JavaScript အားအသုံးပြုခြင်း

Continue statement in Javascript

Continue statement ရဲ့ အလုပ်လုပ်ပုံ သဘောသဘာဝကတော့ သူနဲ့ ဖမ်းထားတဲ့ statement ကို skip, block လုပ်ပါတရ်။ များသောအားဖြင့် looping တွေမှာ ထည့်သွင်းအသုံးပြုပြီး continue ကိုတွေ့လိုက်တာနဲ့ သူနဲ့ဖမ်းထားသော statement ကို block လုပ်ကာ looping ကိုတော့ ပြီးဆုံးသည် အထိ ဆက်လက်အလုပ်လုပ်သွားပါလိမ့်မရ်။ သဘောတရားကို နားလည်ပြီဆိုရင်တော့ ကျနော်တို့ လေ့ကျင့်ခန်းလေး တခုလောက် လေ့လာကြည့်ကြအုံးစို့ဗျာ။

```
<html>
<head><title>Continue Statement</title></head>
<body>
<script language="javascript">
for(var i=0; i<=5; i++)
{
      if(i == 2)
        continue;
       document.write("i is - " +i+ "<br>");
}
</script>
</body>
</html>
```

Result:

i is - 0
i is - 1
i is - 3
i is - 4
i is - 5

အလုပ်လုပ်သွားပုံကို နားလည်မရ်ထင်ပါတရ်။ ကျနော်တို့ for loop အရ I တန်ဖိုးဟာ 0 to 5 ထိပြန်ထုတ်ပေးရမှာဖြစ်ပါတရ်။ ဒါကိုမှ ကျနော်တို့က if statement နဲ့ i==2 ဖြစ်တဲ့အခြေအနေမှာ continue ဆိုတာနဲ့ ဖမ်းထားတဲ့အတွက်ကြောင့် result ကိုကြည့်လိုက်ပါ။ i=2 ဖြစ်တဲ့ statement ကိုကျော်သွားပြီး 0,1,3,4,5 ဆိုပြီးတော့ ပြန်လည်ထုတ်ပြတာပါ။ နားလည်မရ် ထင်ပါတရ်။ Continue အလုပ်လုပ်ပုံကို ပိုမို ရှင်းလင်းစွာသိသွားရအောင် ကျနော်တို့ နောက် လေ့ကျင့်ခန်းတခု ဆက်လေ့လာ ကြည့်ကြအုံးစို့ဗျာ။

```
<html>
<head><title>Continue Statement</title></head>
<body>
<script language="JavaScript">
var x = 0;
```

```
while (x < 20)
{
   x++;
   if (x % 2 == 0)
   {
      continue;
   }
   document.write(x, ",    ");
}
</script>
</body>
</html>
```

Result:

```
1, 3, 5, 7, 9, 11, 13, 15, 17, 19,
```

ရိုးရိုးရှင်းရှင်းပါပဲ။ ကျနော်တို့က 0 to 20 ဆိုတဲ့ series ထဲကမှာ မကိန်းတွေကြီးကို ပြန်ထုတ်ပြစေချင်တဲ့ အတွက်ကြောင့် while(x<20) loop ထဲမှာမှာ if statement နဲ့ x%2==0 ဖြစ်ခဲ့လျှင် continue ဆိုပြီး ဖမ်းထားပါတရ်။ ဒီတော့ x ကို ၂ နဲ့စားပြီး အကြွင်း သုည ရတရ်ဆိုမှတော့ စုံကိန်းတွေပေါ့။ Continue ရဲ့ သဘောသဘာဝက သူနဲ့ဖမ်းထားတဲ့ statement အားလုံးကို skip လုပ်တရ်ဆိုတော့ ရိုးရိုးရှင်းရှင်းပါပဲ။ ကျနော်တို့ 0 to 20 series ထဲက စုံကိန်းတွေဖယ်ထုတ်ပြီး မကိန်းတွေကြီး နောက်ဆုံး ကျနော်တို့ကို ပြန်လည် ထုတ်ပြတာပါ။ Continue statement ကို အသုံးပြုတတ်မရ်လို့ ယူဆပါတရ်။ နောက်လေ့ကျင့်ခန်းတခု ဆက်ကြအုံးစို့ဗျာ။

