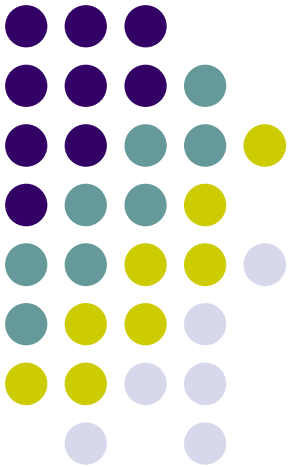
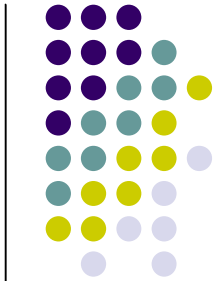
第3回 JavaScriptゼミ



2－6　関数とイベントの処理

発表者：鈴木朋央　発表日06/10/20

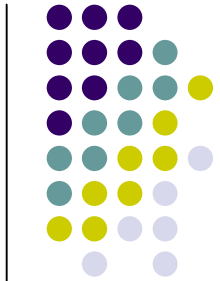
関数の定義方法



```
function 関数名 (引数1, 引数2, ….){
        ステートメント;
        ・・・・
        return 戻り値;
}
```

関数はどこに記述してもよいが、通常ヘッダ部に記述する

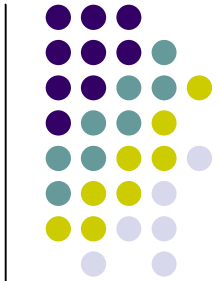
使用例(1/2)



例：ドルを円に換算する関数

```
function exchange(doll, rate){
        var yen;
        yen = doll * rate;
        return yen;
}
```

使用例(2/2)



```
function exchange(doll, rate){
        return doll * rate;
}
```

呼び出し方

```
var dollValue = 5;
var rate = 105;
theYen = exchange(dollValue * 10, rate);
```

変数の有効範囲「スコープ」

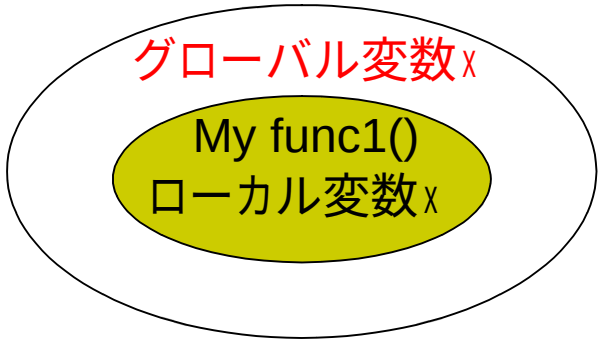
変数には有効範囲があり、それを「スコープ」と呼ぶ

＊ローカル変数：プログラムの一部分で有効

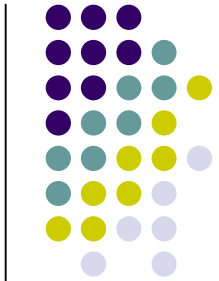
→　varキーワードを使用して関数内で定義した変数

＊グローバル変数：プログラム全体を通して有効

→　関数の外側で定義した変数



引数を格納するarguments配列

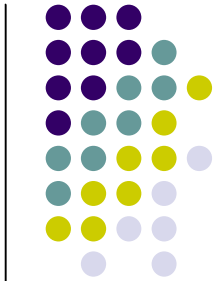


arguments配列には、渡された引数が順に格納される

```
function sum(){
        var result = 0;
        for (var i = 0; i < arguments.length; i++){
                result += arguments[i];
        }
        return result;
}
result = sum(10,100,200,300);
```

例：全ての引数の総和を求める

関数の再起呼び出し

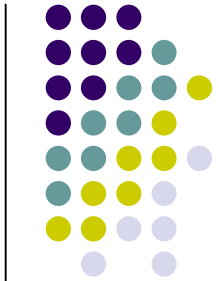


関数の内部で自分自身を呼び出すこと

例：自然数の階乗を求める

```
function fact(n){
        if (n < 1) return null;
        if (n > 1){
                return (n * fact(n – 1));
        } else {
                return 1;
        }
}
```

