



ACHIEVEMENT UNLOCKED  
Reach 2 semester

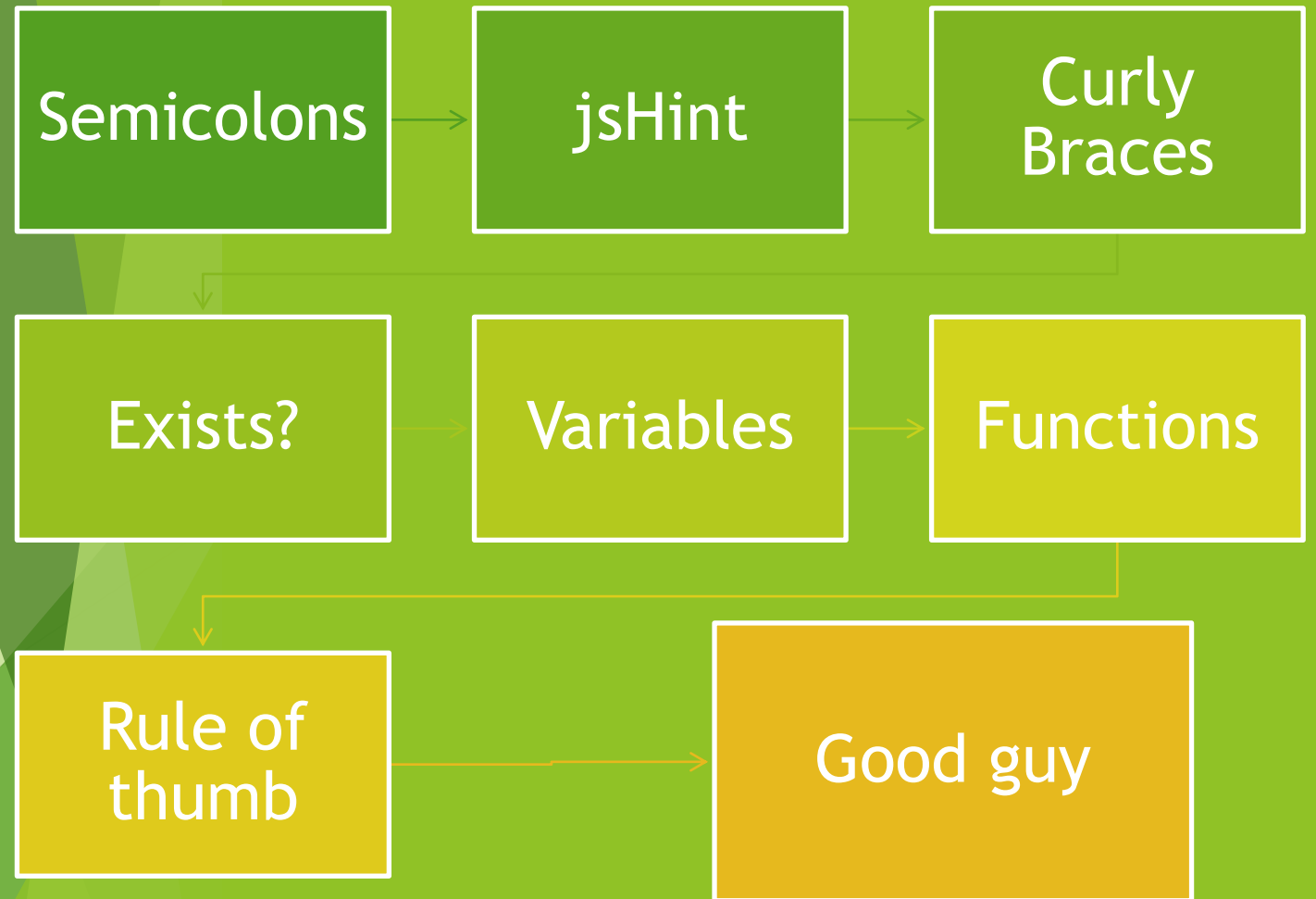


S2 Z1



# Syntax

# Syntax







# Semicolons

“Semicolons  
are optional”

– *the people*



- Ale to chyba ludzie decydują

- To dowiedz się jacy ludzie i porozmawiaj z nimi po swojemu

“Certain ECMAScript statements (variable statement, expression statement, do-while statement, continue statement, break statement, return statement, and throw statement) ***must be terminated*** with semicolons. ”

– *the facts*



“For **convenience**, however, such semicolons may be **omitted** from the source text in certain situations.”

– *the facts*



# RULES

1.

2.

3.