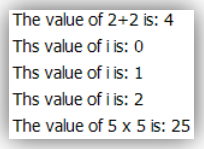
Label statement in Javascript

Label statement ကိုတော့ break, continue စတဲ့ statement တွေနဲ့ တွဲဖတ်အလုပ်လုပ်ခိုင်းနိုင်ပါတရ်။ သူ့ရဲ့ အလုပ်လုပ်ပုံကတော့ statement အုပ်စုတွေကို ကိုယ်စားပြု label name တခု သတ်မှတ်ပေးထားပြီး ထို statement အုပ်စုကြီး တခုလုံး ဆက် အလုပ်လုပ်မရ်၊ မလုပ်ဘူးဆိုတာကို ၎င်း label name နဲ့ break, continue statement တွေနဲ့ တွဲဖတ်ကာ အသုံးပြုနိုင်တာဖြစ်ပါတရ်။ သဘောတရားကို နားလည်ပြီ ဆိုရင်တော့ ကျနော်တို့ လေ့ကျင့်ခန်းလေးတွေ လေ့လာကြည့်ကြအုံးစို့။

```
<html>
<head><title>Label Statement</title></head>
<body>
<script language="JavaScript">
  LABEL_A:
       var x = 2+2;
       var y = 5*5;
       document.write("The value of 2+2 is: " + x + "<br>");
  LABEL_B:
       for(i=0; i<5; i++)
          {
             document.write("Ths value of i is: " + i + "<br>");
```

```
            if(i==2)
                {
                   break  LABEL_B;
                }
         }
       document.write("The value of 5 x 5 is: " + y + "<br>");
</script>
</body>
</html>
```

Result:



သာမန် Break statement အသုံးပြုပုံနဲ့ သဘောတရားမကွာခြားပါဘူး။ ဒီပုစ္ဆာလေးကတော့ အုပ်စုလိုက်လေးတွေ Label တတ်ထားပြီး အုပ်စုတခုလုံးရဲ့ လုပ်ပိုင်ခွင့်ကို ပိတ်ပင်လိုက်တဲ့ သဘောပါပဲ။ နောက်ဥပမာလေး တခု ဆက်လေ့လာကြည့်ကြအုံးစို့ဗျာ။

```
<html>
<head><title>Label Statement</title></head>
<body>
<script type="text/javascript">
document.write("Entering the loop!<br /> ");
outerloop:                              // This is OuterLoop label name
for (var i = 0; i<=10; i++)
{
  document.write("OUTERLOOP: " + i + "<br />");
  innerloop:                            // This is InnerLoop label name
  for (var j = 0; j<=5; j++)
  {
    if (j >  3 ) break ;                // Quit the innermost loop
    if (i == 2) break innerloop;        // Quit the inner loop (or) Don't word innerloop
    if (i == 4) break outerloop;        // Quit the outer loop (or) Don't word Outerloop
    document.write("--------- Innerloop: " + j + "  <br />");
  }
}
document.write("Exiting the loop!<br /> ");
</script>
</body>
</html>
```

Result:

```
Entering the loop!
OUTERLOOP: 0
--------- Innerloop: 0
--------- Innerloop: 1
--------- Innerloop: 2
--------- Innerloop: 3
OUTERLOOP: 1
--------- Innerloop: 0
--------- Innerloop: 1
--------- Innerloop: 2
--------- Innerloop: 3
OUTERLOOP: 2
OUTERLOOP: 3
--------- Innerloop: 0
--------- Innerloop: 1
--------- Innerloop: 2
--------- Innerloop: 3
OUTERLOOP: 4
Exiting the loop!
```

အလုပ်လုပ်သွားပုံကိုကြည့်လိုက်ပါ။ ကျနော်တို့ for loop နှစ်ခု အသုံးပြုပြီး ပထမ for loop ကို OuterLoop, ဒုတိယ for loop ကို InnerLoop အစရှိတဲ့ label name တွေပေးလိုက်ပါတရ်။ ဒါကိုမှ ကျနော်တို့က if statement တွေ အသုံးပြုပြီး Lable နှစ်ခုရဲ့ လုပ်ပိုင်ခွင့်တွေကို ကန့်သတ်ထားတာပါ။ For loop နှစ်ခုအရ i တန်ဖိုးက i<=10 အခြေအနေထိ အလုပ်လုပ်ကို လုပ်ရမှာပါ။ ဒီလိုပါပဲ j တန်ဖိုးကလဲ j<=5 အခြေအနေထိ အလုပ်လုပ်ရပါလိမ့်မရ်။

ပထမ if statement ကိုကြည့်လိုက်ပါ။ if(j<3) လို့ စစ်ထားတဲ့အတွက် innerloop 0 to 5 လုပ်ငန်းစဉ်မှာ j တန်ဖိုးက 3 ရောက်တဲ့ အချိန်တိုင်းမှာ ရပ်တန့်သွားပြီး 5 ထိ ဆက်အလုပ် မလုပ်တော့တာပါ။

