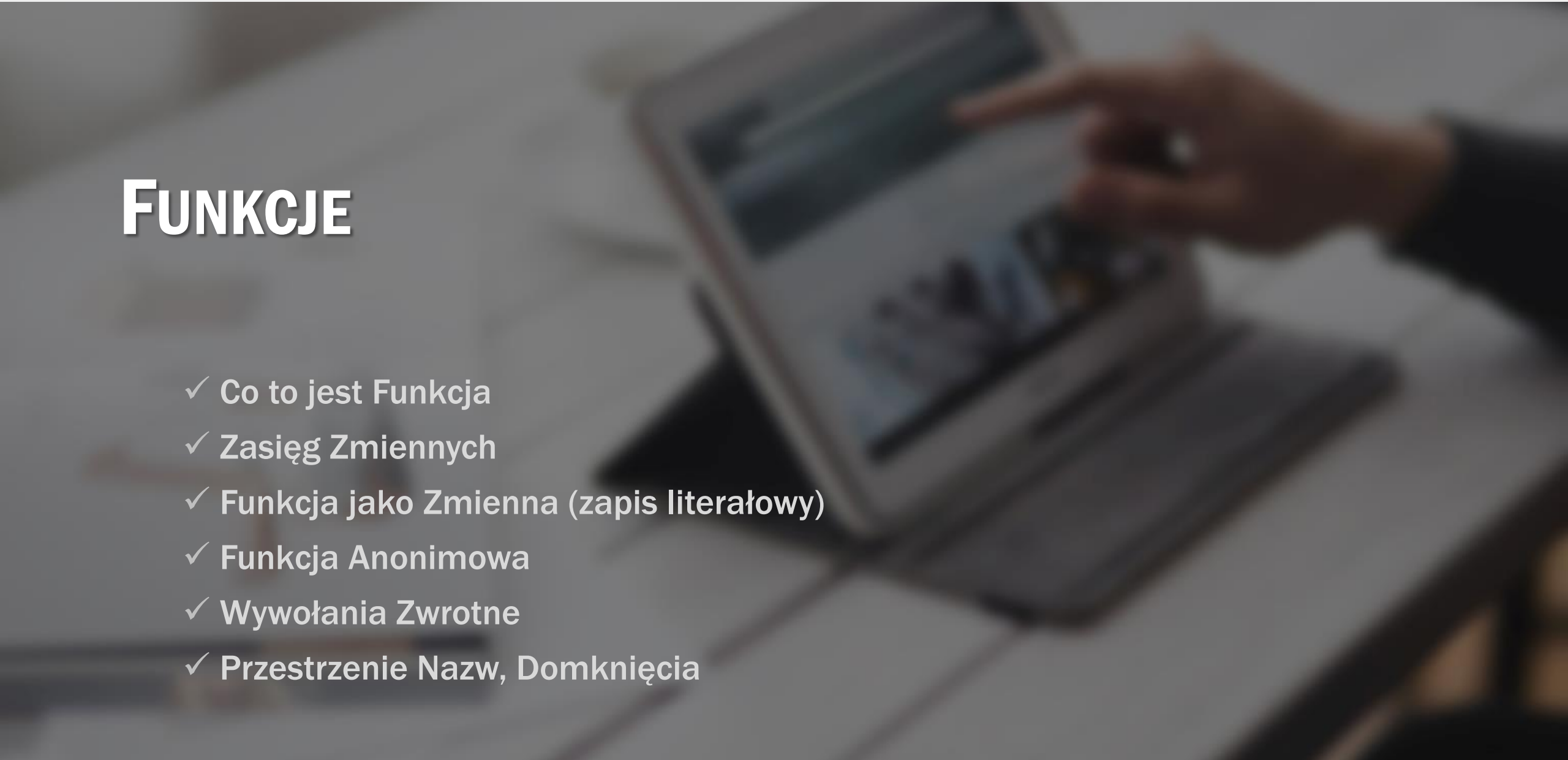
JavaScript - Funkcje

Piotr Mariański
p.marianski@webpros.pl



Funkcje

- ✓ Co to jest Funkcja
- ✓ Zasięg Zmiennych
- ✓ Funkcja jako Zmienna (zapis literałowy)
- ✓ Funkcja Anonimowa
- ✓ Wywołania Zwrotne
- ✓ Przestrzenie Nazw, Domknięcia

JavaScript – Funkcje

Funkcja

obiekt, który dla tego samego **argumentu** zawsze zwraca tą samą **wartość** (odpowiednik funkcji matematycznej). W rozumieniu programistycznym **funkcje** tego typu są nazywane czystymi(ang. pure), nie wykonują one **żadnych dodatkowych czynności**, tzn. nie mają **efektów ubocznych** (ang. side effects).