無名関数

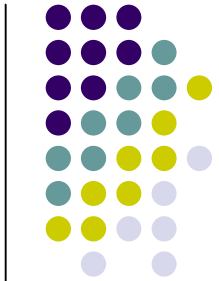


関数へのリファレンスを変数に格納しておいて、その変数名で関数を呼び出すことが出来る

→　これを無名関数と呼ぶ

```
変数A = function ( 引数1,引数2,…..){
        ステートメント;
        ‘‘‘‘
        return 戻り値;
}
```

無名関数の使用例(1/2)



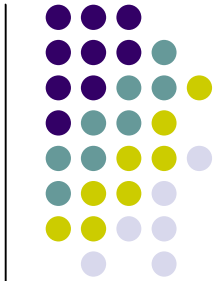
関数の定義部分

```
var exchange = function(doll, rate){
        return doll * rate;
}
```

関数の呼び出し部分

```
var theRate = 113;
var theDoll = 10;
document.write(theDoll, “ドルは”, exchange(theDoll, theRate),”円です<br>”);
var theDoll = 20;
document.write(theDoll, “ドルは”, exchange(theDoll, theRate),”円です<br>”);
```

無名関数の使用例(2/2)

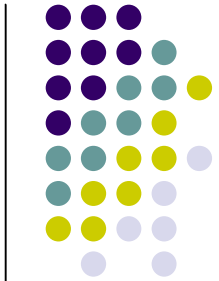


関数の名前がexchangeであるわけではない

別の変数(次の例ではtmp)にexchangeの値を代入すれば、その変数名で無名関数を呼び出すことが出来る

```
var tmp = exchange;
document.write(theDoll,”ドルは”, tmp(theDoll, theRate), “円です<br>”);
```

イベント処理



HTMLドキュメントがロードされる

ユーザがボタンをクリックする

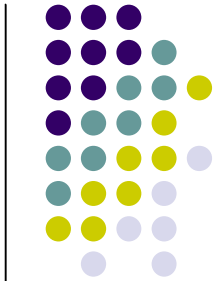
→ イベントが発生

＊イベントを捕まえて処理を行うことを「イベントハンドラ」と呼ぶ

例：ボタンをクリックすると、アラートダイアログボックスが表示

```
<form>
<input type="button" onclick="window.alert('ハンドラのテスト')">
</form>
```

イベントハンドラ



イベント処理の一般的な記述は

イベントハンドラ名＝“実行するステートメント”

＊ステートメントを複数記述するには、「;」で区切る

その他の使用例:

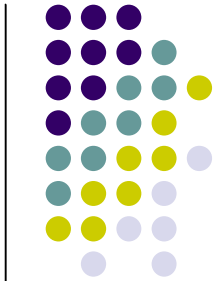
ボタンmyBtnのイベントハンドラonclickが呼び出す関数をfunc2()に設定する

document.myForm.myBtn.onclick = func2;

イベントハンドラの種類(1/2)

イベントハンドラ		イベントが発生する状態
onblur	→	オブジェクトがフォーカスを失ったとき
onfocus	→	オブジェクトがフォーカスを得たとき
onclick	→	オブジェクトがクリックされたとき
ondbclick	→	マウスボタンをダブルクリックしたとき
onmousedown	→	マウスボタンを押したとき
onmouseup	→	マウスボタンを離したとき
onmousemove	→	マウスボタンを動かしたとき
onchange	→	テキストの内容が変更されたとき
onsubmit	→	フォームが送信されたとき
onmouseover	→	オブジェクトの中にマウスが入ったとき

イベントハンドラの種類(2/2)



イベントハンドラ		イベントが発生する状態
onmouseout	→	オブジェクトからマウスが出たとき
onload	→	ドキュメントがロードされたとき
onunload	→	ドキュメントがアンロード（メモリ上から削除）されたとき
onabort	→	イメージのロードが中止されたとき
onreset	→	フォームをリセットしたとき
onerror	→	エラーが起こったとき
onselect	→	テキストが選択されたとき