“When, as the program is parsed from **left to right**, a **token** (called the offending token) is encountered that is not allowed by any production of the **grammar**, then a semicolon is automatically **inserted** before the offending token”

– *the facts*



```
var t = 1
```

```
var r = 4
```

```
if(true){console.log(t,r)}
```

14



“When, as the program is parsed from left to right, the **end of the input stream of tokens is encountered**, then a semicolon is automatically **inserted at the end of the input stream**”

– *the facts*

```
var r = 4  
console.log(r)
```

4





```
var x =1;  
var y =5;  
var d = x + y  
[1,2,3].foreach(e => console.log(e))
```

c:\Users\karol\Desktop\PB\JS\2019-2020\Demo\10)Syntax\1)Semicolons.js:16

```
[1,2,3].forEach(e => console.log(e));
```

^

TypeError: Cannot read property 'forEach' of undefined



```
var x =1  
var y =5  
var d = x + y  
[1].forEach(e => console.log(e));
```

c:\Users\karol\Desktop\PB\JS\2019-2020\Demo\10)Syntax\1)Semicolons.js:21

[1].forEach(e => console.log(e));

^

TypeError: Cannot read property 'forEach' of undefined

```
var x =1;  
var y =5;  
var d = x + y  
(function(){  
console.log('call');  
})();
```

c:\Users\karol\Desktop\PB\JS\2019-2020\Demo\10)Syntax\1)Semicolons.js:25

```
var d = x + y  
^
```

TypeError: y is not a function

```
var x = [1,2,3]  
var t = x  
[1].toString()  
console.log(t);
```

Continue, Break, return ...

“When, a token is produced that is allowed by some production of the grammar, but the production is a **restricted production** and the token would be the first token of a restricted production and the restricted token is separated from the previous token by at least one **LineTerminator**, then a semicolon is automatically inserted before the restricted token”

– *the facts*

```
function semicolonTest()  
{  
    return  
    {  
        test: 1  
    }  
}  
  
console.log(semicolonTest());
```

undefined



```
function example()  
{  
  var get = function()  
  {  
    console.log('get');  
  }  
  
  return  
  {  
    get: get  
  }  
}  
console.log(example());
```

undefined

```
function example()
{
  var get = function()
  {
    console.log('get');
  }

  return{
    get: get
  }
}

console.log(example());
example().get();
```

Object {get: }  
get

```
function example(){  
  var get = function(){  
    console.log('get');  
  };  
  
  return{  
    get: get  
  };  
}
```

```
console.log(example());  
example().get();
```

Object {get: }  
get



Exists?

```
var x;  
if(x){  
    console.log('X exists');  
}  
else{  
    console.log('X does not exists');  
}
```

X does not exists



```
var x=1;  
if(x){  
    console.log('X exists');  
}  
else{  
    console.log('X does not exists');  
}
```

X exists

```
var x=0;  
if(x){  
    console.log('X exists');  
}  
else{  
    console.log('X does not exists');  
}
```

X does not exists

```
if(x){  
    console.log('X exists');  
}  
else{  
    console.log('X does not exists');  
}
```

c:\Users\karol\Desktop\PB\JS\2019-2020\Demo\10)Syntax\1)Semicolons.js:115

```
if(x){  
^
```

ReferenceError: x is not defined

```
var x;  
if(typeof x !== 'undefined'){  
    console.log('X exists');  
}  
else{  
    console.log('X does not exists');  
}
```

X does not exists

```
var x=0;  
if(typeof x !== 'undefined'){  
    console.log('X exists');  
}  
else{  
    console.log('X does not exists');  
}
```

X exists

```
if(typeof x !== 'undefined'){  
    console.log('X exists');  
}  
else{  
    console.log('X does not exists');  
}
```

X does not exists

```
var x;  
if(typeof x !== undefined){  
    console.log('X exists');  
}  
else{  
    console.log('X does not exists');  
}  
console.log(typeof x);  
console.log(typeof typeof x);
```

X exists  
undefined  
string



# Variables



“A **var** statement declares variables that are **scoped** to the running execution context's `VariableEnvironment`. Var variables are **created** when their containing Lexical Environment is instantiated and are initialized to **undefined** when created.”

– *the facts*

```
console.log(r);  
let r;
```

ReferenceError: r is not defined

```
console.log(r);  
var r;
```

undefined

```
console.log(r);  
var r=10;  
console.log(r);
```

undefined

10

```
var myVar = 10;
```

```
function myfun(){  
    myVar = 11;  
}
```

```
console.log(myVar);
```

10

```
var myVar = 10;
```

```
function myfun(){  
    myVar = 11;  
}
```

```
myfun();
```

```
console.log(myVar);
```

```
var myVar = 10;
```

```
function myfun(){  
    myVar = 11;  
    var myVar;  
}
```