Procedura

pewien wykonywany **proces**, sposób na obliczenia jakiejś **wartości**.

- **Funkcje** zwracające **ten sam wynik** na podstawie **tych samych danych** ale innego sposobu wykonywania obliczeń z matematycznego punktu widzenia są tą samą **funkcją** ale różnymi **procedurami**.
- W nomenklaturze programistycznej **procedura** to **funkcja**, która nie zwraca **wartości** (choć te pojęcia często stosowane są zamiennie, niezależnie od zwracanego bądź nie wyniku).
- **Funkcje** w **JavaScript** są sposobem na nadanie **nazwy fragmentowi utworzonego kodu** (utworzenie podprogramu), oraz możliwość późniejszego odwoływania się do niego poprzez **użycie zadeklarowanej nazwy** (pozwala to unikać powtarzania tych samych czynności).

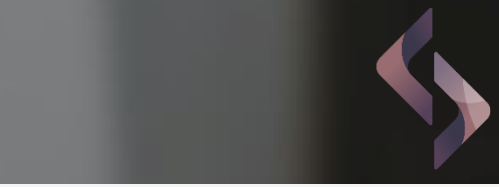
JavaScript – Funkcje

Deklaracja Funkcji

```
function funkcja() { // Deklaracja Funkcji

    var a = Math.round(Math.random() * 100); // Zmienna a
    var b = Math.round(Math.random() * 100); // Zmienna b
    var c = 'A = ' + a + ', B = ' + b + '\n'; // Zmienna c

    if (a > b) { // Instrukcja Warunkowa "if"
        c += 'A jest Większe od B';
    }
    else { // Instrukcja Warunkowa "else"
        c += 'B jest Większe od A';
    }

    return c; // Zwrot Wartości

}
```

Elementy Składowe Funkcji

- Słowo kluczowe **function**.
- Nazwa **funkcji**, w przykładzie jest to "**funkcja**".
- Oczekiwane **parametry** (argumenty). **Funkcja** może posiadać ich zero lub więcej (w tym przypadku ich nie posiada). **Parametry** rozdziela się przecinkami.
- Blok kodu, nazywany **ciałem funkcji**.
- Instrukcja **return**, która umożliwia **zwrócenie obliczonej wartości funkcji**.
- **Funkcja** zawsze **zwraca wartość**. Jeśli nie robi tego w sposób jawny, niejawnie zwraca wartość **undefined**



JavaScript – Funkcje

Deklaracja Funkcji z Parametrami

```
function Params(par1, par2, par3) {

    if (!par1 || !par2 || !par3) {
        /* Nie podano któregoś parametru */
        return 'Nie podano wszystkich parametrów';
    }

    var result;

    result = 'Pierwszy parametr to: ' + par1 + '\n';
    result += 'Drugi parametr to: ' + par2 + '\n';
    result += 'Trzeci parametr to: ' + par3 + '\n';

    /* Zwrot Wartości */
    return result;

}
```

Instrukcja Return

- podaje **wartość** która ma zostać zwrócona z **funkcji**
- powoduje **natychmiastowe przerwanie** wykonywania **funkcji** i przejście do kolejnej po jej wywołaniu **linii w programie**.
- instrukcji **return** wolno użyć **bez** podawania wartości

Wywołanie Funkcji

```
alert(Params(1,2,3)); // Jako Parametr Merody "Alert"

document.write(Params(4,5,6)); // Jako Parametr Merody "Write"

var params = Params(7,8,9); // Jako zmienna Params
```

JavaScript – Funkcje

Zasięg Zmiennych

- **Zmienna** zdefiniowana **wewnątrz funkcji**, **nie jest** widoczna poza jej ciałem i staje się **zmienną lokalną**.
- **Zmienne globalne** to **zmienne** używane **poza funkcjami**, zadeklarowane poza zamkniętą przestrzenią kodu **funkcji**.
- Kod **wewnątrz funkcji** ma dostęp zarówno do **zmiennych globalnych**, jak i do własnych **zmiennych lokalnych**.

```
var global = 1; // Zmienna Globalna

function varRange() {

    var local = 0; // Zmienna Lokalna

    global++; // Zmienna Globalna - Inkrementacja

    return local; // Zwrot Wartości

}

document.write(varRange() + '<br/>'); // Wydrukuje "0"
document.write(global + '<br/>'); // Wydrukuje "2"

document.write(varRange() + '<br/>'); // Wydrukuje "0"
document.write(global + '<br/>'); // Wydrukuje "3"

document.write(local); // Zwróci błąd "local is not defined"
```

JavaScript – Funkcje

Funkcja jako Zmienna

- tzw. zapis **literałowy**

- Jeżeli na **zmiennej**, której została przypisana wartość będąca **funkcją**, wywołamy metodę **typeof()**, jako wynik otrzymamy "**function**".

