1、文本溢出

```
overflow:hidden;
text-overflow: ellipsis; // 溢出显示省略号
-webkit-line-clamp: 2;  // 多少行溢出

/* span标签溢出 */
span {
    white-space: nowrap;
    display: block;
    text-overflow: ellipsis;
    overflow: hidden;

}
/* p标签溢出 */
p {
    display: -webkit-box;
    overflow: hidden;
    -webkit-box-orient: vertical;
    -webkit-line-clamp: 3;
}
```

2、Promises

```
Promises.race() // 给一个promises数组，返回数组中最快执行完的promises的结果
Promises.all(Promises Array[]) // 根据传入的Promises数组的所有运行结果绝顶这个Promises的运行结果
```

3、node.js的http模块

```
const http = require("http");  //导入http模块
const serve = http.createServe() // 创建http服务
// 绑定request事件
// req:服务器接收的信息
// res:服务器返回的信息
serve.on("request",(req,res)=>{
    // 设置 utf8 格式代码:
    res.setHeader("Content-type", "text/html;charset=utf8");
    // 返回数据，结束当前请求
    res.end(String)
})
// 设置服务端口号，
serve.listen(8080,()=>{
    console.log("服务已经启动！！！！")
})
```

4、数组转tree

```
function convertToTree(regions, rootId = "0") {
  let resArr = [];
  regions.forEach((element) => {
      // 判断当前节点的父节点是否与rootId相同
    if (element.pid == rootId) {
      resArr.push(element);
      element.children = convertToTree(regions, element.id);
    }
  });
  return resArr;
}
```

5、tree转数组

```
function treeToArray(tree) {
  let result = [];
  function recurse(nodes) {
    nodes.forEach((node) => {
      result.push(node);
      if (node.children) {
        recurse(node.children);
      }
    });
  }
  recurse([tree]);
  return result;
}
```

6、Proxy代理

```
let person = {
  age: 0,
};
// 创建代理实例，第一个参数要代理的对象，第二个参数get,set方法
person = new Proxy(person,{
    // target:代理的对象
    // key:get的属性key
  get(target,key){
    return target.age
  },
    // target:代理的对象
    // key:set的属性key
    // val:修改的值
  set(target,key,val){
    console.log(target,key,val);
    if (val < 0) {
      target[key] = 0
    }else if(val > 150){
      target[key] = 150
    }else{
      target[key] = val
    }
  }
```

```
})
```

7、vuex开启命名空间使用getters和mutations

["模块名/(getters||mutations)名字"]

```
// 没有开启命名空间
store.getters.welcome
store.commit('say', '登录成功，欢迎你回来！')
// 开启命名空间
store.getters["user/welcome"]
store.commit("user/say",'登录成功，欢迎你回来！')
```

8、vue3自定义组件绑定v-model

```
<template>
    <div>
       <input class="form-input" :placeholder="holder" v-model="inputValue"
@input="handleInput" />
    </div>
</template>
<script >
  props: {
    value: String, // v-model 绑定的值，这里取名为 inputValue
  },
  setup(props, { emit }) {
    const inputValue = ref(props.value);

    // 当输入框的值变化时，触发 input 事件更新父组件的 v-model 值
    const handleInput = ()=>{
      emit("update:modelValue",inputValue.value)
    }

    return {
      handleInput,
      inputValue,
    };

  },
};
</script>
```

9、axios请求添加请求头

```
let {data} =  await axios({
        url:'/spelltwo',
        headers:{
            "Authorization":"2b58f9a8-7d73-4a9c-b8a2-9f05d6e8e3c7"
        }
    })
```

10、fs写入数据，不覆盖原来文件

```
fs.appendFile(url,data,cb())
```

11、取消函数

```
/**
 * @param {Generator} generator
 * @return {[Function, Promise]}
 */
var cancellable = function (generator) {
    let cancel = () => {};
    const p = new Promise((resolve, reject) => {
        cancel = (msg = 'Cancelled') => {
            run(msg, 'throw');
        };
        const run = (ret, fnName = 'next') => {
            try {
                const {value, done} = generator[fnName](ret);
                if (done) {
                    resolve(value);
                    return;
                }
                value.then((val) => {
                    run(val);
                }).catch((err) => {
                    run(err, 'throw');
                });
            } catch (errorByGenerator) {
                reject(errorByGenerator);
            }
        };
        run(null);
    });
    return [cancel, p];
};
```

12、手写扁平化数组

```
/**
 * @param {Array} arr
 * @param {number} depth
 * @return {Array}
 */
var flat = function (arr, n) {
    if (n <= 0) return arr
    return [].concat(...arr.map(item => (Array.isArray(item) ? flat(item, n -
1): item)))
};
```

13、最完美判断数据类型的方法

```
Object.prototype.toString(1)  // [object Number]
Object.prototype.toString(true) //[object Boolean]
```