中南大學
CENTRAL SOUTH UNIVERSITY

第五讲 JavaScript

盛 羽

中南大学计算机学院

shengyu@csu. edu. cn

目录

1. Javascript基础
2. JavaScript 代码块
3. JavaScript 对象
4. JSON
5. Web API
6. Ajax

1.JavaScript基础

- JavaScript（JS）
  - 一种脚本，一门编程语言
  - 通常用于客户端（client-side）的网页动态脚本
  - 也可通过像Node.js这样的包，用于服务器端（server-side）
  - JavaScript 是轻量级解释型语言
- 第一个JS程序
  - 01-first-demo.html
- 本讲内容主要来自MDN Web Docs
  - https://developer.mozilla.org/zh-CN/docs/Learn/CSS/

1.JavaScript基础

- 向页面添加 JavaScript
  - 内部 JavaScript：02-Internal-JavaScript.html
  - 外部JavaScript：03-External-JavaScript.html
  - HTML 元素是按其在页面中出现的次序调用的
    - 如果用 JavaScript 来管理页面上的元素（更精确的说法是使用 文档对象模型 DOM），若 JavaScript 加载于欲操作的 HTML 元素之前，则代码将出错
  - 如何处理?
    - 页面底端
    - async 和 defer
      - async：脚本时不会阻塞页面渲染，而是加载完成时执行的完全独立的脚本。
      - defer：浏览器将继续处理 HTML，构建 DOM。脚本会“在后台”下载，然后等 DOM 构建完成后，脚本才会执行。保持其相对顺序。
  - 注释：04-Comments.html

1.JavaScript基础

■变量

- 变量：用于存放数值的容器
- 声明变量
  - var 或 let 关键字之后加上变量的名字
  - undefined：声明但未赋初值
  - ReferenceError：没有声明
  - var 与 let 的区别
    - var：用于声明一个函数范围或全局范围（任何函数外声明）的变量
    - let：声明一个块级作用域的局部变量
    - var：重复声明
    - var：变量提升，所有的变量声明（仅声明，不含复制）移动到函数或者全局代码的开头位置
    - 建议您在代码中尽可能多地使用 let，而不是 var

```
console.log(bar); // undefined
var bar = 111;
console.log(bar); // 111
```

```
var x = 1;

if (x === 1) {
  var x = 2;
  console.log(x);
  // expected output: 2
}

console.log(x);
// expected output: 2
```

```
let x = 1;

if (x === 1) {
  let x = 2;
  console.log(x);
  // expected output: 2
}

console.log(x);
// expected output: 1
```

中南大學 CENTRAL SOUTH UNIVERSITY

- 变量
  - 命名规则
    - 不要以数字开头
    - 建议使用“小写驼峰命名法”：小写整个命名的第一个字母然后大写剩下单词的首字符（finalOutputValue）
    - 变量名大小写敏感

```
age
myAge
init
initialColor
finalOutputValue
audio1
audio2
```

1.JavaScript基础

中南大学 CENTRAL SOUTH UNIVERSITY

- 变量
  - 变量类型
    - Number
      - 整数（30）、浮点数（2.456）
    - String
      - 字符串
      - 单引号或者双引号把值给包起来
    - Boolean
      - true 或 false
    - Array
      - 方括号括起来，并用逗号分隔
    - Object
      - 对象
    - 动态类型
      - JavaScript 是一种“动态类型语言” 不需要指定变量将包含什么数据类型（例如 number 或 string）

```
let myAge=20;
```

```
let aMsg = 'Hello World';
```

```
let iAmAlive = true;
```

```
let test = 6 < 3;
```

```
let myNameArray = ['Chris', 'Bob', 'Jim’];
let myNumberArray = [10,15,40];

myNameArray[0]; // should return 'Chris’
myNumberArray[2]; // should return 40
```

```
let dog = { name : 'Spot', breed : 'Dalmatian’ };
dog.name
```

```
let myNumber = '500'; // oops, this is still a String
typeof myNumber;
myNumber = 500; // much better – now this is a number
typeof myNumber
```

1.JavaScript基础

- 运算符
  - 算术运算符
    - +、-、*、/、%、**
  - 自增和自减运算符
    - ++、--
  - 赋值运算符
    - =、+=、-=、*=、/=
  - 比较运算符
    - ===、!==、<、>、<=、>=
    - ===：严格测试值和数据类型是否相同

```
let myNum=500;
let myStr='500';

myNum===myStr;// should return false

myNum==myStr;// should return true
```

