UT4. Introducció a Javascript

UT4. Introducció a Javascript

4.5 Estructures de control
4.6 Funcions

4.5. Estructures de control

A Javascript tenim tres tipus de sentències o instruccions:

- Seqüencials
- Selectives
- Iteratives

4.5. Estructures de control

Sentències Seqüencials

Podem descriure tres sentències seqüencials en Javascript:

- **Assignació:** L'assignació es realitza amb el símbol **=**, és a dir: "variable = expressió". Aquesta sentència, per si mateixa, no executa cap acció "visible" per a l'usuari de la nostra pàgina web, però és fonamental perquè els nostres programes puguin anar fent a poc a poc el seu treball, realitzant càlculs que es van emmagatzemant en les variables.

- **Escriptura:** Per simplificar una mica, la sentència d'escriptura podem considerar que és el "write". En realitat, hem de dir que és "**document.write**", ja que és un mètode d'un objecte que es diu **document** (que veurem més endavant). Com hem dit, aquest és un dels mecanismes que podem utilitzar per escriure coses des de JavaScript perquè es puguin veure a través del navegador.

4.5. Estructures de control

Sentències Seqüencials

Podem descriure tres sentències seqüencials en Javascript:

- **Lectura:** La sentència de lectura permet a l'usuari donar-li dades a el programa, o si es prefereix, permet el programa preguntar dades a l'usuari. En Javascript podem emprar l'objecte **window**, que té un mètode anomenat **prompt** que serveix per això.

4.5. Estructures de control

Sentències Condicionals

- **La sentència condicional**: L'if permet avaluar una condició i executar un grup d'instruccions si la condició resulta ser vertadera, i un altre grup d'instruccions, si la condició resulta ser falsa:

Sintaxi:

```
if ( condició ) {
    sentències_true;
}
else {
    sentències_false;
}
```

4.5. Estructures de control

Sentències Condicionals

- **La sentència condicional**: L'if permet avaluar una condició i executar un grup d'instruccions si la condició resulta ser vertadera, i un altre grup d'instruccions, si la condició resulta ser falsa:

Sintaxi:

```
if ( condició ) {
    sentències_true;
}
else {
    sentències_false;
}
```

4.5. Estructures de control

Sentències Condicionals

- **La sentència condicional switch (variable):** Permet comprovar per una mateixa variable distints valors. Segons el valor de la condició del "switch", s'avalua un grup de sentències la seqüència de valors. El darrer valor de la seqüència és "default" per si no trobam un valor coincident. El "break" és opcional, però si no es posa, s'executarien les sentències que es troben en els valors següents del valor coincident

Sintaxi:

```
switch (cond) {
  case valor1 : sentències; break;
  case valor2 : sentències; break;
  case valor3 : sentències; break;
  default: sentències; break;
}
```

Exemple:

```
var institut = prompt( "Indiqui a quin institut li agradaria cursar els seus estudis:
");
switch( institut ) {
  case "Francesc de Borja Moll" : document.write ("El millor" ); break;
  case "IEDIB" : document.write ("A distància" ); break;
  case "IES Na Camel.la" : document.write ("No té cicles d'informàtica" ); break;
  default:  document.write ( "L'institut que indica no existeix" ); break;
}
```

4.5. Estructures de control

Sentències Iteratives

- **Sentència iterativa while (condició):** Permet executar vàries vegades un tros de codi. Es coneix com a bucle.

Sintaxi:

```
while( condició ) {
  sentències;
}
```

Exemple: Un programa que escriu la taula de multiplicar del 15, des de 1 fins 50

```
document.write ( "<h2>Taula de multiplicar del 15</h2>" );
var cont=1;
while ( cont <= 50 ) {
  document.write ( "15 * " + cont + " = " );
  document.write ( 15*cont );
  document.write ( "<br>" );
  cont = cont+1;
}
```

4.5. Estructures de control

Sentències Condicionals

- **Sentència iterativa for():** Es repeteix el cicle fins que una condició especificada s'avalua com a falsa. (És similar al for de Java i C).