ဒုတိယ if statement မှာဆက်ကြည့်ပါ။ if(i==2) berak innerloop; ဆိုပြီး စစ်ထားပါတရ်။ ဒါကြောင့် I တန်ဖိုးက 2 ရောက်တဲ့ အခြေအနေမှာ innerloop ဆိုတဲ့ label ကြီးတခုလုံးကို အလုပ်မလုပ်စေပဲ block လုပ်သွားတာတွေ့ရပါလိမ့်မရ်။ ဒီအခြေအနေမှာ Innerloop label ကြီးကိုသာ ရပ်တန့်ပစ်လိုက်ပေမဲ့ outerloop label ကတော့ ဆက်အလုပ် လုပ်နေအုံးမှာပါ။ ဒါကြောင့် Outerloop 3 နေရာမှာ innerloop 0 to 3 ဆက်အလုပ်သွားတာပါ။

တတိယ if statement ကို ဆက်လေ့လာကြည့်လိုက်ပါအုံး if(i==4) break outerloop; ဆိုပြီး စစ်ထားပါတရ်။ ဒါကြောင့် I တန်ဖိုး 4 ဖြစ်သွားတဲ့ အချိန်မှာ outerloop label ကြီးတခုလုံးရဲ့ အလုပ်ကို ရပ်တန်စေပြီး block လုပ်က program က ထွက်သွားတာပါ။ ဒီမှာ outerloop label ကို skip လုပ်လိုက်ယုံနဲ့ ဘာကြောင့် innerloop label ကြီးပါ ရပ်တန့်သွားလဲ ဆိုရင်တော့ innerloop label ဟာ outerloop lable {...}block အတွင်းမှာ ရှိနေလို့ပါပဲ။ ဒီလောက်ဆိုရင်ဖြစ် lable အသုံးပြုမှုကို အနည်းငယ် နားလည်ပြီလို့ ထင်ပါတရ်။ မိတ်ဆွေအပေါင်း လေ့လာခြင်းဖြင့် ကျေနပ်နိုင်ကြပါစေဗျာ။

Try...Catch Statement and Throw Statement in Javascript

Throw statement နဲ့ Try...Catch statement နှစ်ခုကိုတော့ များသောအားဖြင့် တွဲဖတ်အသုံးပြုကြပါတယ်။ Throw ကိုတော့ မိမိ သွားချင်တော့ statement တခုကို ခုန်သွားချင်တဲ့ အခါမျိုးမှာ အသုံးပြုပြီးတော့ Try... catch statement ကတော့ error code တွေနဲ့ ဖမ်းပြီးအသုံးပြုချင်တဲ့ အခါမျိုးမှာ အသုံးပြုပါတယ်။ Try...catch statement ရဲ့ syntax ပုံစံလေးကတော့....

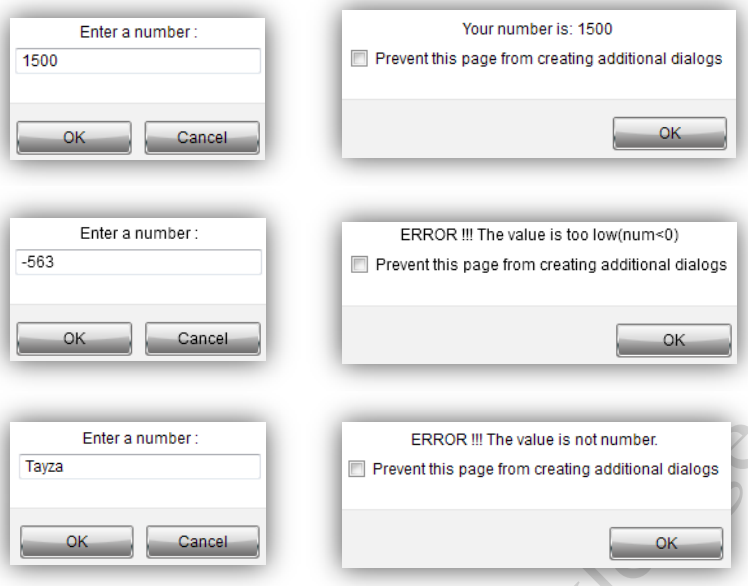
```
Try
{
   Run code here;
}
Catch(...)
{
   Error code here;
}
```

Syntax လေးကို နားလည်ပြီ ဆိုရင်တော့ ကျနော်တို့ လေ့ကျင့်ခန်းလေး တခုလောက် လေ့လာကြည့်ကြရအောင်ဗျာ။

```
<html>
<head><title>Throw and Try...Catch statements</title></head>
<body>
<script type="text/javascript">
var num=prompt("Enter a number :");
try
{
  if(num<0)
     {
       throw "error1";
     }
  else if(isNaN(num))
    {
      throw "error2";
    }
  else
   {
    alert("Your number is: "+num);
   }
}
catch(er)
{
  if(er=="error1")
   {
     alert("ERROR !!! The value is too low(num<0)");
   }
  if(er=="error2")
   {
     alert("ERROR !!! The value is not number.");
```

```
    }
}
</script>
</body>
</html>
```