1.JavaScript基础

- 字符串
  - 连接字符串：+
  - 类型转换
  - 字符串操作
    - 获取长度
    - 获取某个字符（0开始）
    - 获取子串位置
    - 截取子串
    - 大小写转换
    - 字符串替换

```
let myStr='Front ' + 242;
typeof myStr;
```

```
let myString = '123';
let myNum = Number(myString);
typeof myNum;

let myNum = 123;
let myString = myNum.toString();
typeof myString;
```

```
let browserType = 'mozilla;
browserType.length;
```

```
browserType[0];
```

```
browserType.indexOf('zilla');
browserType.indexOf('vanilla');
```

```
browserType.slice(0,3);
browserType.slice(2);
```

```
let radData = 'My NaMe Is MuD';
radData.toLowerCase();
radData.toUpperCase();
```

```
browserType.replace('moz','van');
```

1.JavaScript基础

■数组

- ■ 数组由方括号构成，用逗号分隔
- ■ 访问和修改数组元素
- ■ 获取数组长度
- ■ 字符串和数组之间的转换
- ■ 添加和删除数组项
  - ■ push/pop(尾); unshift/shift(头)

```
let sequence = [1, 1, 2, 3, 5, 8, 13];
let random = ['tree', 795, [0, 1, 2]];
```

```
sequence[2];
sequence[3]=2;
sequence;
```

```
for (let i = 0; i < sequence.length; i++)
{
  console.log(sequence[i]);
}
```

```
let myData = '1,2,3,4,5,6';
let myArray = myData.split(',');
myArray;
```

```
let myNewString = myArray.join(',');
myNewString;
```

```
let dogNames =
["Rocket","Flash","Bella","Slugger"];
dogNames.toString();
//Rocket,Flash,Bella,Slugger
```

```
let myArray = ['1', '2', '3', '4', '5', '6'];
myArray.push('7');
myArray;
myArray.push('8', '9');
myArray;
let removedItem = myArray.pop();
myArray;
removedItem;

myArray.unshift('0');
myArray;
removedItem = myArray.shift();
myArray;
```

目录

1. Javascript基础
2. JavaScript 代码块
3. JavaScript 对象
4. JSON
5. Web API
6. Ajax

2. JavaScript 代码块

条件语句

- if ... else 语句
- && , || 和 !
- 三元运算符

```
let choice,temperature;
choice='sunny';
temperature=28;
if (choice === 'sunny') {
  if (temperature < 30) {
    console.log('nice! ');
  } else if (temperature >= 30) {
    console.log('hot! ');
  }
}
```

```
let choice='sunny',temperature =28;;
if (choice === 'sunny' && temperature < 30) {
  console.log('nice!');
} else if (choice === 'sunny' && temperature >= 30)
{
  console.log('hot!');
}
```

```
if ((x === 5 || y > 3 || z <= 10) && (loggedIn || userName === 'Steve')) {
  // run the code
}
```

```
let choice;
choice='sunny';
(choice==='sunny')?console.log('good!'):console.log('not so good!');
```

2. JavaScript 代码块

中南大學 CENTRAL SOUTH UNIVERSITY

■条件语句

■switch 语句

```
let choice;
choice='sunny';
switch (choice) {
  case 'sunny':
    console.log(choice);
    break;
  case 'rainy':
    console.log(choice);
    break;
  case 'snowing':
    console.log(choice);
    break;
  case 'overcast':
    console.log(choice);
    break;
  default:
    console.log('default');
}
```

2. JavaScript 代码块

- 循环语句
  - for循环
  - while 语句和
  - do ... while 语句
  - 使用 break 退出循环
  - 使用 continue 跳过迭代

```
let cats = ['Bill', 'Jeff', 'Pete', 'Biggles', 'Jasmin'];
let info = 'My cats are called ';

for (let i = 0; i < cats.length; i++) {
  info += cats[i] + ', ';
}
```

```
let i = 0;
while (i < cats.length) {
  if (i === cats.length - 1) {
    info += 'and ' + cats[i] + '. ';
  } else if (i>10){
    break;
  } else if (i===1){
    i++;//不写?
    continue;
  }else{
    info += cats[i] + ', ';
  }
  i++;
}
console.log(info);
```