Sintaxi:

```
for ( inicialització; condició; increment ) {
   sentències;
}
```

Exemple: El mateix programa d'abans però amb la sentència for

```
for ( cont=1; cont<=50; cont=cont+1 ) {
 document.write ( "15 * " + cont + " = " );
 document.write ( 15*cont );
 document.write ( "<br>" );
}
```

4.5. Estructures de control

Sentències Condicionals

- **Sentència iterativa do while ():** Aquest bucle és exactament igual que l'anterior amb while però amb la diferència que la condició es comprova al final.

Sintaxi:

```
do {
   sentències;
}
while ( condició ):
```

Exemple:

```
var resultat = 1;
var numero = 5;
do {
        resultat = resultat * numero;
        numero--;
}
while (numero <= 50);
alert (resultat);
```

4.5. Estructures de control

Instruccions break i continue

- **break:** Server per aturar l'execució d'un bucle i sortir d'ell.

Exemple:

```
for (i=0;i<10;i++) {
   document.write (i);
   escriu = prompt ("Vols continuar?");
   if (escriu == "no");
  break;
}
```

4.5. Estructures de control

Instruccions break i continue

- **continue:** Serveix per aturar la iteració actual i tornar al principi del bucle per realitzar una altra iteració si correspon.

Exemple:

```
var i=0
while (i<7) {
   incrementar = prompt ("El valor de "i" és: " + i + ", vols continuar incrementar en un?", "si")
   if (incrementar == "no")
  continue;
   i++;
}
```

4.6. Funcions

Important:

- **Una funció JavaScript és un bloc de codi dissenyat per realitzar una tasca determinada.**
- **Una funció JavaScript s'executa quan "alguna cosa" la invoca (la crida).**
- **Les funcions permeten reutilitzar el codi.**
- **La definició d'una funció (també anomenada declaració de funció o sentència de funció) consisteix en emprar la paraula clau reservada: function.**
- **Les sentències JavaScript que defineixen la funció, van tancades per clau, { }.**
- **Les funcions s'han de carregar abans de l'execució del codi.**

4.6. Funcions

Sintaxi de la funció:

```
function nom (paràmetre1, paràmetre2, paràmetre3, ...) {
  // codi que s'ha d'executar, sentències o instruccions.
}
```

- Els paràmetres no sempre existeixen.
- Els paràmetres de funció es mostren dins dels parèntesis () a la definició de la funció.
- Els arguments de funció són els valors que rep la funció quan s'invoca.
- Dins de la funció, els arguments (els paràmetres) es comporten com a variables locals.
- Una funció és el mateix que un procediment o una subrutina, en altres llenguatges de programació.

Exemple:

```
function producte(p1, p2) {
  return p1 * p2;
}

alert(producte(3,4));
```

4.6. Funcions

Cridar les funcions

El codi dins de la funció s'executarà quan:

- Quan es produeix un esdeveniment (per exemple, quan un usuari fa clic a un botó).
- Quan s'invoca (es crida) des del propi codi JavaScript.
- Automàticament (auto-invocat).

Exemple:

```
var Cantar = function(){
  alert("Cantarem una cançó");
}

Cantar();
```

4.6. Funcions

Retorn de les funcions

- Les funcions que volem que retornin algun valor, inclouran un "return" a la darrera línia de codi de la funció.

Exemple:

```
function Valors(a,b,c){
  return a+b+c;
}

var suma = Valors(3,4,5); // Aquesta funció ens retorna la suma dels tres paràmetres.
```

4.6. Funcions

Retorn de les funcions

- Les funcions que volem que retornin algun valor, inclouran un "return" a la darrera línia de codi de la funció.

Exemple:

```
function Valors(a,b,c){
  return a+b+c;
}

var suma = Valors(3,4,5); // Aquesta funció ens retorna la suma dels tres paràmetres.
```

4.6. Funcions

Funcions predefinides de JavaScript

Javascript conté una gran quantitat de funcions en les seves llibreries. Moltes de les llibreries s'implementen a través d'objectes, per exemple l'objecte Math i l'objecte String.