- **funkcje** w języku **JavaScript** są specjalnym **typem danych**. Posiadają dwie istotne cechy:
  - zawierają w swym ciele **kod**
  - są **wykonywalne** (mogą być wywoływane).

- **funkcja** może być traktowana jak zwykła wartość, którą można przypisać nowej **zmiennej**.

```
var func = function(){
    return 1;
};

document.write('Typ zmiennej "func" to "<b>');
document.write(typeof(func) + '</b>"<br/><br/>');

var sum = function(a, b) {
    return a + b;
};

document.write('Typ zmiennej "sum" to "<b>');
document.write(typeof(sum) + '</b>"<br/><br/>');

var newSum = sum;

document.write('Typ zmiennej "newSum" to "<b>');
document.write(typeof(newSum) + '</b>"<br/><br/>');
```

JavaScript – Funkcje

Funkcja Anonimowa

- **funkcja anonimowa** to taka, która została **zadeklarowana bez identyfikatora** przez który można ją wywołać.
- **funkcję anonimową** można przekazać jako **parametr** do innej **funkcji**. **Funkcja** odbierająca ten **parametr** może przeprowadzić operacje na otrzymanej **funkcji**.
- **funkcje anonimowe** można definiować i **od razu** uruchamiać.
- Mogą służyć jako **zamknięta przestrzeń kodu**

```
var zmienna = function() { // Deklaracja zmiennej jako funkcji
    alert('Funkcja jako zmienna');
};

zmienna(); // Wywołanie Funkcji będącej zmienną


function funcAsReturn() { // Funkcja Zwracająca Funkcję
    return function() { // Zwracana Wartość - Funkcja
        return'Funkcja jako Zwracana Wartość'; // Zwracana Wartość
    };
}

document.write(funcAsReturn()); // Wywołanie - Zwróci Kod Funkcji
document.write('<br/><br/>');

document.write(funcAsReturn()()); // Wywołanie - Wynik Działania
document.write('<br/><br/>');

var zmienna2 = funcAsReturn(); // Deklaracja Zmiennej
document.write(zmienna2()); // Wywołanie - Wynik Działania
document.write('<br/><br/>');
```

JavaScript – Funkcje

CallBack – Wywołanie Zwrotne

callback – **funkcja** przekazywana jako **parametr** innej **funkcji** w celu późniejszego jej wywołania we wnętrzu **funkcji** dla której jest **parametrem**.

kiedy **funkcja** A jest przekazywana jako **parametr** funkcji B, oznacza to że A jest **funkcją zwrotną** (callback function).

callback może być **funkcją** **anonimową**.

```
function param() { // Funkcja, która zostanie przekazana
    alert('jestem callbackiem'); // Kod Funkcji
}


function main(callback) { // Funkcja Główna
    if (typeof(callback) != 'function') {
        return alert('Niewłaściwy Callback');
    }
    return callback(); // Użycie przekazanej funkcji
}

main(param); // Wywołanie funkcji głównej

main(function() { // Parametr - Funkcja anonimowa
    alert('jestem drugim callbackiem'); // Kod Funkcji Anonimowej
}); // Wywołanie funkcji głównej
```

JavaScript – Funkcje

Przestrzenie Nazw

Abstrakcyjne uniwersum, którego elementami mogą być **nazwy**, **terminy techniczne** lub **słowa**. W obrębie dowolnej **przestrzeni nazw** każda nazwa musi być **niepowtarzalna**.

W **Javascript** – takie **izolowanie kodu** aby dodanie go do szeregu bibliotek lub nowego skryptu, nie spowodowało nadpisywania **funkcji** i **zmiennych globalnych**.

Domknięcia (Closures)

Domknięcie występuje wtedy, gdy postawimy pewnego rodzaju granicę, **zamykając wewnątrz** wszystkie obsługiwane i dostępne w obrębie **przestrzeni nazw** **zmienne** i **funkcje**.

```
(function() { // Funkcja Anonimowa Tworząca Izolowaną Przestrzeń

    var locGlob1; // Zmienna dostępna w Pzestrzeni Nazw
    var locGlob2; // Zmienna dostępna w Pzestrzeni Nazw
    locGlob1 = locGlob2;

    function funkcja1(param) { // Funkcja dostępna w Pzestrzeni Nazw
        alert(param);
    }

    funkcja1('Jestem Izolowana!');

}()); // Koniec Izolowanej Przestrzeni - Automatyczne Wywołanie


for (var i = 0; i < 5; i++) {
    var global = 'Instrukcje nie tworzą domknięć!';
}

alert(global);
```

JavaScript - Funkcje

Piotr Mariański
p.marianski@webpros.pl