2. JavaScript 代码块

中南大學 CENTRAL SOUTH UNIVERSITY

■函数

- ■函数定义和调用
- ■函数参数及返回值
- ■匿名函数

```
function myFunction() {
  alert('hello');
}
myFunction();
```

```
function random(number) {
  return Math.floor(Math.random()*number);
}
random(100);
//random('100');
//random('abc');
```

```
var myGreeting = function() {
  alert('hello');
}
myGreeting();
//更多的是用于:
let myButton = document.querySelector('button');
myButton.onclick = function() {
  alert('hello');
}
```

2. JavaScript 代码块

函数

函数作用域和冲突

```
let name = 'Chris';
function greeting() {
  alert('Hello ' + name + ': welcome to our company. ');
}
let name = 'Zaptec';
function greeting() {
  alert('Our company is called ' + name + '.');
}
greeting();
```

2. JavaScript 代码块

中南大學 CENTRAL SOUTH UNIVERSITY

- 事件
  - 事件：系统内发生的动作或者发生的事情
  - 如：
    - 用户在某个元素上点击鼠标或悬停光标。
    - 用户在键盘中按下某个按键。
    - 用户调整浏览器的大小或者关闭浏览器窗口。
    - 一个网页停止加载。
    - 提交表单。
    - 播放、暂停、关闭视频。
    - 发生错误。
  - 事件监听器：用来回应事件被激发的代码块
  - 示例：05-Event.html
    - 事件处理器属性
    - 内联事件处理器
    - addEventListener() 和 removeEventListener()
      - 可以向同一类型的元素添加多个监听器（指定不同的函数）

2. JavaScript 代码块

- 事件
  - 事件对象：event。（06-Event-Target.html）
  - 阻止默认行为（如提交表单）
    - 07-Preventing-Default-Behavior.html
  - 事件冒泡及捕获
    - 08-Event-Bubbling-Capture.html
  - 事件委托
    - 将事件监听器设置在其父节点上，并让子节点上发生的事件冒泡到父节点上，而不是每个子节点单独设置事件监听
    - 09-Event-Delegation.html

目录

1. Javascript基础
2. JavaScript 代码块
3. JavaScript 对象
4. JSON
5. Web API
6. Ajax

3. JavaScript 对象

■基础

- ■对象是一个包含相关数据和方法的集合
  - ■通常由一些变量和函数组成，我们称之为对象里面的属性和方法

```
let person = {
  name : ['Bob', 'Smith'],
  age : 32,
  gender : 'male',
  interests : ['music', 'skiing'],
  bio : function() {
     alert(this.name[0] + ' ' + this.name[1] + ' is ' +
this.age + ' years old. He likes ' + this.interests[0] + ' and
' + this.interests[1] + '.');
   },
  greeting: function() {
    alert('Hi! I\'m ' + this.name[0] + '.');
  }
};
```

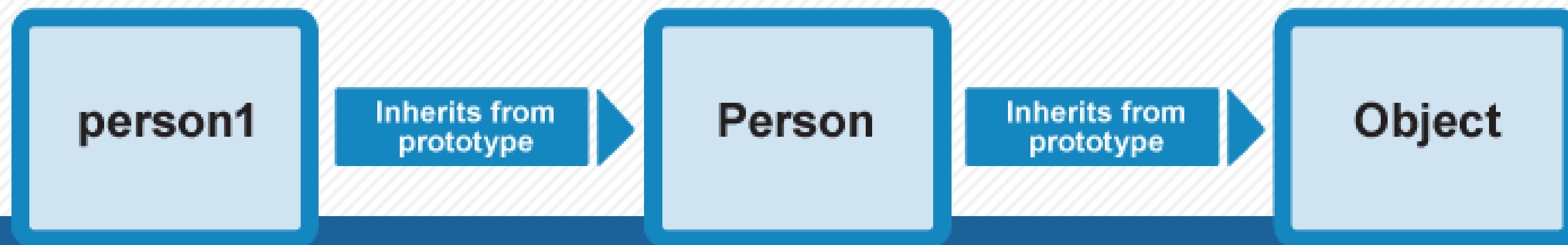
```
person.name[0];
person.age;
person.interests[1];
person.bio();
person.greeting();
```