Result:



Program ရဲ့ လုပ်ငန်းစဉ်က ရိုးရှင်းပါတယ်။ Run လိုက်လိုက်တယ်ဆိုရင်ပဲ prompt box နဲ့ ကျနော်တို့ကို element တခုခု ရိုက်ထည့်ခိုင်းပါလိမ့်မယ်။ ကျနော်တို့က သုညထပ်ကြီးတဲ့ ကိန်းရိုက်မိရင် ထိုကိန်းကို ပြန်ပြပေးတဲ့ message ထုတ်ပြပေးပြီး သုညအောက် ငယ်တဲ့ကိန်း ရိုက်မိရင်တော့ ကျနော်တို့ catch မှာ ဖမ်းထားတဲ့ error message ကို ထုတ်ပြပေးပါလိမ့်မယ်။ ဒီလိုပါပဲ ကျနော်တို့က ကိန်းဂဏန်းတွေ မဟုတ်တဲ့ string တွေရိုက်ထည့်မိရင်တော့ ဖော်ပြပါအတိုင်း error message ကို ပြန်ထုတ်ပြပေးပါလိမ့်မယ်။

အလုပ်လုပ်သွားပုံကတော့ prompt box မှာ ရိုက်ထည့်လိုက်တဲ့ num တန်ဖိုးကို try statement နဲ့ စစ်ဆေးပါတယ်။ အဲ့ဒီမှာ if statement တွေကို အသုံးပြုပြီး ဘရဲ့ condition က ဘရဲ့ကို သွားဆိုတဲ့ statement ကို throw ကို အသုံးပြုပြီး ညွှန်းထားပါတယ်။ ဒုတိယ statement ကတော့ catch ပေါ့ဗျာ။ ကျနော်တို့ try statement က throw ကိုသုံးပြီး ညွှန်းထားတဲ့ message တွေကို catch statement ထဲမှာ ကြေငြာထားလိုက်ယုံပါပဲ။ အခုလို Try...catch, throw statement တွေကို အသုံးပြုတဲ့အခါမှာ အခု ကျနော်ဖောပြခဲ့သလို Error message တွေကို ကိုယ်တိုင် ဖန်တီး ကြေငြာလို့ရနိုင်သလို window system, softwore system ရဲ့ ခွင့်ပြုထားတဲ့ error code တွေကို ညွှန်းပေးလိုက်ယုံနဲ့ system ရဲ့ error message တွေကို ဖော်ပြနိုင်ပါသေးတယ်။ လေ့လာကြည့်ကြပေါ့ဗျာ။ မိတ်ဆွေအားလုံး လေ့လာခြင်းဖြင့် ကျေနပ်နိုင်ကြပါစေ။

ဒီနေ့တော့ ဒီလောက်နဲ့ပဲ နားကြအုံးစို့ဗျာ။ အားလုံးက အခြေခံလေးတွေ ဖြစ်လို့ စတင်လေ့လာသူများ အတွက်သာ ရည်ရွယ်ပါတရ်။ နောက်နေ့ အပိုင်းလေးမှာတော့ Javascript Function သင်ခန်းစာလေးတွေအကြောင်း ဆွေးနွေးသွားပါ့မရ်။

ဒါကတော့ Javascript အားအသုံးပြုခြင်း အပိုင်း(15)လေးပါ။ အားလုံးက အခြေခံလေးတွေဖြစ်လို့ စတင်လေ့လာသူများ အတွက်သာ ရည်ရွယ်ပါတရ်။ အခုမှ စဖတ်မိတဲ့ မိတ်ဆွေ၊ ညီအစ်ကို၊ မောင်နှမများအနေနဲ့ ရှေ့က ပို့စ်လေးကို အရင် ဖတ်စေချင်ပါတရ်။ ကျနော့် မှားယွင်းတင်ပြမှုများ ရှိရင်လဲ နားလည်ပေးကြပေါ့ဗျာ။ ကျနော် ရည်ရွယ်ချက်ကတော့ အခြေခံလေးတွေကို မြန်မာလို ဖတ်ရှုခြင်းအားဖြင့် လျှင်မြန်စွာ နားလည် သွားနိုင်ပြီး ဒီထပ် မြင့်သော တခြားသော နယ်ပယ်များကို ကူးသွားနိုင်အောင်ပါ။ လေ့လာပြီးသား သူတွေအတွက်လဲ ပြန်လည် အမှတ်ရစေနိုင်မရ်လို့ မျော်လင့်ပါတရ်။ မိတ်ဆွေအပေါင်း လေ့လာခြင်းဖြင့် ကျေနပ်နိုင်ကြပါစေ။

Tay Zar Lin
Koyinmaung007@gmail.com
Programmingknowledge.totalh.com