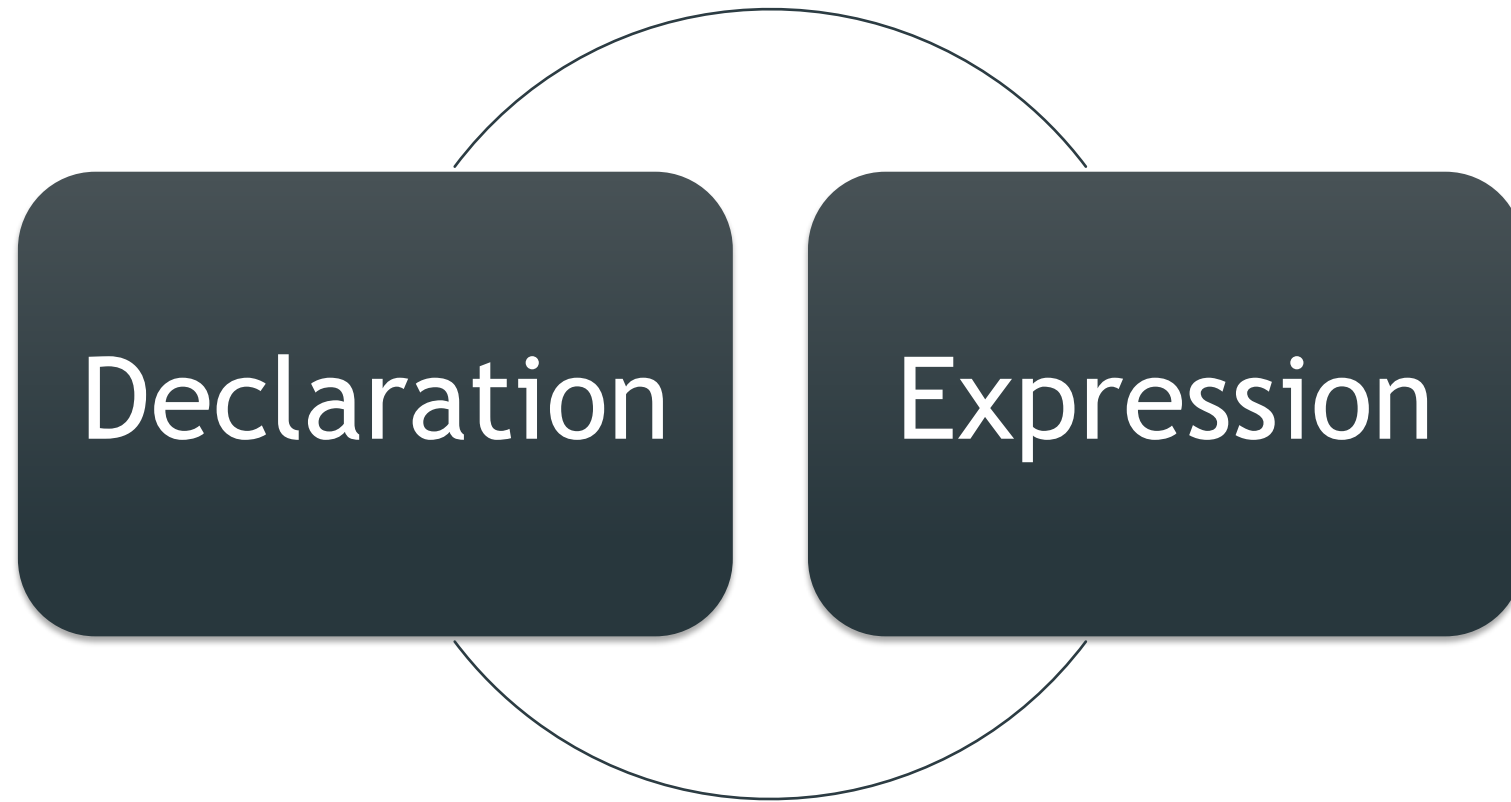
```
myfun();
```

```
console.log(myVar);
```





# Functions



Functions

```
function declarationFunc(){  
    console.log('declarationFunc');  
}
```

```
declarationFunc();
```

declarationFunc

```
declarationFunc();
```

```
function declarationFunc(){  
    console.log('declarationFunc');  
}
```

declarationFunc

```
var expresionFunc = function(){  
    console.log('expresionFunc');  
};
```

```
expresionFunc();
```

expresionFunc



```
expressionFunc();
```

```
var expressionFunc = function(){  
    console.log('expressionFunc');  
};
```

```
expressionFunc();
```