3. JavaScript 对象

中南大學 CENTRAL SOUTH UNIVERSITY

■对象原型

- ■JavaScript ： 基于原型的语言
- ■对象以其原型为模板、从原型继承方法和属性
- ■原型链：原型对象也可能拥有原型
- ■继承：对象实例和它的构造器之间建立一个链接（它是__proto__属性，是从构造函数的prototype属性派生的），之后通过上溯原型链，在构造器中找到这些属性和方法。
- ■构造函数
  - ■ 它们的命名以大写字母开头。
  - ■ 它们只能由 "new" 操作符来执行。
- ■示例：11-OO-Person.html



3. JavaScript 对象

- 对象原型
  - prototype 属性：继承成员被定义的地方
    - Object.prototype; String.prototype;
  - create()
    - let person2 = Object.create(person1);
  - constructor 属性

3. JavaScript 对象

- 类
  - 使用 class 关键字声明一个类
  - 继承
  - 私有属性/方法：#

```
class Person {

  name;

  constructor(name) {
    this.name = name;
  }

  introduceSelf() {
    console.log(`Hi! I'm ${this.name}`);
  }
}
```

```
class Professor extends Person {
  teaches;
  //覆盖
  constructor(name, teaches) {
    super(name);
    this.teaches = teaches;
   }
  //覆盖
  introduceSelf() {
    console.log(`My name is ${this.name}, and I will be your ${this.teaches} professor.`);
  }
  //新增
  grade(paper) {
    const grade = Math.floor(Math.random() * (5 - 1) + 1);
    console.log(grade);
  }
}
```

目录

1. Javascript基础
2. JavaScript 代码块
3. JavaScript 对象
4. JSON
5. Web API
6. Ajax

4.JSON

- 基本概念
  - JavaScript Object Notation
  - 按照 JavaScript 对象语法的数据格式
  - 基于 JavaScript 语法，独立于 JavaScript
  - JSON 可以作为一个对象或者字符串存在
- 示例
  - 12-JSON-superHeroes.js
- JSON 数组： 13-JSON-array.js
- 其他注意事项
  - JSON 是一种纯数据格式，它只包含属性，没有方法。
  - JSON 要求在字符串和属性名称周围使用双引号。单引号无效
- 示例
  - 14-JSON-heroes.html

目录

1. Javascript基础
2. JavaScript 代码块
3. JavaScript 对象
4. JSON
5. Web API
6. Ajax

5.Web API

■操作文档

- 用Document对象来控制 HTML 和样式信息的文档对象模型（DOM）
- 选择元素并在一个变量中存储对其的引用
  - Document.querySelector()
    - 使用 CSS 选择器选择单个元素
  - Document.querySelectorAll()
    - 文档中每个匹配选择器的元素，存储在一个array中
  - Document.getElementById()
    - 一个id属性值已知的元素

```
<p id="myId">My paragraph</p>
let elementRef = document.getElementById('myId');
```

  - Document.getElementsByTagName()
    - 所有已知类型元素的数组

```
let elementRefArray = document.getElementsByTagName('p');
```

5.Web API

操作文档

创建并放置新的节点

- Document.createElement()
  - 创建一个元素对象
- element.appendChild(aChild)
  - 一个节点附加到指定父节点的子节点列表的末尾处
  - Node.cloneNode、Node.insertBefore

移动和删除元素

- removeChild
- oldChild = node.removeChild(child);
  - child 是要移除的那个子节点。
  - node 是child的父节点。
  - oldChild 保存对删除的子节点的引用。oldChild === child.

```
var sect =
document.querySelector('section');
var para = document.createElement('p');
para.textContent = 'We hope you enjoyed
the ride. ';
sect.appendChild(para);
var text = document.createTextNode(' —
the premier source for web development
knowledge. ');
var linkPara =
document.querySelector('p');
linkPara.appendChild(text);
```

操作文档

操作样式

HTMLElement.style

```
para.style.color = 'white';
para.style.backgroundColor = 'black';
para.style.padding = '10px';
para.style.width = '250px';
para.style.textAlign = 'center';
```

Element.setAttribute()

```
<style>
.highlight {
color: white;
background-color: black;
padding: 10px;
width: 250px;
text-align: center;
}
</style>
```

```
para.setAttribute('class', 'highlight');
```