Per exemple, l'objecte Math (la classe), s'utilitza per realitzar càlculs matemàtics. Per utilitzar-los s'opera a través de la classe, en lloc del objecte, per treballar amb la classe Math no s'utilitza la instrucció new, sinó que s'utilitza el nom de la classe per accedir a les seves propietats i mètodes.

Hi ha algunes funcions de JavaScript que no estan associades a cap objecte.

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions per a strings:

- **length()**: Calcula el nombre de caràcters d'una variable de tipus text.

  Exemple:

  ```
  var cadena = 'Hola';
  alert ('La variable té: ' + cadena.length + ' lletres');
  ```

  Resultat:

  ```
  La variable té: 4 lletres
  ```

- **concat()**: Realitza la concatenació de dues o més variables de tipus text. Hem de recordar que l'operador + té la mateixa funció.

  Exemple:

  ```
  var cadena1 = "Hola";
  var cadena2 = cadena1.concat(" com estàs?");
  alert (cadena2);
  ```

  Resultat:

  ```
  Hola com estàs?
  ```

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions per a strings:

- **toUpperCase()**: Transforma tots els caràcters d'una variable tipus text a majúscules, independentment de com estiguessin a l'inici.

  Exemple:

  ```
  var cadena = 'Text De Prova';
  cadena = cadena.toUpperCase();
  alert(cadena);
  ```

  Resultat:

  ```
  TEXT DE PROVA
  ```

- **toLowerCase()**: És la funció contrària a l'anterior. Transforma els caràcters d'una variable tipus text a minúscules, independentment de com estiguessin a l'inici.

- **charAt(posició)**: Retorna el caràcter contingut dins de la variable de text que indiqui la posició.

  Exemple:

  ```
  var cadena = "Hola";
  var lletra = cadena.charAt(3); // lletra = a
  console.log(lletra)
  ```

  Resultat:

  ```
  a
  ```

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions per a strings:

- **indexOf(caràcter)**: Retorna un número que correspon a la posició en què es troba el caràcter indicat dins dels parèntesis. Si el caràcter s'inclou diverses vegades, indicarà només la seva primera posició començant des de l'esquerra. Si el caràcter indicat no es troba en cap posició, la funció torna el valor -1.

  Exemple:

  ```
  var cadena = 'Text De Prova';
  cadena = cadena.toUpperCase();
  alert(cadena);
  ```

- **lastIndexOf(caràcter)**: Cerca la darrera posició on es troba el caràcter. Si la cadena no conté el caràcter, la funció torna el valor -1.

  Exemple:

  ```
  var cadena = "Hola, som estudiant de FP Informàtica, a FP de grau superior";
  var n = cadena.lastIndexOf("FP");
  ```

  Resultat:

  ```
  n=39
  ```

- **substring(inici, final)**: Extreu els caràcters d'una cadena, entre els dos indexes especificats.

  Exemple:

  ```
  cadena = "Bona nit"
  var cadena = cadena.substring(1, 6);
  ```

  Resultat:

  ```
  ona ni
  ```

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions per a strings:

- **split(separador)**: Retorna un array que contindrà cadenes de text que seran subcadenes de la variable original, separades pel caràcter separador que hagem indicat.

  Exemple:

  ```
  var cadena = "Hola com estàs!";
  var paraules = cadena.split(""); // paraules = ["Hola", "com", "estàs!"];
  ```

- **replace(string1, string2):** Cerca i reemplaça un string per un altre (treballa amb expressions regulars).

  Exemple:

  ```
  var frase = "Tenc tres mòduls aprovats";
  frase = frase.replace("tres","dos");
  alert(frase);
  ```

  Resultat:

  ```
  Tenc dos mòduls aprovats
  ```

  Per a poder realitzar substitucions més complexes, podem utilitzar expressions regulars.

  Exemple:

  ```
  cadena = cadena.replace(/expressió regular/gi,'caràcter nou');
  ```

  On:

  ```
  g: Indica que se aplicarà al llarg de tota la cadena.
  i: Indica que no es sensible a majúscules i minúscules.
  ```

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions per a strings:

- **search(text):** Cerca un text dins d'un string, si el troba retorna l'índex, si no retorna -1 (pot treballar amb expressions regulars).

