necessitam cridar una funció ja definida sense paràmetres posam el seu nom sense parèntesis.

Si necessitam cridar una funció ja definida amb paràmetres l'hem de situar dins d'una funció anònima.

Recapitulació

Utilitzau el *document.write* si voleu **generar contingut al document mentre aquest encara s'està carregant,** és a dir:

- El javascript s'executa directament, el codi es fora d'una funció.
- El javascript és dins una funció que és segur que es crida directament, sense la intervenció de l'usuari, abans de tancar-se l'etapa de representació de la pàgina.

Utilitzau *document.getElementById* si voleu **modificar el contingut del document quan aquest ja s'ha acabat de carregar**:

- El javascript s'executa com a resposta a un esdeveniment provocat per l'usuari, com per exemple:
  - Pitjar un botó
  - Canviar la selecció d'una llista
  - Canviar un radiobutton o un checkbox...
- El javascript s'executa en haver-se carregat la pàgina, amb l'esdeveniment *window.onload*.

Per mantenir el codi HTML lliure de javascript, millor utilitzar sempre *document.getElementById* i crear una funció que inicialitzi la pàgina cridada des de l'onload.