5.Web API

中南大學 CENTRAL SOUTH UNIVERSITY

Window 对象

- 表示浏览器中打开的窗口
  - 属性：history、innerHeight、innerWidth、screen…
  - 方法：alert()、confirm()、open()、resizeTo()、setInterval()/learInterval()、setTimeout()/clearTimeout()

```
let div = document.querySelector('div');
let WIDTH = window.innerWidth;
let HEIGHT = window.innerHeight;

div.style.width = WIDTH + 'px';
div.style.height = HEIGHT + 'px';

window.onresize = function() {
  WIDTH = window.innerWidth;
  HEIGHT = window.innerHeight;
  div.style.width = WIDTH + 'px’;
  div.style.height = HEIGHT + 'px';
}
```

```
let myVar =
setInterval(function(){ myTimer() }, 1000);

function myTimer() {
  let d = new Date();
  let t = d.toLocaleTimeString();
  document.getElementById("demo").innerHTML
= t;
}
```

中南大学 CENTRAL SOUTH UNIVERSITY

<canvas>

```
<canvas width="320" height="240"></canvas>
```

```
<canvas class="myCanvas">
  <p>添加恰当的反馈信息。</p>
</canvas>
```

```
let canvas = document.querySelector('.myCanvas');
let width = canvas.width = window.innerWidth;
let height = canvas.height = window.innerHeight;
```

```
<style>
  body {
    margin: 0;
    overflow: hidden;
  }
</style>
```

```
var ctx = canvas.getContext('2d');
```

```
ctx.fillStyle = 'rgb(0, 0, 0) ';
ctx.fillRect(0, 0, width, height);
```

中南大學 CENTRAL SOUTH UNIVERSITY

2D 画布基础

CanvasRenderingContext2D 对象

简单矩形：

```
ctx.fillStyle = 'rgb(0, 0, 0) ';
ctx.fillRect(0, 0, width, height);
ctx.fillStyle = 'rgba(255, 0, 255, 0.75) ';
ctx.fillRect(25, 100, 175, 50);
```

描边（stroke）和线条宽度

```
ctx.strokeStyle = 'rgb(255, 255, 255)';
ctx.strokeRect(25, 25, 175, 200);
```

```
ctx.lineWidth = 5;
```

5.Web API

2D 画布基础

绘制路径

- 通过绘制路径来绘制比矩形更复杂的图形
  - beginPath()：在钢笔当前所在位置开始绘制一条路径。在新的画布中，钢笔起始位置为 (0, 0)。
  - moveTo()：将钢笔移动至另一个坐标点，不记录、不留痕迹，只将钢笔“跳”至新位置。
  - fill()：通过为当前所绘制路径的区域填充颜色来绘制一个新的填充形状。
  - stroke()：通过为当前绘制路径的区域描边，来绘制一个只有边框的形状。
- 画线
  - 绘制一个等边三角形

```
ctx.fillStyle = 'rgb(255, 0, 0) ';
ctx.beginPath();
ctx.moveTo(50, 50);
```

```
function degToRad(degrees) {
  return degrees * Math.PI / 180;
};
```

```
ctx.lineTo(150, 50);
var triHeight = 50 * Math.tan(degToRad(60));
ctx.lineTo(100, 50+triHeight);
ctx.lineTo(50, 50);
ctx.fill();
```

中南大學 CENTRAL SOUTH UNIVERSITY

2D 画布基础

绘制路径

画圆：arc()

- 前两个指定圆心的位置坐标，第三个是圆的半径，第四、五个是绘制弧的起、止角度（给定 0° 和 360° 便能绘制一个完整的圆），第六个是绘制方向（false 是顺时针，true 是逆时针）

```
ctx.fillStyle = 'rgb(0, 0, 255) ';
ctx.beginPath();
ctx.arc(150, 106, 50, degToRad(0), degToRad(360), false);
ctx.fill();
```

```
ctx.fillStyle = 'yellow';
ctx.beginPath();
ctx.arc(200, 106, 50, degToRad(-45), degToRad(45), true);
ctx.lineTo(200, 106);
ctx.fill();
```

中南大學 CENTRAL SOUTH UNIVERSITY

2D 画布基础

文本

- fillText()：绘制有填充色的文本
- strokeText()：绘制文本外边框（描边）

```
ctx.strokeStyle = 'white';
ctx.lineWidth = 1;
ctx.font = '36px arial';
ctx.strokeText('Canvas text', 50, 50);

ctx.fillStyle = 'red';
ctx.font = '48px georgia';
ctx.fillText('Canvas text', 50, 150);
```

