Objectes

Object () Constructor :

JavaScript proporciona un constructor predefinit especial anomenat Object (), que s'utilitza amb la paraula clau new per crear un objecte genèric.

```
<html>
  <head><title>The Object() Constructor</title>
    <script type = "text/javascript">
1     var cat = new Object();
2     alert(cat);
    </script>
  </head>
  <body></body>
</html>
```

Propietats del objecte : Les propietats descriure l'objecte. El nom de l'objecte és seguit per un punt i una propietat. Les propietats són accessibles només a través del seu objecte

```
<html>
  <head><title>User-defined objects</title>
1   <script type = "text/javascript">
2     var toy = new Object(); // Create an instance of the object
3     toy.name = "Lego";      // Assign properties to the object
      toy.color = "red";
      toy.shape = "rectangle";
4   </script>
  </head>
  <body bgcolor="lightblue">
```

```
5   <script type = "text/javascript">
6     document.write("<b>The toy is  a " + toy.name + ".");
7     document.write("<br />It is a " + toy.color + " "
                     + toy.shape+ ".");
8   </script>
  </body>
</html>
```

Mètodes d'un objecte : Els mètodes són funcions especials que els llenguatges orientats a objectes utilitzen per descriure el comportament del objecte . Els mètodes només són accessibles a través d'un objecte.

```
<html>
  <head><title>User-defined objects</title>
    <script type= "text/javascript">
1     var toy = new Object();    // Create the object
2     toy.name = "Lego";       // Assign properties to the object
      toy.color = "red";
      toy.shape = "rectangle";
3     toy.display=printObject; // Function name is assigned as a
                               // property of the object
4     function printObject(){
        document.write("<b>The toy is a " + toy.name + ".<br>");
```

```
        document.write("It is a " + toy.color + " " + toy.shape
                       + ".<br />");
      }
    </script>
  </head>
  <body>
    <script type = "text/javascript">
5     toy.display();  //Object method is called
6     toy.color="blue";
7     toy.display();
    </script>
  </body>
</html>
```

4 : Associem la funció printObject() a la propietat display del objecte toy

Classes : Una classe amb JavaScript es defineix mitjançant la creació d'una funció . El nom de la funció servirà com el nom de la classe d'un objecte, i la funció definirà les seves propietats i mètodes

```
<script type="text/javascript">
1   function Book(){  // Create a Book class
2     this.title = "The White Tiger";  // Properties
      this.author = "Aravind Adiga";
    }
3   var bookObj = new Book; // Create new Book object
    alert(bookObj.title + " by " + bookObj.author);
</script>
```

```
<html>
  <head><title>User-defined objects</title>
    <script type ="text/javascript">
1     function Book(title, author, publisher){   // Receiving
                                                 // parameters
2       this.pagenumber=0;      // Properties
        this.title = title;
        this.author = author;
        this.publisher = publisher;
3       this.uppage = pageForward;   // Create methods
4       this.backpage = pageBackward;
      }
```

```
5     function pageForward(){   // Functions to be used as methods
        this.pagenumber++;
        return this.pagenumber;
      }
6     function pageBackward(){
        this.pagenumber--;
        return this.pagenumber;
      }
    </script>
  </head>
  <body>
    <script type = "text/javascript">
7     var myBook = new Book("                    ',
        "              ", "              " );   // Create new object
8     myBook.pagenumber=5;  //Assign a page number
9     document.write( "<b>"+ myBook.title +
                      "<br>" + myBook.author +
                      "<br>" + myBook.publisher +
                      "<br>Current page is " + myBook.pagenumber );
      document.write("<br>Page forward: " );
10    for(i=0;i<3;i++){
11      document.write("<br />" + myBook.uppage());
        // Move forward a page
      }
      document.write("<br />Page backward: ");
      for(;i>0; i--){
12      document.write("<br />" + myBook.backpage());
        // Move back a page
      }
    </script>
  </body>
```

Funcions en línia com a mètodes : En lloc de nomenar una funció fora de la classe, una funció en línia o anònima pot assignar directament a una propietat dins de d'una funció constructora

```
<html>
  <head><title>functions</title>
    <script type="text/javascript">
1     function Distance(r, t){ //Constructor function
2       this.rate = r;
        this.time = t;
3       this.calculate=function() { return r * t; } // anonymous
      }
    </script>
  </head>
  <body>
    <script type ="text/javascript">
4     var trip1 = new Distance(50, 1.5);
5     var trip2 = new Distance(75, 3.2);
6     alert("trip 1 distance: "+ trip1.calculate());
      alert("trip 2 distance: "+ trip2.calculate());
    </script>
  </body>
</html>
```

