FONTE

Formas de usar eventos da web

Propriedades do manipulador de eventos

Essas são as propriedades que existem para conter o código do manipulador de eventos.

```
var btn = document.querySelector('button');
btn.onclick = function() {
  var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) +
')';
  document.body.style.backgroundColor = rndCol;}
```

A propriedade `onclick` é a propriedade do manipulador de eventos que está sendo usada nesta situação. É essencialmente uma propriedade como qualquer outra disponível no botão (por exemplo, `btn.textContent`, ou `btn.style`), mas é um tipo especial, quando é configurada para ser igual a algum código, esse código será executado quando o evento é acionado no botão.
Também é possível definir a propriedade handler para ser igual a um nome de função nomeado.

```
var btn = document.querySelector('button');
function bgChange() {
  var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) +
')';
  document.body.style.backgroundColor = rndCol;}
btn.onclick = bgChange;
```

addEventListener() e removeEventListener()

addEventListener() funciona de maneira semelhante às propriedades do manipulador de eventos, mas a sintaxe é obviamente diferente.
Em Vez disso: `btn.onclick = bgChange;`
Isso: `btn.addEventListener('click', bgChange);`

Dentro da função `addEventListener()`, deve ser especificado dois parâmetros, o nome do evento que queremos registrar esse manipulador, e o código que queremos executar em resposta a ele. É perfeitamente apropriado colocar todo o código dentro da função `addEventListener()`, em uma função anônima.

```
btn.addEventListener('click', function() {...});
```

Esse mecanismo tem algumas vantagens sobre os mecanismos mais antigos. Para começar, há uma função de contraparte, `removeEventListener()`, que remove um listener adicionado anteriormente.

Isso não é significativo para programas pequenos e simples, mas para programas maiores e mais complexos, pode melhorar a eficiência limpando antigos manipuladores de eventos não mais utilizados. Além disso, por exemplo, isso permite que o mesmo botão execute ações diferentes em circunstâncias diferentes, só é preciso adicionar / remover manipuladores de eventos conforme apropriado.

Também pode registrar vários manipuladores para o mesmo ouvinte.

```
myElement.addEventListener('click', functionA);
myElement.addEventListener('click', functionB);
```

Outros conceitos de evento

Objetos de evento

Às vezes, dentro de uma função de manipulador de eventos, pode-se ver um parâmetro especificado com um nome como `event`, `evt`, ou simplesmente `e`. Isso é chamado de **event object**, e é passado automaticamente para os manipuladores de eventos para fornecer recursos e informações extras.

```
function bgChange(e) {
  var rndCol ='rgb('+random(255)+','+random(255)+','+random(255)+ ')';
  e.target.style.backgroundColor = rndCol;
  console.log(e);}
btn.addEventListener('click', bgChange);
```

Aqui é incluído um objeto de evento, **e**, na função, e na função definindo um estilo de cor de fundo em `e.target` — que é o próprio botão. A propriedade `target` do objeto de evento é sempre uma referência ao elemento em que o evento acabou de ocorrer. Portanto, neste exemplo, é definido uma cor de fundo aleatória no botão, não na página.

Evitando o comportamento padrão

Evitar o comportamento padrão é interromper a execução do evento atual caso algum erro seja detectado, por exemplo, não enviar um formulário caso alguma coisa esteja em branco ou com um valor inválido.