中南大學 CENTRAL SOUTH UNIVERSITY

2D 画布基础

在画布上绘制图片

- drawImage() 方法可将图片绘制在画布上

```
var image = new Image();
image.src = 'firefox.png';
```

```
image.onload = function() {
  ctx.drawImage(image, 50, 50);
}
```

5.Web API

2D 画布基础

循环

```
ctx.translate(width/2, height/2);
```

```
function degToRad(degrees) {
  return degrees * Math.PI / 180;
};

function rand(min, max) {
  return Math.floor(Math.random() *
(max-min+1)) + (min);
}

var length = 250;
var moveOffset = 20;

for(var i = 0; i < length; i++) {
  //将右侧代码填入
}
```

```
ctx.fillStyle = 'rgba(' + (255-length) + ', 0,
' + (255-length) + ', 0.9) ';
ctx.beginPath();
ctx.moveTo(moveOffset, moveOffset);
ctx.lineTo(moveOffset+length, moveOffset);
var triHeight = length/2 *
Math.tan(degToRad(60));
ctx.lineTo(moveOffset+(length/2),
moveOffset+triHeight);
ctx.lineTo(moveOffset, moveOffset);
ctx.fill();

length--;
moveOffset += 0.7;
ctx.rotate(degToRad(5));
```

5.Web API

2D 画布基础

- 动画
- window.requestAnimationFrame()
  - 它只取一个参数，即每帧要运行的函数名。
  - 下一次浏览器准备好更新屏幕时，将会调用你的函数

```
function loop() {
  ctx.fillStyle = 'rgba(0, 0, 0, 0.25) ';
  ctx.fillRect(0, 0, width, height);
  while(balls.length < 25) {
     var ball = new Ball();
     balls.push(ball);
  }
  for(i = 0; i < balls.length; i++) {
    balls[i].draw();
    balls[i].update();
    balls[i].collisionDetect();
   }
   requestAnimationFrame(loop);
}
loop();
```

5.Web API

- 视频和音频 API
  - HTML5 提供了用于在文档中嵌入富媒体的元素
    - <video>和<audio>
    - 通过自带的 API 来控制视频或音频的播放，定位进度等
    - 创建自定义播放控件
    - 17-Videoplayer.html

- 客户端存储
  - 传统方法：cookies
    - 服务器发送到用户浏览器并保存在本地的一小块数据
    - 浏览器会存储 cookie 并在下次向同一服务器再发起请求时携带并发送到服务器上
    - 功能
      - 会话状态管理：如用户登录状态、购物车、游戏分数或其它需要记录的信息
      - 个性化设置：如用户自定义设置、主题和其他设置
      - 浏览器行为跟踪：如跟踪分析用户行为等
    - Cookie 的生命周期
      - 会话期 Cookie 会在当前的会话结束之后删除
      - 持久性 Cookie 在过期时间（Expires）指定的日期或有效期（Max-Age）指定的一段时间后被删除
    - 限制访问 Cookie
      - Secure
        - 标记为 Secure 的 Cookie 只应通过被 HTTPS 协议加密过的请求发送给服务端
      - HttpOnly
        - JavaScript Document.cookie 无法访问带有 HttpOnly 属性的 cookie；此类 Cookie 仅作用于服务器
      - Domain
        - 指定了哪些主机可以接受 Cookie。如果不指定，该属性默认为同一 host 设置 cookie，不包含子域名
        - 例如，如果设置 Domain=mozilla.org，则 Cookie 也包含在子域名中（如developer.mozilla.org）
      - Path 属性
        - 指定了一个 URL 路径，该 URL 路径必须存在于请求的 URL中，以便发送 Cookie 标头

客户端存储

传统方法：cookies

- Document.cookie：获取并设置与当前文档相关联的 cookie
  - 通过 Document.cookie 属性可创建新的 Cookie。
  - 如果未设置 HttpOnly 标记，你也可以从 JavaScript 访问现有的 Cookie。

```
document.cookie = "yummy_cookie=choco";
document.cookie = "tasty_cookie=strawberry";
console.log(document.cookie);
```