3 : Assignem una funció anònima com a propietat calculate del objecte Distance

Objectes literals : Els objectes literals permeten crear objectes que admeten moltes característiques sense directament nvocar una funció constructora. Aquest objectes literals són similars als hashes en altres llenguatges utilitzant parells clau / valor per representar les propietats i els mètodes del objecte .

```
var object = { property1: value, property2: value };
```

```
<html>
  <head><title>working with literal objects</title>
    <script type="text/javascript">
1     var soldier = {
2       name: undefined,
3       rank: "captain",
4       picture: "keeweeboy.jpg",
5       fallIn: function() {
          alert("At attention, arms at the side, head and
                 eyes forward.");
        },
6       fallOut: function() {
          alert("Drop out of formation, step back, about face!");
        }
      };
    </script>
  </head>
  <body>
    <big>
    <script type="text/javascript">
7     soldier.name="Tina Savage";  // Assign value to object
                                   // property
8     document.write("The soldier's name is ",
                     soldier.name,".<br />");
      document.write(soldier.name+"'s rank is ",
                     soldier.rank+."<br />");
9     document.write("<img src='"+soldier.picture+"'>")
10    soldier.fallIn();  //call object's methods
11    soldier.fallOut();
    </script>
    </big>
  </body>
</html>
```

F

Bucle for/in

```
for(var property_name in object){
   statements;
}
```

```
<html>
  <head><title>User-defined objects</title>
    <script type = "text/javascript">
1       function book(title, author, publisher){
2          this.title = title;
           this.author = author;
           this.publisher = publisher;
3          this.show=show;   // Define a method for the object
        }
```

```
4       function show(obj, name){
        // Function to show the object's properties
          var result = "";
5         for (var prop in obj){
6           result += name + "." + prop + " = " +
                  obj[prop] + "<br />";
          }
7         return result;
        }
    </script>
  </head>
  <body bgcolor="lightblue">
    <script type="text/javascript">
8       myBook = new book("                    ",        ",
                          "              ");
9       document.write("<br /><b>" + myBook.show(myBook, "myBook"));
    </script>
  </body>
</html>
```

Prototipus : Les funcions amb Javascript se'ls assigna automàticament un objecte prototip buit.
Si la funció actua és una funció constructora per a una classe d'objectes, l'objecte prototip es pot utilitzar per estendre la classe.

```
    </html>
      <head><title>Object Properties</title>
1       <script type="text/javascript">
          function Book(title, author){
            this.title =title;
            this.author=author;
          }
        </script>
      </head>
      <body bgColor="#EOFFFF">
        <big>
        <script>
2         var book1 = new Book("Kidnapped","R.L.Stevenson");
          var book2 = new Book("Tale of Two Cities", "Charles Dickens")
3         book1.publisher="Penguin Books";
          document.write(book1.title + " is published by "+
4         book1.publisher + "<br />");
          document.write(book2.title + " is published by " +
5         book2.publisher);  //Doesn't have this property
        </script>
        </big>
      </body>
    </html>
```

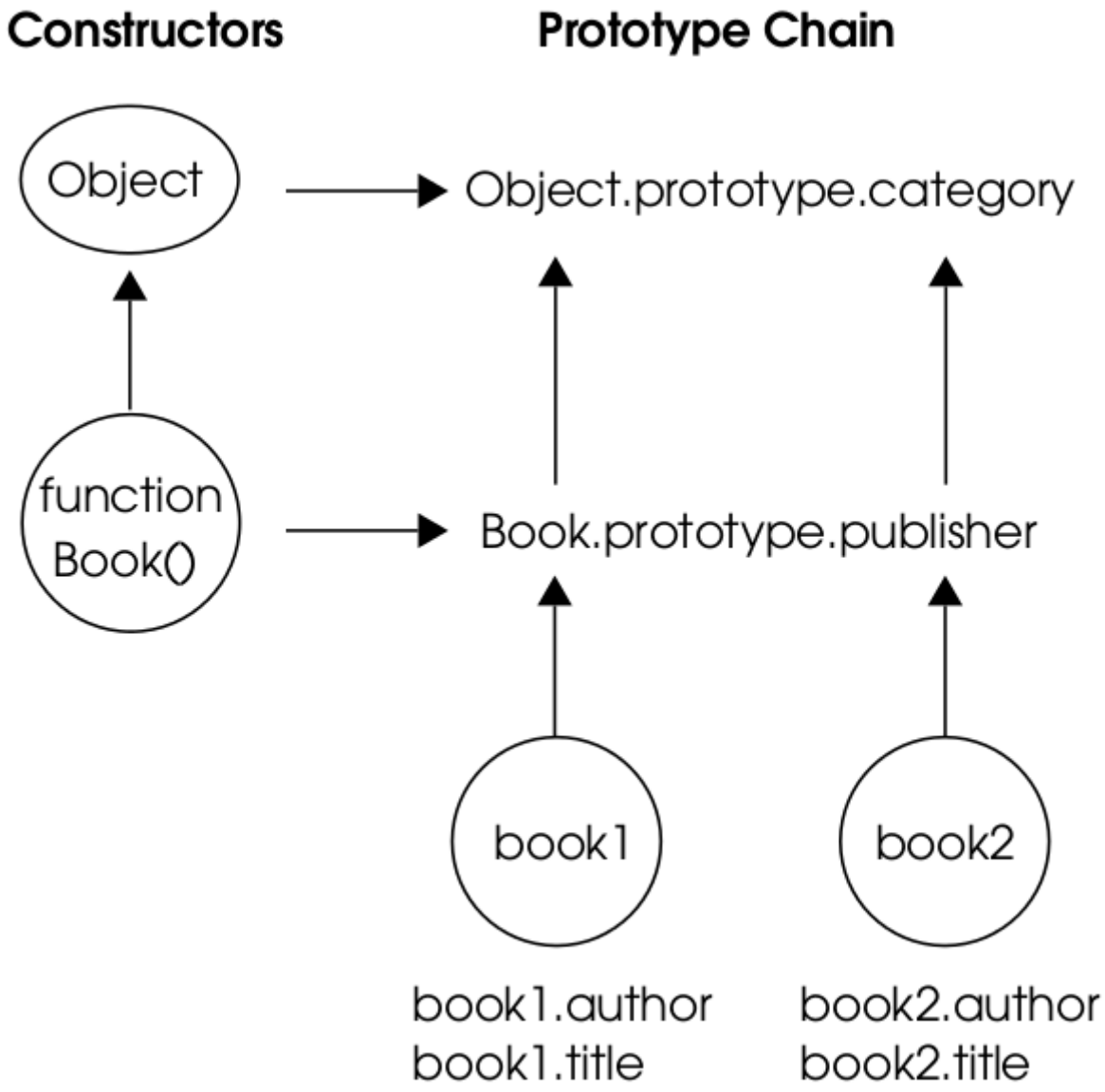
3 : Es crea una nova propietat anomenada publisher en l'objecte book1
4 : La propietat publisher no està inclosa en l'objecte book2

```
<html>
  <head><title>Object Properties</title>
    <script type="text/javascript">
1     function Book(title, author){
        this.title =title;
        this.author=author;
      }
    </script>
  </head>
  <body bgColor="#EOFFFF">
    <big>
    <script>
2     var book1 = new Book("Kidnapped","R.L.Stevenson");
      var book2 = new Book("Tale of Two Cities", "Charles Dickens")
3     Book.prototype.publisher = "Penguin Books";

      document.write(book1.title + " is published by "+
4          book1.publisher + "<br />");
      document.write(book2.title + " is published by " +
          book2.publisher);
    </script>
    </big>
  </body>
</html>
```

