

# 生命周期钩子

生命周期钩子函数分为三个阶段：

- 挂载阶段：beforeCreate、created、beforeMount、mounted；
- 更新阶段：beforeUpdate、updated；
- 卸载阶段：beforeDestroy、destroyed。

在 beforeDestroy 阶段，解除绑定，销毁子组件和事件监听器，清除定时器。

父子组件生命周期钩子，示例代码：

```
<!-- index.vue -->
<template>
  <div>
    <div class="name">姓名: {{ name }}</div>
    <Input :age="age" @changeName="onChangeName" />
  </div>
</template>

<script>
import Input from './input.vue';

export default {
  name: 'LifeCycle',
  data() {
    return {
      age: 18,
      name: ''
    }
  },
  components: {
    Input,
  },
  methods: {
    onChangeName(value) {
      this.name = value;
    }
  },
  beforeCreate() {
    console.log("index beforeCreate");
  },
  created() {
    console.log("index created");
  },
  beforeMount() {
    console.log("index beforeMount");
  },
  mounted() {
    console.log("index mounted");
  }
}
```

```
    },
    beforeUpdate() {
      console.log("index beforeUpdate");
    },
    updated() {
      console.log("index update");
    },
    beforeDestroy() {
      console.log("index beforeDestroy");
    },
    destroyed() {
      console.log("index destroyed");
    }
  }
}
</script>
```

```
<!-- input.vue -->
<template>
  <div>
    <div>{{ age }}</div>
    <input type="text" v-model="name" @change="onChangeName">
  </div>
</template>

<script>
export default {
  name: 'LifeCycleInput',
  props: {
    age: {
      type: Number,
      default: 20
    }
  },
  data() {
    return {
      name: 'li'
    }
  },
  methods: {
    onChangeName() {
      this.$emit('changeName', this.name);
    }
  },
  beforeCreate() {
    console.log("input beforeCreate");
  },
  created() {
    console.log("input created");
  },
  beforeMount() {
    console.log("input beforeMount");
  }
}
```

```
    },
    mounted() {
      console.log("input mounted");
    },
    beforeUpdate() {
      console.log("input beforeUpdate");
    },
    updated() {
      console.log("input update");
    },
    beforeDestroy() {
      console.log("input beforeDestroy");
    },
    destroyed() {
      console.log("input destroyed");
    }
  }
</script>
```

执行顺序流程图：

```
index beforeCreate
index created
index beforeMount
input beforeCreate
input created
input beforeMount
input mounted
index mounted
```

挂载阶段，beforeCreate、created、beforeMount 三个钩子，父组件都在子组件前面，因为只有父组件开始渲染（beforeMount）了，才会有子组件。而子组件渲染完成后，才能算父组件渲染完成，因为有包含关系。

```
input beforeUpdate
input update
index beforeUpdate
index update
```

更新阶段：子组件中 emit 一个事件，传到父组件时，子组件已经更新完成，所以子组件 updated 钩子应该比父组件 beforeUpdate 更早。

```
index beforeDestroy
```

```
input beforeDestroy
```

```
input destroyed
```

```
index destroyed
```

卸载阶段，同上（挂载解读那）的包含关系，只有子组件卸载完成（destroyed）了，父组件才能算是卸载完成。

## 连环问：created 和 mounted 的区别

created 表示实例创建完成，挂载阶段还未开始。

mounted 表示实例已挂载完成，但不保证子组件也挂载完成，如果要等到整个视图都渲染完毕再执行某些操作，可以在 mounted 内部使用 vm.\$nextTick：

```
mounted: function () {  
  this.$nextTick(function () {  
    // 仅在整个视图都被渲染之后才会运行的代码  
  })  
}
```