客户端存储

Web Storage

- Web Storage API 提供了一种非常简单的语法，用于存储和检索较小的、由名称和相应值组成的数据项
- sessionStorage
  - 为每一个给定的源（given origin）维持一个独立的存储区域，该存储区域在页面会话期间可用（即只要浏览器处于打开状态，包括页面重新加载和恢复）
  - Window.sessionStorage
- localStorage
  - 同样的功能，但是在浏览器关闭，然后重新打开后数据仍然存在
  - Window.localStorage
- Storage
  - 允许你在一个特定域中设置，检索和删除数据和储存类型 (session or local.)
  - 示例：18-Web-Storage-API-example.html
- Window
  - Web Storage API 继承于Window 对象，并提供两个新属性 — Window.sessionStorage 和 Window.localStorage — 它们分别地提供对当前域的会话和本地Storage 对象的访问。
- StorageEvent
  - 当前页面使用的 storage 被其他页面修改时会触发 StorageEvent 事件

5.Web API

- 客户端存储
  - IndexedDB 是一种底层 API
  - 用于在客户端存储大量的结构化数据（也包括文件/二进制大型对象（blobs））
  - 是一个基于 JavaScript 的面向对象数据库

目录

1. Javascript基础
2. JavaScript 代码块
3. JavaScript 对象
4. JSON
5. Web API
6. Ajax

6.Ajax

Ajax

- AJAX 是异步的 JavaScript 和 XML（Asynchronous JavaScript And XML）
- 使用 XMLHttpRequest 对象与服务器通信
- “异步”特性：可以在不重新刷新页面的情况下与服务器通信，交换数据，或更新页面
- 你可以使用 AJAX 最主要的两个特性做下列事：
  - 在不重新加载页面的情况下发送请求给服务器。
  - 接受并使用从服务器发来的数据。

```
httpRequest = new XMLHttpRequest();
```

```
httpRequest.onreadystatechange = function(){
// Process the server response here.
};
```

```
httpRequest.open('GET', 'http://www.example.org/some.file', true);
httpRequest.send();
```

```
if (httpRequest.status === 200) {
  // Perfect!
} else {
  // There was a problem with the request.
  // For example, the response may have a
404 (Not Found)
  // or 500 (Internal Server Error)
response code.
}
```

19-Ajax-Base-Demo.html

目录

中南大學 CENTRAL SOUTH UNIVERSITY

1. Javascript基础
2. JavaScript 代码块
3. JavaScript 对象
4. JSON
5. Web API
6. Ajax
7. 其他

7.其他

中南大學 CENTRAL SOUTH UNIVERSITY

■箭头函数

■基础语法

```
(param1, param2, …, paramN) => { statements }
(param1, param2, …, paramN) => expression
//相当于：(param1, param2, …, paramN) =>{ return expression; }

// 当只有一个参数时，圆括号是可选的：
(singleParam) => { statements }
singleParam => { statements }

// 没有参数的函数应该写成一对圆括号。
() => { statements }
```

7.其他


```
var elements = [
'Hydrogen',
'Helium',
'Lithium',
'Beryllium'
];

elements.map(function(element) {
return element.length;
}); // 返回数组：[8, 6, 7, 9]

// 上面的普通函数可以改写成如下的箭头函数
elements.map((element) => {
return element.length;
}); // [8, 6, 7, 9]

// 当箭头函数只有一个参数时，可以省略参数的圆括号
elements.map(element => {
return element.length;
}); // [8, 6, 7, 9]
```

```
// 当箭头函数的函数体只有一个 `return` 语句时，可以省略 `return` 关键字和方法体的花括号
elements.map(element => element.length); // [8, 6, 7, 9]

// 在这个例子中，因为我们只需要 `length` 属性，所以可以使用参数解构
// 需要注意的是字符串 `"length"` 是我们想要获得的属性的名称，而 `lengthFooBArX` 则只是个变量名，
// 可以替换成任意合法的变量名
elements.map(({ "length": lengthFooBArX }) => lengthFooBArX); // [8, 6, 7, 9]
```