  Exemple:

  ```
  cadena.search('n'); // On cadena conté per exemple "Això és una prova"
  ```

  Resultat:

  ```
  10
  ```

- **trim():** Elimina els espais inicials i finals d'un string.

  Exemple:

  ```
  var cadena = '   superman ';
  alert(cadena);
  ```

  Resultat:

  ```
  superman
  ```

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions per a arrays:

- **join(separador):** És la funció contrària a split(). Retorna una variable de text que és el resultat d'unir tots els elements d'un array. pop(): Retorna l'últim element d'un array. L'array original modifica la longitud en un element.

  Exemple:

  ```
  var array1 = ["Bon", "Dia"];
  var cadena = array1.join(""); // cadena = "BonDia"
  cadena = array1.join(" ");   // cadena = "Bon Dia"
  ```

- **push()**: Afegeix un element al final de l'array. L'array original es modifica i augmenta la seva longitud en 1 element. (També és possible afegir més d'un element a la vegada).

  Exemple:

  ```
  var array1 = [1, 2, 3, 4];
  array1.push(5); // array1 = [1, 2, 3, 4, 5]
  ```

- **shift()**: Retorna el primer element de l'array. La longitud de l'array disminueix amb un element.

  Exemple:

  ```
  var colors = [“grog”, “verd”, “vermell”];
  colors.shift();
  ```

  Resultat:

  ```
  grog // i l’array colors=[“verd”,”vermell”]
  ```

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions per a arrays:

- **unshift(valors_a_afegir)**: Afegeix un o més elements al començament d'un array. La seva longitud augmentarà en un element.

  Exemple:

```
let array = [1, 2, 3, 4];
alert(array.unshift(-1, 0));// Retorna 6
alert(array);// Retorna array [-1, 0, 1, 2, 3, 4]
```

- **reverse()**: Retorna l'array amb tots els seus elements col·locats en ordre invers a la seva posició original.

  Exemple:

```
let colors = ['verd', 'vermell', 'blanc'];
alert(colors.reverse()); //color= ["blanc", "vermell", "verd"]
```

- **pop()**: Elimina l'últim element de l'array i el torna. L'array original es modifica i la seva longitud disminueix en 1 element.

  Exemple:

```
var array = [1, 2, 3, 4];
var num = array.pop(); // array = [1, 2, 3], num = 4
```

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions per a arrays:

- **slice(index1, index2):** Extreu una secció de l'array.

  Exemple:

  ```
  fruites =  ["meló", "mandarina", "poma", "cirera"]
  fruites.slice(0,2); // Sempre agafa index1 i no agafa index2. Si son iguals és buit.
  ```

  Resultat:

  ```
  Retorna: ["meló", "poma"]
  ```

- **splice(index, num, elements):** Afegeix/elimina elements d'un array, i retorna els elements eliminats. Té els següents elements:
  - index: Un sencer que especifica en quina posició afegeix/elimina elements.
  - num (opcional): El número d'elements que han de ser eliminats. Si es posa 0, no s'eliminaran elements.
  - elements: Els nous elements que s'afegiran a l'array.

  Exemple:

  ```
  let colors = ["Blanc", "Verd", "Grog", "Marró", "Ocre"];
  colors.splice(2, 1, "Vermell", "Blau"); // A la posició 2, afegeix els nous elements, i n'esborra 1 (Grog)
  alert(colors) // [ 'Blanc', 'Verd', 'Vermell', "Blau", 'Marró', 'Ocre' ]
  ```

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions per a arrays:

- **sort():** Ordena alfabèticament els elements.
  - Per ordenar array amb valors de tipus string, no hi ha cap problema:

    Exemple:

    ```
    let plantes = ['ametller', 'roser', 'palmera', 'garrover'];
    plantes.sort();
    ```

    Resultat:

    ```
    ['ametller', 'garrover', 'palmera', 'roser'];
    ```

  - En el cas d'ordenació de números, té el següent problema: Si volem ordenar els números 1, 100, 98, 56, no ens donaria 1, 56, 98, 100, si no 1, 100, 56, 98. Per evitar que passi, podrem utilitzar una funció de comparació i passar-la per paràmetre a sort. La funció serà: function(a, b){return a - b}. Aquesta funció el que fa, es retornar a-b, o sigui un valor negatiu, positiu o cero:

    - Si el resultat és negatiu, a s'ordena abans que b.
    - Si el resultat és positiu, b s'ordena abans que a.
    - Si el resultat és 0, nada canvia.