3 : La propietat publisher s'ha inclòs en el prototipus Book fent que totes les instàncies a aquest objecte continguin aquesta propietat. Per tant tant book1 com book2 contenen aquesta.

Recerca d'un prototipus : Quan Javascript intenta recuperar la propietat d'un objecte, primer busca si la propietat ha estat definida directament per aquest objecte.
Si és així, JavaScript recupera la propietat. Si la propietat no s'ha definit en el propi objecte, llavors JavaScript busca la propietat l'objecte prototipus. Si no es defineix allà, JavaScript el busca en l'objecte pare d'aquest utilitzant la herència



```
<html>
  <head><title>Object Properties</title>
    <script type="text/javascript">
1     function Book(title, author){
        this.title =title;
        this.author=author;
      }
    </script>
  </head>
  <body bgColor="#EOFFFF">
    <big>
    <script>
2     var book1 = new Book("Kidnapped","R.L.Stevenson");
      var book2 = new Book("Tale of Two Cities", "Charles Dickens")
3     Object.prototype.category="Fiction";
4     Book.prototype.publisher = "Penguin Books";
      document.write(book1.title + " is published by "+
5       book1.publisher + " and is in the " + book1.category +
        " category <br />");
      document.write(book2.title + " is published by "+
        book2.publisher + " and is in the " + book1.category +
        " category <br />"); </script>
    </script>
    </big>
  </body>
</html>
```

3 : S'ha creat una propietat anomenada category en l'objecte Object. Això implica que TOTS els objectes instanciats tindran aquesta propietat.
4: La propietat publisher es creada en l'objecte prototipus fent que totes les instàncies al objecte Book tinguin aquesta propietat

EXERCICIS

- Crear un objecte anomenat cercle amb el constructor Object() i un mètode que calculi la serva circunferència
- Crear una funció que crei l'objecte rellotge
  - Tindrà 3 propietats : segons,minuts i hores
  - Escriure dos mètodes un que es pugui canviar l'hora i l'altre que la visualitzi
  - En el mètode que visualitzi l'hora l'usuari podrà especificar 12 a.m 12 p.m o bé 0-24

- Donada la següent funció

```
<script type="text/javascript">
    function Treballador(){
      var = non;
     this.donarNom = function(){
    return this.nom;
    }
    this.posarNom = function(name){
    this.nom = nom;
   };
  }
```

Crear 3 nous treballadors i imprimir els seus noms. Afegir una nova propietat anomenada “telèfon”

- Crear un objecte literal anomenat Client. Aquest ha de tindre les propietats nom,sexe,foto i ocupació. Hi ha d'haver un mètode anomenat MostrarClient() on es visualitzaran totes les seves propietats. Crear una altre mètode anomenat MostraClientTaula() on es mostrin les seves propietats en una taula de HTML
- Utilitzar el prototipus del exercici de objecte Treballador per crear les propietats DonarSalari i PosarSalari i crear un mètode PujarSalari() que permeti elevar un 15 el salari d'un treballador.