7.其他

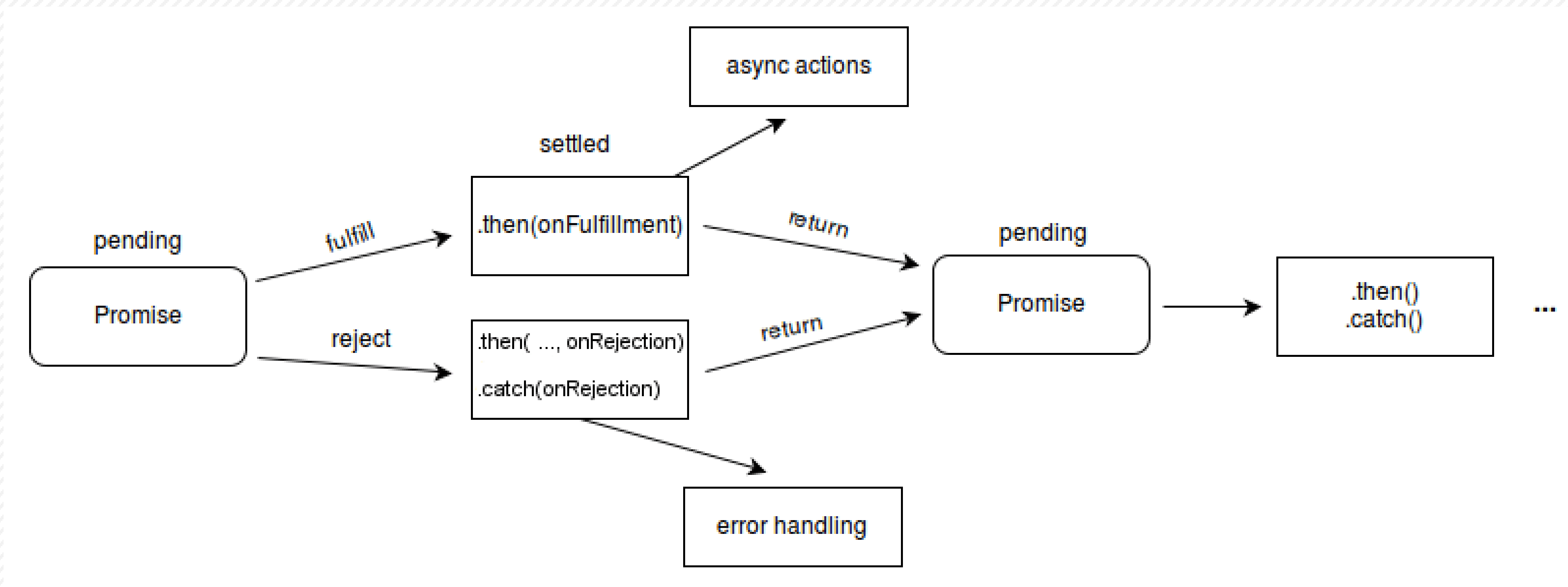
中南大學 CENTRAL SOUTH UNIVERSITY

- Promise
  - 一个 Promise 对象代表一个在这个 promise 被创建出来时不一定已知值的代理
  - 它让你能够把异步操作最终的成功返回值或者失败原因和相应的处理程序关联起来。
  - 这样使得异步方法可以像同步方法那样返回值：异步方法并不会立即返回最终的值，而是会返回一个 promise
  - 一个 Promise 必然处于以下几种状态之一：
    - 待定（pending）：初始状态，既没有被兑现，也没有被拒绝。
    - 已兑现（fulfilled）：意味着操作成功完成。
    - 已拒绝（rejected）：意味着操作失败。

7.其他

Promise

- 因为 Promise.prototype.then 和 Promise.prototype.catch 方法返回的是 promise，所以它们可以被链式调用。



7.其他

Promise

```
var p1 = new Promise(test);
var p2 = p1.then(function (result) {
  console.log('成功: ' + result);
});
var p3 = p2.catch(function (reason) {
   console.log('失败: ' + reason);
});
```

```
new Promise(function (resolve, reject) {
  console.log('start new Promise...’);
  var timeOut = Math.random() * 2;
  console.log('set timeout to: ' + timeOut + ' seconds.’);
   setTimeout(function () {
      if (timeOut < 1) {
        console.log('call resolve()...’);
        resolve('200 OK’);
      } else {
        console.log('call reject()...’);
        reject('timeout in ' + timeOut + ' seconds.’);
      }
    }, timeOut * 1000);
  }).then(function (r) {
     console.log('Done: ' + r);
   }).catch(function (reason) {
     console.log('Failed: ' + reason);
});
```