    Exemple:

    ```
    let numeros = [3, 23, 12];
    numeros.sort(function(a, b){return a - b}); // --> 3, 12, 23
    ```

    Resultat:

    ```
    [3, 12, 23];
    ```

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions numèriques:

- **IsNaN()**: Retorna si l'expressió és o no un número. Permet protegir a l'aplicació de possibles valors numèrics no definits.

  Exemple:

```
var numero1 = prompt("Introdueix el primer número);
var numero2 = prompt("Introdueix el segon número");
if (isNaN(numero1/numero2)) {
  alert ("La divisió donarà error");
}
else {
  alert ("La divisió es igual a => " + numero1/numero2);
}
```

- **toFixed(dígits)**: Retorna el número amb tants decimals com indiquem en el paràmetre dígits.

  Exemple:

```
var num = 25.1267877;
var n = num.toFixed(2);
```

  Resultat:

```
n=25.13;
```

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions numèriques:

- **ParseInt i ParseFloat:** Són dues funcions globals que ens permeten convertir cadenes de text en valors numèrics si és possible. Els espais inicials s'ignoren. El seu funcionament és: s'analitza la cadena per extreure les xifres que es troben a l'inici. Aquestes xifres inicials, són les que es transformen a tipus numèric, quan es troba un caràcter no numèric, s'ignora la resta de la cadena. Si el primer caràcter trobat no és convertible a número, el resultat és **NaN** (Not a Number).

  Exemple:

```
parseInt("10");       // 10
parseInt("10.8");     // 10
parseInt("10 anys");       // 10
parseInt("Tenc 18 anys"); // NaN

parseFloat("3.14");   // 3.14
parseFloat("314e-2"); // 3.14
parseFloat("3.14 número pi"); // 3.14
parseFloat("el numero pi és 3.14");  // NaN
```

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions matemàtiques:

- **pi**: Nombre PI.
- **sin(x), cos(x), tan(x)**: Sinus, cosinus i tangent.
- **asin(x), acos(x), atan(x)**: Arcosinus, arcocosinus i arcotangent.
- **max([x[,y[,…]]]), min([x[,y[,…]]])**: Màxim i mínim d'una llista de números.
- **sqrt(x)**: Arrel quadrada.
- **round(x)**: Arrodoniment a sencer.
- **random()**: Genera un número aleatori entre [0 i 1].

4.6. Funcions

A continuació es mostren algunes de les funcions predefinides de JavaScript, però n'hi ha més:

Funcions de dates:

- **getDay(), getDate(), getMonth(), getFullYear(**): Retornen el dia de la setmana, del mes, i l'any, en hora local respectivament.

  Exemple:

```
var data = new Date();
var dia = data.getDate();
var mes = data.getMonth();
var any = data.getFullYear();
```

- **getHours(), getMinutes(), getSeconds(), getMilliseconds()**: Retornen les hores, minuts, segons i milisegons.
- **setDay(dia), setDate(dia), setMonth(mes), setFullYear(any)**: Estableixen el dia de la setmana, del mes, i del any.
- **setHours(hora), setMinutes(minut), setSeconds(segon), setMilliseconds(milisegon)**: Estableixen les hores, minuts, segons i milisegons respectivament.
- **toString(), toDateString(), toTimeString()**: Retornen la data i hora.
- **toLocaleString(), toLocaleDateString(), toLocaleTimeString()**: Retornen la data i hora amb format de sistema.

Referència: https://www.w3schools.com/js/default.asp i https://www.w3schools.com/jsref/default.asp