JavaScript 进阶 - 第1天

学习作用域、变量提升、闭包等语言特征，加深对 JavaScript 的理解，
掌握变量赋值、函数声明的简洁语法，降低代码的冗余度。

理解作用域对程序执行的影响
能够分析程序执行的作用域范围
理解闭包本质，利用闭包创建隔离作用域
了解什么变量提升及函数提升
掌握箭头函数、解析剩余参数等简洁语法

作用域

作用域（scope）规定了变量能够被访问的“范围”，离开了这个“范围”变量便不能被访问，作用域分为全局作用域和局部作用域。

局部作用域分为函数作用域和块作用域。

局部作用域：在函数内部声明的变量只能在函数内部被访问，外部无法直接访问。

块作用域：在 JavaScript 中使用 {} 包裹的代码称为代码块，代码块内部声明的变量外部将【有可能】无法被访问。

函数作用域

在函数内部声明的变量只能在函数内部被访问，外部无法直接访问。

总结：

1. 函数内部声明的变量，在函数外部无法被访问

2. 函数的参数也是函数内部的局部变量

3. 不同函数内部声明的变量无法互相访问

4. 函数执行完毕后，函数内部的变量实际被清空了

块作用域

在 JavaScript 中使用 {} 包裹的代码称为代码块，代码块内部声明的变量外部将【有可能】无法被访问。

JavaScript 中除了变量外还有常量，常量与变量本质的区别是【常量必须要有值且不允许被重新赋值】，常量值为对象时其属性和方法允许重新赋值。

- 总结：
- let 声明的变量会产生块作用域，var 不会产生块作用域
- const 声明的常量也会产生块作用域
- 不同代码块之间的变量无法互相访问
- 推荐使用 let 或 const
- 注：开发中 let 和 const 经常不加区分的使用，如果担心某个值会不小被修改时，则只能使用 const 声明成常量。

全局作用域

<script> 标签和 .js 文件的【最外层】就是所谓的全局作用域，在此声明的变量在函数内部也可以被访问。

- 总结：

- 为 window 对象动态添加的属性默认也是全局的，不推荐！
- 函数中未使用任何关键字声明的变量为全局变量，不推荐！！！
- 尽可能少的声明全局变量，防止全局变量被污染
- JavaScript 中的作用域是程序被执行时的底层机制，了解这一机制有助于规范代码书写习惯，避免因作用域导致的语法错误。

作用域链

作用域链本质上是底层的变量查找机制，在函数被执行时，会优先查找当前函数作用域中查找变量，如果当前作用域查找不到则会依次逐级查找父级作用域直到全局作用域

- 总结：
- 嵌套关系的作用域串联起来形成了作用域链
- 相同作用域链中按着从小到大的规则查找变量
- 子作用域能够访问父作用域，父级作用域无法访问子级作用域（就近原则）

闭包

闭包是一种比较特殊和函数，使用闭包能够访问函数作用域中的变量。从代码形式上看闭包是一个做为返回值的函数

- 总结：
- 闭包本质仍是函数，只不是从函数内部返回的
- 闭包能够创建外部可访问的隔离作用域，避免全局变量污染
- 过度使用闭包可能造成内存泄漏
- 注：回调函数也能访问函数内部的局部变量。

变量提升

变量提升是 JavaScript 中比较“奇怪”的现象，它允许在变量声明之前即被访问，

- 总结：
- 变量在未声明即被访问时会报语法错误
- 变量在声明之前即被访问，变量的值为 undefined
- let 声明的变量不存在变量提升，推荐使用 let【也有人认为具有提升但是不赋值不能使用】
- 变量提升出现在相同作用域当中
- 实际开发中推荐先声明再访问变量

函数提升

函数提升与变量提升比较类似，是指函数在声明之前即可被调用。

- 总结：
- 函数提升能够使函数的声明调用更灵活
- 函数表达式不存在提升的现象
- 函数提升出现在相同作用域当中

参数

函数参数的使用细节，能够提升函数应用的灵活度。

- 总结：
- 声明函数时为形参赋值即为参数的默认值
- 如果参数未自定义默认值时，参数的默认值为 undefined
- 调用函数时没有传入对应实参时，参数的默认值被当做实参传入

动态参数

arguments 是函数内部内置的伪数组变量，它包含了调用函数时传入的所有实参。

- 总结：
- arguments 是一个伪数组
- arguments 的作用是动态获取函数的实参

剩余参数

剩余值写法

- 总结：
- ... 是语法符号，置于最末函数形参之前，用于获取多余的实参
- 借助 ... 获取的剩余实参

箭头函数

箭头函数是一种声明函数的简洁语法，它与普通函数并无本质的区别，差异性更多体现在语法格式上。

- 总结：
- 箭头函数属于表达式函数，因此不存在函数提升
- 箭头函数只有一个参数时可以省略圆括号 ()
- 箭头函数函数体只有一行代码时可以省略花括号 {}，并自动做为返回值被返回
- 箭头函数中没有 arguments，只能使用 ... 动态获取实参

解构赋值

知道解构的语法及分类，使用解构简洁语法快速为变量赋值。

- 解构赋值是一种快速为变量赋值的简洁语法，本质上仍然是为变量赋值，分为数组解构、对象解构两大类型。
- 数组解构
- 对象解构

数组解构

数组解构是将数组的单元值快速批量赋值给一系列变量的简洁语法

- 总结：
- 赋值运算符 = 左侧的 [] 用于批量声明变量，右侧数组的单元值将被赋值给左侧的变量
- 变量的顺序对应数组单元值的位置依次进行赋值操作
- 变量的数量大于单元值数量时，多余的变量将被赋值为 undefined
- 变量的数量小于单元值数量时，可以通过 ... 获取剩余单元值，但只能置于最末位
- 允许初始化变量的默认值，且只有单元值为 undefined 时默认值才会生效
- 注：支持多维解构赋值，比较复杂后续有应用需求时再进一步分析

对象解构

对象解构是将对象属性和方法快速批量赋值给一系列变量的简洁语法

- 总结：
- 赋值运算符 = 左侧的 {} 用于批量声明变量，右侧对象的属性值将被赋值给左侧的变量
- 对象属性的值将被赋值给与属性名相同的变量
- 对象中找不到与变量名一致的属性时变量值为 undefined
- 允许初始化变量的默认值，属性不存在或单元值为 undefined 时默认值才会生效
- 注：支持多维解构赋值，比较复杂后续有应用需求时再进一步分析

传智教育旗下高端IT教育品牌