^

TypeError: expressionFunc is not a function

```
var outName = function inName(){  
    // ..  
};
```

```
var outName = function inName() {  
    console.log(inName);  
};
```

```
outName();
```

```
console.log(inName);
```

function inName() { ... }

ReferenceError: inName is not defined

```
var outName = function inName() {  
    inName = 42;  
    console.log(inName);  
};
```

```
outName();
```

```
function inName() { ... }
```

```
askQuestion();  
let studentName = "Suzy";  
function askQuestion() {  
    console.log(`${ studentName }, what do you think?`);  
}
```

```
console.log(`${ studentName }, what do you think?`);
```

^

ReferenceError: studentName is not defined



```
askQuestion();  
var studentName = "Suzy";  
function askQuestion() {  
    console.log(`${ studentName }, what do you think?`);  
}
```

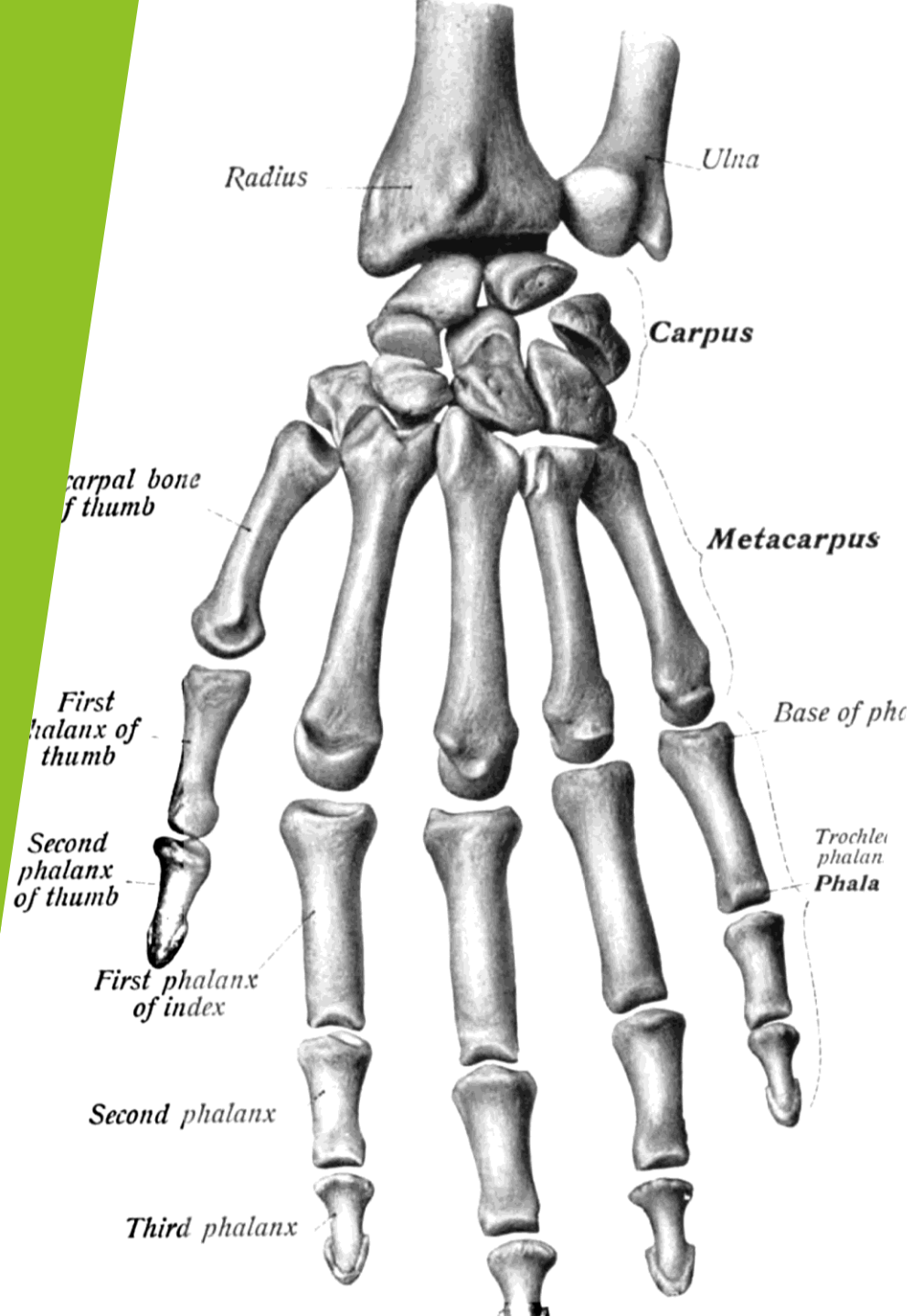
undefined, what do you think?

```
let studentName = "Suzy";  
askQuestion();  
function askQuestion() {  
    console.log(`${ studentName }, what do you think?`);  
}
```

Suzy, what do you think?

# Rule of Thumb

- Variables
- Functions
- Code





Good guy